Progress In Electromagnetics Research B, Vol. 65, 49–63, 2016

Magnetic Induction Tomography with High Performance GPU Implementation

Lu Ma[1], Robert Banasiak[2], and Manuchehr Soleimani[1, *]

Abstract—Magnetic induction tomography (MIT) is a non-invasive medical imaging technique with promising applications such as brain imaging and cryosurgery monitoring. Despite its potential, the realisation of medical MIT application is challenging. The computational complexity of both the forward and inverse problems, and specific MIT hardware design are the major limitations for the development of MIT research in medical imaging. The MIT forward modeling and linear system equations for large scale matrices are computationally expensive. This paper presents the implementation of GPU (graphics processing unit) for both forward and inverse problems in MIT research. For a given MIT mesh geometry composed of 167,488 tetrahedral elements, the GPU accelerated Biot-Savart Law for solving the free space magnetic field and magnetic vector potential is proved to be over 200 times faster compared to the time consumption of a CPU (central processing unit). The linear system equation arising from the forward and inverse problem, can also be accelerated using GPU. Both simulations and experimental results are presented based on a new GPU implementation. Laboratory experimental results are shown for a phantom study representing potential cryosurgery monitoring using an MIT system.

1. INTRODUCTION

Magnetic Induction Tomography (MIT) is the most recent and least developed electrical tomography technique. It does not require electrical contact with the body and instead uses the interaction of an oscillating magnetic field with a conductive medium. The field is excited by inductive coils with an alternating current, which are arranged around the region of interest (ROI). Eddy currents arise when a conductive object is present within the sensing zone, which causes a perturbation to the magnetic field. The passive electromagnetic properties of the object under testing can be reconstructed from the degree of perturbation of this field. It is important to note that MIT is sensitive to conductivity, permeability and permittivity; however, sensitivity to conductivity is usually the primary focus for research [1].

In MIT, the relationship between the field perturbation and the primary magnetic field can be expressed by $\frac{\Delta B}{B} \propto \omega(\omega\varepsilon_0\varepsilon_r - j\sigma)$ [2], where it can be seen that perturbation of the secondary magnetic field is proportional to the square of the frequency. If the object is highly conductive, such as metal, the field perturbation ΔB is dominated by a negative imaginary part. Therefore, the measurements can be taken from the amplitude of the imaginary part, as the real component is negligible. For biomedical applications, the conductivity of biological tissues is many orders of magnitude lower than that of metals and they have no appreciable permeability above that of free space. Therefore, both the real and imaginary parts need to be taken into consideration. This paper presents the reconstruction of a non-conductive inclusion in a low conductive background, which has an electrical conductivity of 1.58 S/m. This might be of interest in medical applications, for example, cryosurgery, where undesired tissues need to be frozen as part of the treatment [3–7]. Human tissues have conductivity values less than 2 S/m; in this context, frozen cells/tissues can be considered non-conductive [8, 9], and as such this

Received 19 October 2015, Accepted 19 December 2015, Scheduled 2 January 2016

* Corresponding author: Manuchehr Soleimani (m.soleimani@bath.ac.uk).
[1] Engineering Tomography Laboratory (ETL), Department of Electronic and Electrical Engineering, University of Bath, BA2 7AY, UK. [2] Institute of Applied Computer Science, Lodz University of Technology, 90-924, Poland.

conductivity contrast can be used for imaging. The aim of this study is to provide insight as to how MIT can be of general use for medical applications as a non-destructive tomographic technique. In this paper, both simulations and experimental results are presented.

In the area of tomographic research, there are usually two problems which need to be solved, the forward problem and the inverse problem. Both problems can be computationally expensive and time consuming, resulting in a growing demand for advanced computing techniques to accelerate the solving process. This could be of particular use in brain/lung function imaging as the finite element model could incorporate up to millions of elements [10–13]. It has also been shown that the calculation of Jacobian matrix is the most expensive computing process in the forward solver [14], especially for 3D tomography approaches. As a result, the implementation of graphics processing units (GPU) has gradually gained popularity [15]. The implementation of GPU techniques has been investigated in the area of both hard-field [16] and soft-field tomography [14] to achieve computational gain, including electrical impedance tomography (EIT) [17] and electrical capacitance tomography (ECT) [18]. Compared to the use of GPU in previous MIT research [19, 20], where the parallelisation of the finite difference (FD) algorithm was achieved, in this work, the time reduction using GPU is realised differently. An edge finite element method (FEM) is used to solve the forward model, which involves generating the meshes, solving the magnetic vector potential and fields in every single tetrahedral element, as well as solving the system linear equation which is a large scale sparse matrices calculation. In the algorithm section, a high performance computation is achieved using GPU implementation, which improved the computational efficiency in solving both MIT forward and inverse problems.

2. METHODS

2.1. Measurement System

The MIT system used for this study consists of (i) a coil array of 16 equally spaced air-core sensors, (ii) a National Instrument (NI) based data acquisition system and (iii) a host computer. The sensing zone has a diameter of approximate 25 cm, each coil has 6 turns, a side length of 1 cm, and a radius of 2 cm. Figure 1 shows the medical MIT system used in this study. Among 16 coils, 8 coils are engaged for transmitting signals, and the remaining 8 coils are dedicated for receiving signals; the total number of independent measurements is therefore 64. A grounded aluminium cylinder is used to prevent external electric field perturbation. As the signal perturbation caused by the secondary magnetic field is usually low for a MIT system, the measurement system requires accurate measurement devices to improve the sensitivity. A *NI2953* is employed to realise the multi-channel switching process. The measured data are collected, combined and displayed through a *NI5781* device. In order to improve the system efficiency, a *NI7951* FlexRIO board is utilized to accelerate the data acquisition process. An excitation frequency of 13 MHz is chosen for this system. The sampling rate of this system is 100 megasamples/s. The image reconstruction module extracts 64 independent measurements to perform the reconstruction algorithm, displays and updates the images through the LabView and Matlab programs. A detailed description of the system design and evaluation of driving signal level, phase noise and phase drift have been reported in [21].

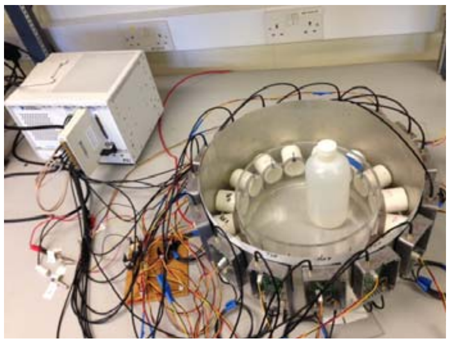

Figure 1. Bath-MK II medical magnetic induction tomography system.

2.2. Forward and Inverse Problem

In this study, a reduced magnetic vector potential is used to avoid modelling the structure of the coils using edge FEM [22, 23]. In the (A, A) formulation we have:

$$\nabla \times \frac{1}{\mu} \nabla \times A + j\sigma\omega A = J_s \tag{1}$$

where the current density, J_s, can be prescribed by the magnetic vector potential according to the Biot-Savart Law.

The magnetic potential constitutes two parts $A = A_s + A_r$, where A_s represents the impressed magnetic vector potential as a result of current source J_s, and A_r represents the reduced magnetic vector potential in the eddy current region Ω_e.

In the non-conductive region Ω_n:

$$\nabla \times H_s = J_s \tag{2}$$

where H_s is the magnetic field generated by an excitation coil, which can be directly computed from any point P in a free space from J_s:

$$H_s = \int_{\Omega_n} \frac{J_s(Q) \times r_{QP}}{4\pi |r_{QP}|^3} d\Omega_Q \tag{3}$$

where r_{QP} is the vector pointing from the source point Q to the field point P.

The impressed magnetic vector potential A_s can be written by:

$$\nabla \times A_s = \mu_0 H_s \tag{4}$$

From Equation (3) and Equation (4), A_s is readily shown as:

$$A_s = \int_{\Omega_n} \frac{\mu_0 J_s(Q)}{4\pi |r_{QP}|^2} d\Omega_Q \tag{5}$$

In the entire region $\Omega_n + \Omega_e$, using time harmonic and complex notation, the magnetic flux density can be expressed by:

$$B = \mu_0 H_s + \nabla \times Ar \tag{6}$$

According to Faraday's Law of induction, ignoring the displacement current, the induced electric field can be written in the following form:

$$\nabla \times E = -j\omega B \tag{7}$$

From Equation (6) and Equation (7), the induced electric field in the entire region is:

$$E = -j\omega A_s - j\omega A_r \tag{8}$$

Eddy currents arise when a conductor is exposed in a time varying magnetic field; therefore, in the eddy current region Ω_e:

$$J_{\text{eddy}} = \sigma E \tag{9}$$

In the region of $\Omega_n + \Omega_e$, the magnetic field can be expressed by:

$$\nabla \times H = J_{\text{eddy}} + J_s \tag{10}$$

It is known that the relationship between B and H is:

$$B = \mu H \tag{11}$$

where μ is the permeability of the medium.

Combining Equations (2), (6), (8), (10) and (11), we have:

$$\nabla \times \left(\frac{1}{\mu} \nabla \times A_r \right) + j\omega\sigma A_r = \nabla \times H_s - j\omega\sigma A_s - \nabla \times \frac{1}{\mu} \mu_0 H_s \tag{12}$$

In MIT, the inductive coils are considered magneto-static, not antennas; as such the wave propagation effect can be ignored. Edge FEM is a useful technique to solve such problems by approximating the system as a combination of linear equations in small elements with appropriate boundary conditions. In edge FEM on a tetrahedral mesh, a vector field is represented using a basis vector function N_{ij} associated with the edge between nodes i and j:

$$N_{ij} = L_i \nabla L_j - L_j \nabla L_i \tag{13}$$

where L_i is a nodal shape function. Applying the edge element basis function to Galerkin's approximation [24, 25], one can obtain:

$$\begin{aligned}\int_{\Omega_e} \left(\nabla \times N \cdot \frac{1}{\mu} \nabla \times A_r \right) dv + \int_{\Omega_e} j\omega\sigma N \cdot A_r dv &= \int_{\Omega_c} \left(\nabla \times N \cdot H_s \right) dv - \int_{\Omega_c} \left(N \cdot j\omega\sigma A_s \right) dv \\ &\quad - \int_{\Omega_c} \left(\frac{1}{\mu} \mu_0 \nabla \times N \cdot H_s \right) dv \end{aligned} \tag{14}$$

where N is any linear combination of edge basis functions, Ω_e the eddy current region, and Ω_c the coil region. It can be seen from Equation (14) that the right hand side can be solved by Equations (3) and (5), the only unknown variable is the reduced vector potential A_r. By applying edge FEM, the second order partial differential equations can be computed by a combination of system linear equations, which can then be processed through Matlab. A_r can be obtained using the BiConjugate Gradients Stabilized Method to solve the system linear equation [26]:

$$SA_r = b \tag{15}$$

where $S = S_r + jS_i$, which is a complex matrix; b is the current density, which is also complex. By solving the reduced magnetic vector potential A_r, one is able to evaluate the induced voltages in the measuring coils. The induced voltages can be calculated by using a volume integration form:

$$V_{mn} = -j\omega \int_{\Omega_c} A \cdot J_0 dv \tag{16}$$

where $A = A_s + A_r$, and J_0 is a unit current density passing through the coil.

The sensitivity matrix is essential in MIT as it realises the linearisation between the conductivity and the induced voltages. The elements of the Jacobian matrix can be expressed by [27, 28]:

$$\frac{\partial V_{mn}}{\partial \sigma_k} = -\omega^2 \frac{\int_{\Omega_k} A_m \cdot A_n dv}{I_0} \tag{17}$$

where V_{mn} is the measured voltage, σ_k the conductivity of pixel k, and Ω_k the volume of the perturbation (pixel k). A_m and A_n are respectively solutions of the forward solver when the excitation coil m is excited by I_0 and the sensing coil n excited with unit current. It is worth mentioning that if the object under testing has a significant relative permittivity ε_r, the imaginary term $j\omega\varepsilon_0\varepsilon_r$ must be added to the complex conductivity. In this case, the complex conductivity can be written as $\xi = \sigma + j\omega\varepsilon_0\varepsilon_r$.

The inverse problem usually makes use of the forward model as part of the solving process. Let the forward model be expressed in a linear approximation.

$$V = J\sigma \tag{18}$$

where V is a column vector consisting of M voltage measurements and σ also a column vector representing K pixels. J is an $M \times K$ matrix of the sensitivity map, calculated from Equation (17). The inverse problem is solved to compute the distribution of the conductivity through measured voltages. In most cases, this is an underdetermined problem as the number of measured voltages is far less than the number of pixels, resulting in the solution being ill-posed. In order to ensure the uniqueness and the stability of the solution, the least square solution (LSS) is applied.

$$\min \parallel V - J\sigma \parallel^2 = \min (V - J\sigma)^T (V - J\sigma) \tag{19}$$

$$\frac{\partial}{\partial \sigma} \left[(V - J\sigma)^T (V - J\sigma) \right] = -2J^T (V - J\sigma) \tag{20}$$

$$\sigma = (J^T J)^{-1} J^T V \tag{21}$$

Although the LSS guarantees the uniqueness of the solution, it does not ensure the stability. A regularisation function can be added to ensure the accuracy and robustness of the solution. In this paper, the Tikhonov regularisation method is used to solve the inverse problem, where the reconstruction relies on the fact that for small changes, the measurements can be approximated in a linear fashion with a regularisation parameter α and regularisation matrix R [29].

$$\sigma = \left(J^T J + \alpha R\right)^{-1} J^T V \tag{22}$$

2.3. GPU High Performance Computing

2.3.1. Parallelisation Scheme

The computation of both forward and inverse problems for MIT is time consuming, and as such, there is a pressing need to accelerate it. One approach is to adapt and convert MIT data processing into parallel computing architectures. There are two possible ways to achieve this goal: the classic parallel solution, i.e., running on multiple CPU (central processing unit) cores, and the GPU (graphics processing units) solution. The former uses multi-node PC (personal computer) systems to perform one global task by splitting it into subtasks. This method is well established, requiring a specialised software, a low-latency network infrastructure to exchange data between nodes and an increased hardware complexity. The latter uses the parallel architecture of GPUs, through a novel technique called GP GPU (General-Purpose computing on Graphics Processing Units) (http://gpgpu.org).

A growing interest in GPU computation began in general with the difficulty in clocking CPUs above a certain level due to the limitations of silicon-based transistor technology. Although sequential programs are easier to understand, the growing demand for improvements in modern computing led to an increased interest in multi-core technology and parallel architectures [30, 31]. Nevertheless, even with parallel architectures, there is still a limit as to how fast programs can run. Speed increases are now based primarily on improving the architecture of the CPU rather than higher clock rates [32]. Until recently, parallel programming was reserved for high performance clusters with multiple processors, as the cost was often prohibitively expensive. However, this changed with the introduction of new types of multi-core processor to the mainstream market, including GPUs.

Compared to CPUs which today have a maximum of 2 to 12 cores, GPUs consist of dozens, or even hundreds, of smaller, simpler cores designed for a high level of parallelisation (Figure 2).

CPUs are designed to execute a large set of instructions (which often includes a high level of branching) as fast as possible, while GPUs mostly take care of computations using a more limited instruction set with less branching. For this reason their execution units, or cores, can be much simpler, i.e., smaller. Therefore, a single chip may have dozens or even hundreds of cores on it. Moreover, graphics computations can be easily parallelised in GPU work based on a SIMD (Single Instruction, Multiple Data), or more precisely SIMT (Single Instruction, Multiple Thread) architecture, where the

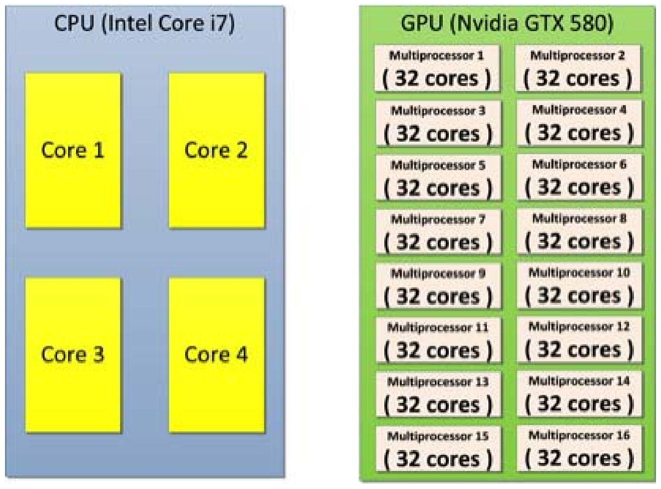
CPU (Intel Core i7)
Core 1
Core 2
Core 3
Core 4
GPU (Nvidia GTX 580)
Multiprocessor 1 (32 cores)
Multiprocessor 2 (32 cores)
Multiprocessor 3 (32 cores)
Multiprocessor 4 (32 cores)
Multiprocessor 5 (32 cores)
Multiprocessor 6 (32 cores)
Multiprocessor 7 (32 cores)
Multiprocessor 8 (32 cores)
Multiprocessor 9 (32 cores)
Multiprocessor 10 (32 cores)
Multiprocessor 11 (32 cores)
Multiprocessor 12 (32 cores)
Multiprocessor 13 (32 cores)
Multiprocessor 14 (32 cores)
Multiprocessor 15 (32 cores)
Multiprocessor 16 (32 cores)

Figure 2. The difference in parallel architecture between CPU (Intel i7) and GPU (Nvidia GTX 580).

same instruction can be applied to numerous subsets of data, in effect allowing a large dataset to be processed rapidly. This allows a much higher number of operations per second than can otherwise be achieved on a CPU. Many time consuming computations can now run in close to real-time with a smaller investment in hardware compared to before the advent of GP GPU.

A real breakthrough in the ease-of-use of GP GPU came with the introduction of Nvidia CUDA (Compute Unified Device Architecture) in 2007 (http://docs.nvidia.com). Programmers could then use generic software libraries to exploit the potential of the massively parallel GPU architecture and reduce the running time of many algorithms (http://docs.nvidia.com/cuda/cuda-c-programming-guide/). Nevertheless, GPU computation is not without its limitations. One of the key challenges is that GPUs were originally designed for parallel operations, and as such they will not do well on sequential tasks, or those that involve a lot of branching [30].

In this work, the AccelerEyes Jacket GPU processing toolbox is used to run the calculations in parallel. The Jacket GPU library is a mature solution that uses Nvidia's CUDA parallel computing libraries and enables the writing and running of code on the GPU in a way similar to how it would be done in the Matlab language. Jacket accomplishes this by automatically wrapping the M-language into a GPU CUDA-compatible form by adapting input data to Jacket's GPU data structure, Matlab functions can be easily transformed into GPU functions. Jacket also preserves the interpretive nature of the M-language by providing real-time transparent access to the GPU compiler, which increases the efficiency of parallel computing code. A new GPU computing library is also available to be embedded in the standard Matlab engine; however, it is currently at an early stage of development and therefore relatively limited.

2.3.2. GPU Accelerated MIT forward Solver

High time consumption due to the expensive computation of the solving process has put constraints on the CPU-based MIT forward solver (Equations (3) and (5)). The magnetic field and magnetic vector potential need to be calculated for every single tetrahedral element, and then integrated over all the elements for a given mesh. This needs to be repeated for every single coil in the MIT forward model (Figure 3). This is a very repetitive process, particularly for a very dense mesh, and as such is both time consuming and computationally expensive. However, all other steps of the MIT forward solution can be efficiently accelerated using a Matlab GPU-based approach.

In this work, a new hybrid type of Biot-Savart equation-solving infrastructure is developed which performs parallel computations using a GPU. The Biot-Savart calculations for Equations (3) and (5) are computed for each individual tetrahedral element in the MIT finite element model without depending on the results of other elements and therefore these can be performed in parallel. Figure 3 depicts this process using a flow chart. Note that there are three main, but completely different in their nature, sequential loops. To properly adapt this kind of iterative algorithm into a GPU parallel scheme, one first

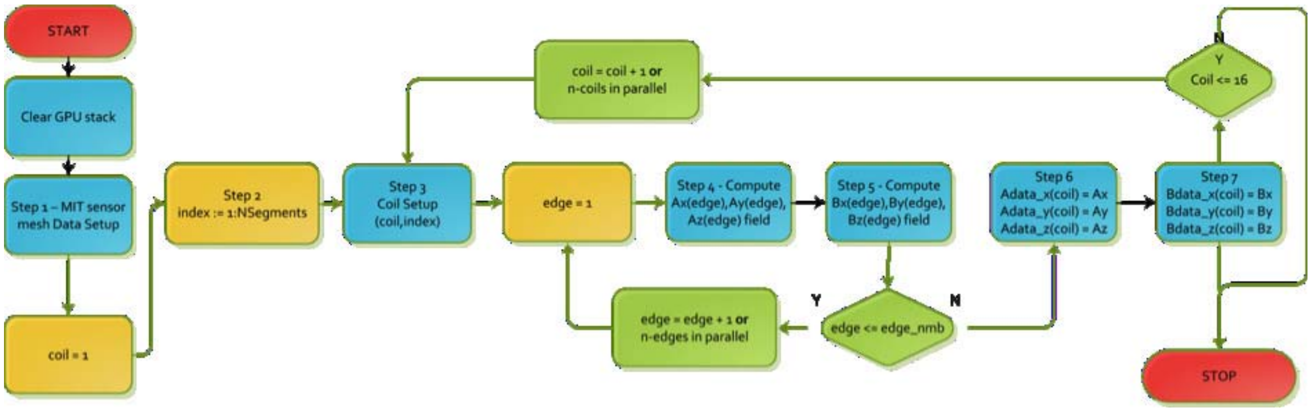
START
Clear GPU stack
Step 1 – MIT sensor mesh Data Setup
coil = 1
Step 2 index := 1:NSegments
Step 3 Coil Setup (coil,index)
coil = coil + 1 or n-coils in parallel
edge = 1
Step 4 - Compute Ax(edge),Ay(edge), Az(edge) field
Step 5 - Compute Bx(edge),By(edge), Bz(edge) field
edge <= edge_nmb
Y
N
edge = edge + 1 or n-edges in parallel
Step 6 Adata_x(coil) = Ax Adata_y(coil) = Ay Adata_z(coil) = Az
Step 7 Bdata_x(coil) = Bx Bdata_y(coil) = By Bdata_z(coil) = Bz
Coil <= 16
N
Y
STOP

Figure 3. The flow diagram showing the main steps performed while solving MIT Biot-Savart equations.

needs to consider the limitations of using the Jacket GPU library. The major drawback is that nested loops are not currently supported. This means that the selection of each loop must be done according to the GPU highly scalable parallel architecture to achieve computational gain. The outermost loop (Steps 1–2) represents a typical CPU-type loop. It computes n number of coil segments for an assigned index, where n is typically much smaller than the total number of elements in a given mesh. In this case, the use of Matlab native CPU vectorisation is an optimal approach because the expense of marshaling the data onto the GPU and retrieving it would not be paid off for a small number of segments (e.g., 2000–5000 segments).

There are two remaining loops that can be adapted into a Jacket parallel scheme (Steps 3–7). The outer loop iterates over the number of coils — here called the "coils" loop — with the number of coils defined as N_c (Step 3). This loop includes a second inner loop — called the "edges" loop — defined as a stream of matrix linear algebra computations (additions, subtractions and multiplications) that need to be calculated for a given mesh geometry for a given coil number from the outer loop. The edges loop iterates over a large number of tetrahedral edges that correspond to the density of a given mesh.

There are two choices to accelerate the computation: parallelising either the coils loop or the edges loop. Parallelising the coils loop is an inefficient means of acceleration as each coils loop contains embedded edges loops. In our MIT system, the number of coils, N_c, is 16, but there are hundreds of thousands of edges. Parallelising the coils loop in this manner can be accomplished just as efficiently as using CPU vector operations. As the GPU computing architecture is internally optimised for a large number of parallel calculations (many cores and many threads), the inner edges loop is selected for GPU parallelisation.

With regards to selecting the edges loop for parallelisation, there are two methods to achieve this goal. One solution is to predefine a Jacket GPU type index variable that iterates over the number of edges, using this predefined index for all variables that need to be iterated in this way. The Jacket GPU vectorisation mechanism is able to use this index to parallelise all algebraic functions and process edge data by hundreds of separate computing threads (and physical cores). Another solution is the use of the in-built Jacket GPU type loop, which looks similar to classic Matlab "for-end" type loop. The difference is that all the GPU loop iterations will run in separate computing threads. To use the GPU loop, all the internal MIT physics functions (Steps 4–7, A, B magnetic field properties computing, edge-based function specified in Figure 3) and edge data must use the same loop (edge number) iterator, which must be defined as a GPU type variable at the beginning of the loop. This approach is the most efficient way of using GPU parallel architecture. One difficulty associated with implementing this innovative computation method is that the GPU code requires that all processing data be transferred into GPU memory at the beginning of the computation, and transferred back to CPU memory at the end of the computation. Repetitious conversion of variables between CPU and GPU type "on-the-fly" (inside loops) is computationally inefficient and can significantly increase the processing time according to the scale of MIT forward and inverse mesh. To avoid this problem, the GPU memory needs to be cleaned and all the data that are going to be parallelised need to be preloaded once into GPU memory before any loop starts.

Table 1 shows the comparison of MIT forward problem processing times for CPU (Inter Core i7 2.8 GHz) and Tesla C2070 GPU. For a mesh geometry composed of 167,488 tetrahedral elements, we achieved a 200x speedup when using a GPU compared to using a CPU alone. It is important to note that these speedups (shown in Table 1) were critically evaluated with careful profiling of CPU and GPU programs and in particular in comparison with multi-CPU systems. This way the computational gain of using GPU can be illustrated, and this speedup increases for denser meshes. In this study, the coil arrangement is fixed, thus the Biot-Savart calculation for the free space field is calculated once; however, for a MIT system where the coil geometry is changing, such as in a rotational MIT system, the coils are rotated according to the rotational scheme, and the forward model needs to be updated accordingly. In this context, the implementation of GPU techniques becomes more advantageous. It is also favorable to use GPU-accelerated Biot-Savart calculations in the forward model when a MIT system has a large number of coils, for instance, 256 coils.

Another computational challenge is in solving the linear system equation (Equation (15)), especially for large scale MIT meshes. This matrix equation is part of the edge FEM algorithm and its solution requires matrix inversion, a time consuming process for sequentially operating processors. The desire

Table 1. Time consumption using CPU and GPU for Biot-Savart calculation for different mesh densities.

Nmb.	Mesh density	Biot-Savart Law Intel Core i7, 2.8 GHz Elapsed time (s)	Biot-Savart Law Tesla C2070 Elapsed time (s)	Gain
1	1797	13.96	0.45	31.02
2	119280	1571.30	11.24	139.80
3	167488	3374.82	16.75	201.48

Table 2. Processing times for solving MIT linear system equation using CPU and GPU.

Nmb.	MIT computational Model mesh Number of voxels	LUPIV method Intel Core i7, 2.8 GHz Elapsed time (s)	LUPIV method Tesla C2070 Elapsed time (s)	Gain
1	8488	35.26	6.54	5.39
2	20499	49.75	14.35	3.47
3	60896	179.25	38.64	4.64
4	87172	352.08	54.75	6.43
5	157264	1051.42	101.49	10.36

for reduced computational times for calculations involving a large number of matrix inversions has motivated the adaption of numerical methods for parallel computing architectures [30], including some for GPU architectures.

Applicable techniques for our method include the BiConjugate Gradients Stabilized Method (BICGS) [33], LU factorisation with partial pivoting (LUPIV) [34] and Gaussian elimination (GE) [35]. The BICGS is optimised for solving large-scale linear system equations in which the system matrices are sparse. This approach is therefore the most efficient for MIT problems whose system matrices are sparse. However, at this stage of research, it is not feasible to use the dense algebra library (Jacket DLA), which leads to the choice of either the LUPIV or GE method to solve Equation (15). We use the built-in Jacket CUDA-optimised LUPIV implementation to maximise performance when running on a GPU to minimize power usage. The comparison of MIT forward problem processing times for CPU and Tesla C2070 GPU are presented in Table 2.

As seen in Table 2 the use of the GPU-adapted LUPIV results in an increase in speed compared to an equivalent CPU implementation. To perform LUPIV computations using the GPU Jacket library, the system matrix needs to be converted from a sparse type to a full type. As the number of system matrix elements is defined by the square of the number of mesh vertices, the full representation of the system matrix can consume gigabytes of memory. For instance, a full representation of the system matrix using mesh model number 5 (Table 2) consumes approximately 4.6 GB of memory. Therefore the Jacket-based LUPIV memory consumption is the major drawback of using this algorithm for a large scale MIT mesh model and it can seriously limit its application. To avoid the problem of memory consumption, the sparse linear algebra and GPU-adapted BICGS algorithm (as stated in Section 2.2) will be applied for further improvement.

2.3.3. GPU Accelerated MIT Inverse Problem

The inverse problem in MIT can be solved using various electrical tomography image reconstruction techniques. As shown in Equation (22), in this study, we used the Tikhonov regularisation based method to reconstruct images. The computational time and memory consumption depends on both the size of the mesh in the forward problem and the complexity of the image reconstruction algorithm in the inverse problem. The inverse problem of MIT can be time consuming when the Jacobian matrix is large scale. The optimisation functions (Equation (22)) are composed of linear algebraic functions, performing addition, subtraction, multiplication and division for matrices. All these algebraic routines are parallelisable, especially multiplication [14, 36]. To improve the efficiency of solving the MIT inverse problem, we use the GPU Jacket library.

To apply GPU to the Tikhonov regularisation, we use a direct method, which is relatively easy to convert to GPU-based structure because the use of the linear matrix equation (Equation (22)). The CPU implementation is able to process this equation in its compact form; however, the GPU conversion requires it be decomposed into fewer simpler and shorter lines. This concept of equation adaptation is crucial for the management of Jacket GPU memory and data processing efficiency. A typical GPU data processing pipeline is optimised using many small linear algebra routines, and as such the efficiency of complex matrix operations is significantly improved. This approach is presented and discussed in Table 3.

Table 3. Fully sequential CPU-based MIT inverse problem vs. hybrid parallel GPU-based MIT inverse problem.

Sequential CPU implementation	Parallel GPU implementation
$x = (J^T J + \alpha R)^{-1} J^T b$	$A_{\text{GPU}} = J^T(*)$ $A_{\text{GPU}} = A_{\text{GPU}} J$ $A_{\text{GPU}} = A_{\text{GPU}} + \alpha R$ $A_{\text{GPU}} = inv(A_{\text{GPU}})(**)$ $B_{\text{GPU}} = J^T$ $B_{\text{GPU}} = B_{\text{GPU}} b$ $x = A_{\text{GPU}} B_{\text{GPU}}(***)$
The above equation is solved directly and sequentially from left to right. All linear algebra routines: multiplication,transposition addition and inversion are processed sequentially by CPU	The input is decomposed into new A and B temporal variables. Expressions are solved sequentially from top to bottom. All simplified linear algebra routines are processed in parallel by GPU

Although the sequential CPU approach is simpler and more intuitive, the GPU-based reconstruction is much faster. For example, the GPU-based transposition (*) occurs three orders of magnitude faster than its CPU equivalent [37]. The speed of matrix inversion (**) has been already proved in Table 2. The most efficient line is the last one (***). It includes the matrix multiplication process, which is considered the most efficient parallelised linear algebra transformation [18]. The acceleration of matrix multiplication is huge, up to several hundredfold. The same principles can be applied to other direct reconstruction methods, for example, NOSER (Newton-one-step-error-reconstruction) [38]. There are still no methods for the parallelisation of iterative reconstruction processes, such as the Landweber technique or the Gauss Newton iterations. This is due to the fact that the iterations are not independent of each other. Nevertheless, for an iterative process inside another iterative loop, it is still feasible to accelerate the computation using GPU implementation.

3. RESULTS

The purpose of this section is to demonstrate the feasibility of implementing MIT in a biomedical application, whereby both the solving processes of forward and inverse problems are accelerated using GPU high performance computing technique.

3.1. Simulations

In the simulations, we are interested in measuring the voltage perturbations caused by the changing properties of the non-conductive inclusion in a conductive background, including the locations and the

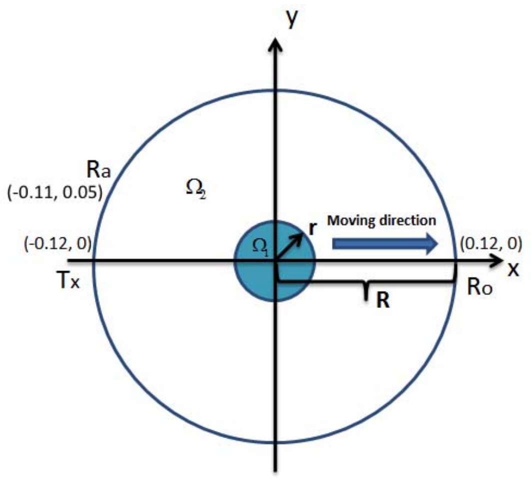

Figure 4. Cross-section of inclusion region Ω_1 in the sensing zone Ω_2: T_x shows a transmitting coil, R_a and R_o represent the locations of adjacent and opposite measuring coil in relation to T_x.

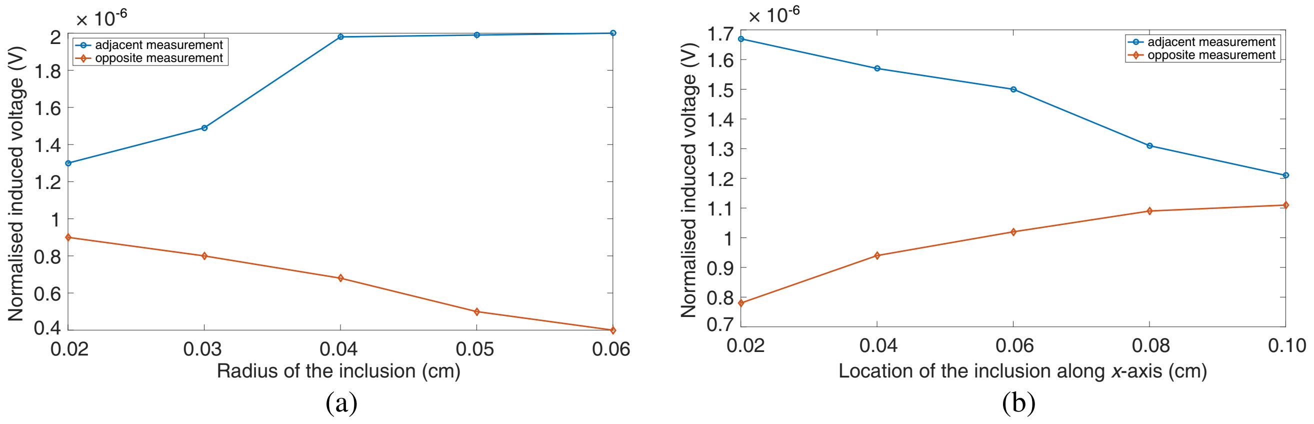

Figure 5. (a) Normalised induced voltages vs. radius of the inclusion; (b) normalised induced voltages vs. location of the inclusion along x-axis as shown in Figure 4.

volume of the inclusion. In the simulation model, an inclusion of 2 cm radius is located at the centre of the imaging region at the initial state. The conductivity of the background is 1.58 S/m. In all cases, an adjacent and an opposite measurement are simulated in order to show the changes in induced voltages. All the measurements are normalised against the background measurement. The moving direction of the inclusion and the locations of adjacent and opposite coils are shown in Figure 4.

The simulations presented in this section demonstrate the signal perturbations caused by changing the characteristics of the inclusion. Figure 5(a) shows the changes in induced voltages versus the volume of the non-conductive region. Note that the volume of the inclusion is approximated by the radius of the inclusion as the axial direction is uniform. During the process of volume expansion, the inclusion remains in the centre of the sensing zone. Figure 5(b) shows the induced voltage change versus the location of inclusion along positive x-axis, where the inclusion is moving off the centre. In this case, the induced voltages according to the adjacent measurement decrease as the inclusion moves away from the adjacent coil pair, while the induced voltages according to the opposite measurement increase as the distance between the inclusion and the opposite coil pair is reduced.

3.2. Experimental Results

In order to investigate the feasibility of using the MIT technique for reconstructing conductivity contrasts in biomedical applications, two sets of experiments are conducted (shown in Figure 6). Increasing the radius of the inclusion in the sensing zone (Figure 6(a)) corresponds to the voltage perturbation in Figure 5(a). The change in displacement of the inclusion from the centre of the sensing region, (Figure 6(b)), corresponds to the voltage perturbation in Figure 5(b). For each case, four different sizes of non-conductive bottle are tested, with diameters of 2, 6.5, 9.5 and 13 cm respectively. As this paper is focused on static 2D reconstructions, the bottles are uniform in the axial direction; therefore, the volume of the inclusion can be approximated by the cross-sectional area of the bottle.

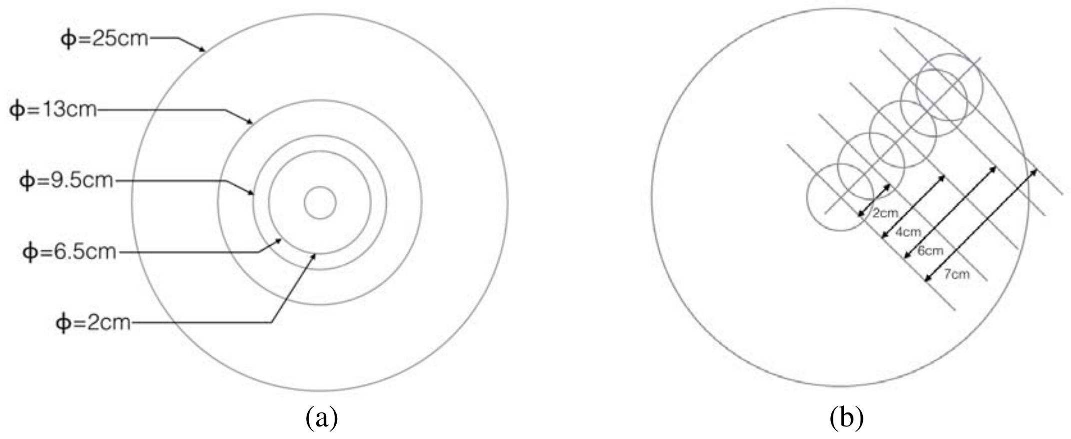

Figure 6. (a) Increasing the radius of the inclusion in the sensing zone; (b) the change in displacement of the inclusion from the centre of the sensing region.

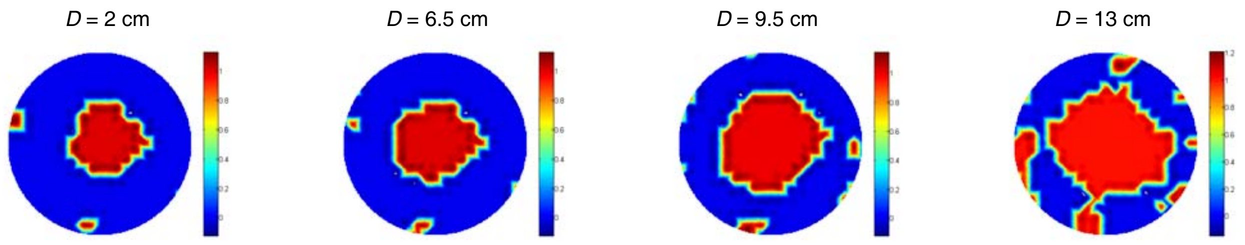

Figure 7. Reconstructed images of a non-conductive inclusion located in the centre of the measuring region filled with 0.9% saline solution (conductivity of 1.58 S/m). The diameters of the inclusions are 2, 6.5, 9.5 and 13 cm respectively, in columns left to right.

Figure 7 shows the first set of experiments, where the four different sizes of non-conductive bottle are placed in the centre of the imaging region. It can be seen that for all cases, the system can identify the non-conductive inclusion in the centre of the imaging region. Due to the low resolution of the imaging system and modest image reconstruction technique, the differences in the dimensions of the bottles cannot be easily distinguished from the reconstructed images; therefore, a post-processing tool is applied to make the images more intuitive. A blue to red colourmap is defined to show the conductivity contrast, with blue colour showing the conductive imaging region filled with 0.9% saline solution (conductivity of 1.58 S/m), and red colour showing a non-conductive inclusion. Any colour between blue and red represents an intermediate conductivity value. The reconstructed image data is normalised against its maximum value in order to demonstrate the distribution of the non-conductive inclusion in the same colour scale. This image reconstruction approach is adopted for the remainder of the experimental results.

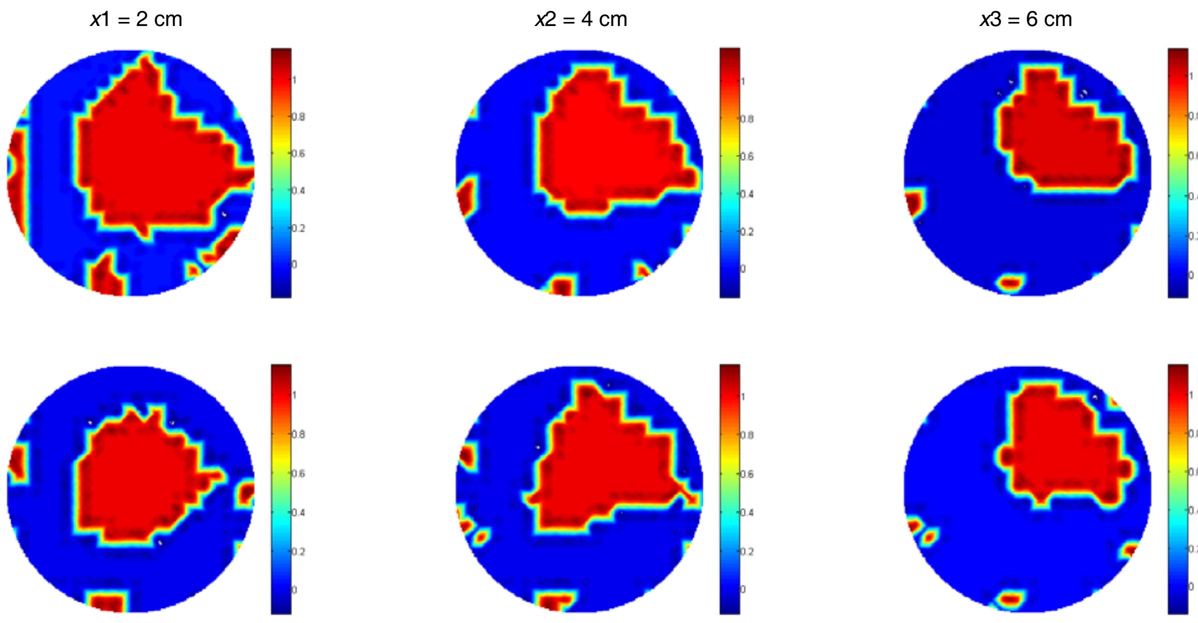

Figure 8. Reconstructed images of three steps of moving direction of a non-conductive inclusion in the measuring region filled with 0.9% saline solution (conductivity of 1.58 S/m). Reconstructions of inclusion of diameters 13 and 9 cm are shown from the top to the bottom row.

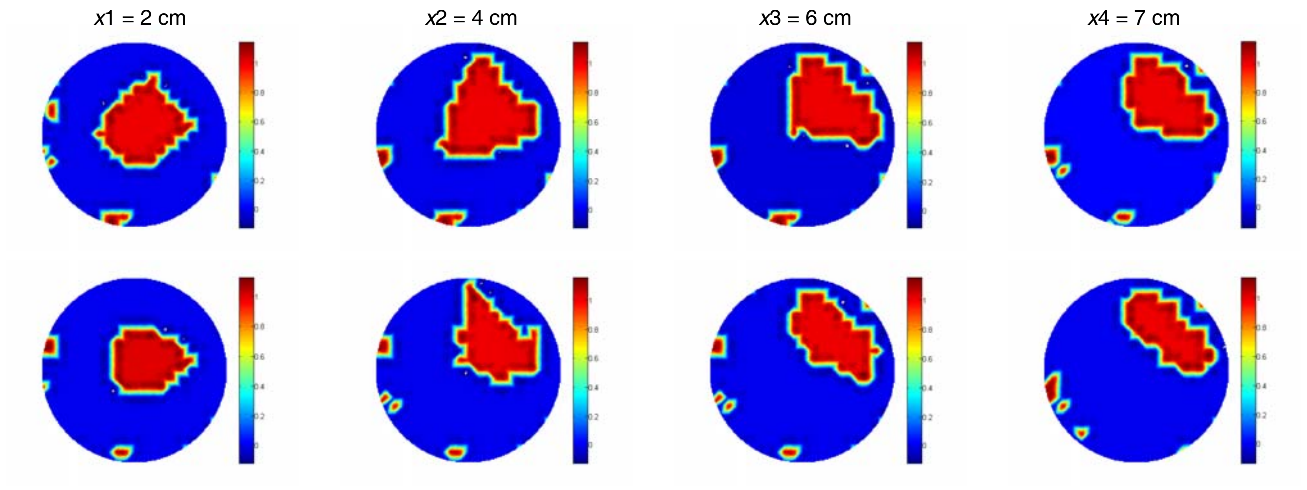

Figure 9. Reconstructed images of three steps of moving direction of a non-conductive inclusion in the measuring region filled with 0.9% saline solution (conductivity of 1.58 S/m). Reconstructions of inclusion of diameters 6.5 and 2 cm are shown from the top to the bottom row.

Figure 8 shows the reconstructed images of non-conductive inclusions moving increasingly off centre in increments of 2 cm. Two different sizes of inclusion are tested — of diameters 13 cm and 9.5 cm — reconstructions of which are shown from the top to the bottom row, of Figure 8 respectively.

Figure 9 shows the reconstructed images of four non-conductive inclusions off centre by 2, 4, 6 and 7 cm, shown in columns left to right, respectively. In addition, two different sizes of inclusion are tested — of diameters 6.5 cm and 2 cm — reconstructions of which are shown from the top to the bottom row, of Figure 9 respectively.

It can be seen from the experimental results presented in Figures 8 and 9 that as the inclusion moves increasingly off centre, there is a tendency for the reconstructed images to become elliptical. This is due to the low resolution of MIT. Although both simulations and experimental results demonstrate that the reconstruction of high conductive contrast is feasible, further development is essential to fully employ MIT as a useful imaging technique in biomedical applications, where 3D dynamic reconstruction and real-time monitoring are required. In this work, a linear reconstruction algorithm is used to solve the inverse problem. As the experimental data are in 2D, the scale of this problem is relatively small. In addition, the Jacobian matrix is calculated once and used in the inversion as a result of implementing the one step reconstruction algorithm. The reconstructed images shown in Figure 8 and Figure 9 are the proof of principle for potential medical imaging results. In practical cases, if higher resolution of the reconstructed image is required, the Jacobian matrix needs to be updated in an iterative reconstruction process. This way, the advantage of using GPU would be more beneficial in reducing the time consumption. However adapting GPU implementation in an iterative reconstruction algorithm for medical imaging is beyond the scope of this work.

4. CONCLUSION

In this study, a high frequency MIT system is used to study the reconstruction of conductive contrasts, with Tikhonov regularisation used to reconstruct 2D static images. Due to the nature of eddy currents, the MIT forward problem can be a large scale problem, and as such the forward solver is computationally expensive using traditional CPU computation techniques. This problem becomes more severe if the forward model is very dense, i.e., millions of elements, which is common in biomedical applications for brain/lung function imaging. This is one of the key limitations in MIT research. For a 2D problem, the inverse solver can usually be handled by a CPU, however, if the problem is 3D or involves a large number of voxels, this could be inefficient for the solving process.

In this study, both forward and inverse solvers are developed in-house. There are two major challenges in the program pipelines: the calculation of the Biot-Savart law and the calculation of the system linear equation, both of which can be accelerated using GPU implementation. General programming guidance of GPU implementation for Tikhonov regularisation is also presented, which offers computational gain by using matrix decomposition. The overall performance of the GPU implementation can be evaluated from a computational point of view. In order to reconstruct one frame of image, a Biot-Savart law for magnetic field calculation, a linear system equation and an inverse algorithm need to be solved. These take approximately 76%, 23% and $< 1\%$ of the total computational time, respectively. These estimates are drawn from the last model of Tables 1 and 2, where the sizes of both models are closer to those that would be used in a biomedical application. Considering the gain achieved by using GPU techniques, it is reasonable to believe that high performance computing techniques could reduce the overall processing time by 97%.

Due to the use of the accelerator package Jacket, some information relating to high performance computing is not accessible; however, the parallel implementation of MIT data processing using high level functions, such as transpose, multiplication, subtraction, addition and numerical API for inversion are included.

The aim of this paper is to demonstrate a means by which high performance computing techniques can be used for performance improvement of magnetic induction tomography, especially in the field of medical imaging. Although the model used in this study is a relatively simple 2D model, both the simulations and experimental results show that this GPU-based implementation can achieve uncompromising performance. One could also argue that the performance of GPU-based implementation in MIT would be further enhanced in more sophisticated 3D biomedical models. Future study will expand the scope of this paper to include the implementation of GPU for: i) inverse solution to Krylov subspace based methods, such as conjugate gradient and its preconditioning; ii) various regularisation methods such as total variation for a large scale 3D MIT; iii) detailed profiling, parallelisation scheme, and architecture for such implementations.

REFERENCES

1. Griffiths, H., "Magnetic induction tomography," *Measurement Science and Technology*, Vol. 12, 1126–1131, 2001.

2. Griffiths, H., W. R. Stewart, and W. Gough, "Magnetic induction tomography: A measuring system for biological tissues," *Annals of the New York Academy of Sciences*, Vol. 873, 335–345, 1999.
3. Otten, D. M. and B. Rubinsky, "Cryosurgical monitoring using bioimpedance measurements — A feasibility study for electrical impedance tomography," *IEEE Transactions on Biomedical Engineering*, Vol. 47, No. 10, 1376–1381, 2000.
4. Soleimani, M., O. Dorn, and W. R. B. Lionheart, "A narrow-band level set method applied to eit in brain for cryosurgery monitoring," *IEEE Transactions on Biomedical Engineering*, Vol. 53, No. 11, 2257–2264, 2006.
5. Zlochiver, S., M. M. Radai, M. Rosenfeld, and S. Abboud, "Induced current impedance technique for monitoring brain cryosurgery in a two-dimensional model of the head," *Annals of Biomedical Engineering*, Vol. 30, 1172–1180, 2002.
6. Zlochiver, S., M. Rosenfeld, and S. Abboud, "Contactless bio-impedance monitoring technique for brain cryosurgery in a 3D head model," *Annals of Biomedical Engineering*, Vol. 33, No. 5, 616–625, 2005.
7. Ma, L., H.-Y. Wei, and M. Soleimani, "Cryosurgical monitoring using electromagnetic measurements: A feasibility study for magnetic induction tomography," *13th International Conference in Electrical Impedance Tomography*, 2012-05-23–2012-05-25, Tianjin University, Tianjin, 2012.
8. Gabriel, C., A. Peyman, and E. H. Grant, "Electrical conductivity of tissues at frequencies below 1 MHz," *Physics in Medicine and Biology*, Vol. 54, No. 16, 4863–4878, 2009.
9. Gencer, N. G. and M. N. Tek, "Imaging tissue conductivity via contactless measurements: A feasibility study," *Elektrik*, Vol. 6, No. 3, 183–200, 1998.
10. Bagshaw, A. P., A. D. Liston, R. H. Bayford, A. Tizzard, A. P. Gibson, A. T. Tidswell, M. K. Sparkes, H. Dehghani, C. D. Binnie, and D. S. Holder, "Electrical impedance tomography of human brain function using reconstruction algorithms based on the finite element method," *Neuro Image*, Vol. 20, No. 2, 752–764, 2003.
11. Hwu, W.-M. W., *GPU Computing Gems*, Emerald Edition, Morgan Kaufmann, 2011.
12. Steuwer, M. and S. Gorlatch, "High-level programming for medical imaging on multi-GPU systems using the skelcl library," *2013 International Conference on Computational Science*, Vol. 18, 749–758, 2013.
13. Shi, L., W. Liu, H. Zhang, Y. Xie, and D. Wang, "A survey of gpu-based medical image computing techniques," *Quantitative Imaging in Medicine and Surgery*, Vol. 2, No. 3, 188–206, 2012.
14. Borsic, A., E. A. Attardo, and R. J. Halter, "Multi-GPU jacobian accelerated computing for soft field tomography," *Physiological Measurements*, Vol. 33, 1703–1715, 2012.
15. Eklund, A., P. Dufort, D. Forsberg, and S. M. LaConte, "Medical image processing on the GPU: Past, present and future," *Medical Image Analysis*, Vol. 17, No. 8, 1073–1094, 2013.
16. Jia, X., B. Dong, Y. F. Lou, and S. B. Jiang, "GPU-based iterative cone-beam CT reconstruction using tight frame regularization," *Physics in Medicine and Biology*, Vol. 56, No. 13, 3787, 2011.
17. Tavares, R. S., T. C. Martins, and M. S. G. Tsuzuki, "Electrical impedance tomography reconstruction through simulated annealing using a new outside-in heuristic and GPU parallelization," *Journal of Physics Conference Series*, Vol. 407(Conference 1), 012015, 2012.
18. Kapusta, P., M. Majchrowicz, D. Sankowski, and R. Banasiak, "Application of GPU parallel computing for acceleration of finite element method based 3D reconstruction algorithms in electrical capacitance tomography," *Image Processing and Communications*, Vol. 17, No. 4, 339–346, 2013.
19. Maimaitijiang, Y., M. A. Roula, S. Watson, R. Patz, R. J. Williams, and H. Griffiths, "Parallelization methods for implementation of a magnetic induction tomography forward model in symmetric multiprocessor systems," *Parallel Computing*, Vol. 34, No. 9, 497–507, 2008.
20. Maimaitijiang, Y., M. A. Roula, S. Watson, G. Meriadec, K. Sobaihi, and R. J. Williams, "Evaluation of parallel accelerators for high performance image reconstruction for magnetic induction tomography," *Journal of Selected Areas in Software Engineering (JSSE)*, January 2011.

21. Wei, H.-Y. and M. Soleimani, "Hardware and software design for a national instrument-based magnetic induction tomography system for prospective biomedical applications," *Physiological Measurement*, Vol. 33, No. 5, 863–879, 2012.
22. Bíró, O., "Edge element formulations of eddy current problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 169, 391–405, 1999.
23. Bíró, O. and K. Preis, "An edge finite element eddy current formulation using a reduced magnetic and a current vector potential," *IEEE Transactions on Magnetics*, Vol. 36, No. 5, 3128–3130, 2000.
24. Golias, N. A., C. S. Antonopoulos, T. D. Tsiboukis, and E. E. Kriezis, "3D eddy current computation with edge elements in terms of the electric intensity," *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, Vol. 17, No. 5/6, 667–673, 1998.
25. Barber, D. C. and B. H. Brown, "Applied potential tomography," *Journal of Physics E: Scientific Instruments*, Vol. 17, No. 9, 723–734, 1984.
26. Soleimani, M. and W. R. B. Lionheart, "Absolute conductivity reconstruction in magnetic induction tomography using a nonlinear method," *IEEE Transactions on Medical Imaging*, Vol. 25, No. 12, 1521–1530, 2006.
27. Dyck, D. N., D. A. Lowther, and E. M. Freeman, "A method of computing the sensitivity of the electromagnetic quantities to changes in the material and sources," *IEEE Transactions on Magnetics*, Vol. 30, 3415–3418, 1994.
28. Soleimani, M. and W. R. B. Lionheart, "Image reconstruction in three-dimensional magnetostatic permeability tomography," *IEEE Transactions on Magnetics*, Vol. 41, 1274–1279, 2005.
29. Calvetti, D., S. Morigi, L. Reichel, and F. Sgallari, "Tikhonov regularization and the L-curve for large discrete ill-posed problems," *Journal of Computational and Applied Mathematics*, Vol. 123, No. 1, 423–446, 2000.
30. Kirk, D. B. and W-M. W. Hwu, *Programming Massively Parallel Processors: A Hands on Approach*, Elsevier, 2010.
31. Bernstein, A. J., "Program analysis for parallel processing," *IEEE Transactions on Electronic Computers*, Vol. 15, 757–762, October 1966.
32. Arora, M., "The architecture and evolution of CPU-GPU systems for general purpose computing," University of California, 2012.
33. Soleimani, M., C. E. Powell, and N. Polydorides, "Improving the forward solver for the complete electrode model in eit using algebraic multigrid," *IEEE Transactions on Medical Imaging*, Vol. 24, No. 5, 577–583, 2005.
34. Lezar, E. and D. Davidson, "GPU-based LU decomposition for large method of moments problems," *Electronics Letters*, Vol. 46, No. 17, 1194–1196, 2010.
35. Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 3rd Edition, 1992.
36. Humphrey, J. R., D. K. Price, K. E. Spagnoli, A. L. Paolini, and E. J. Kelmelis, "Cula: Hybrid GPU accelerated linear algebra routines," *SPIE Defense and Security Symposium (DSS)*, April 2010.
37. Wei, H.-Y. and M. Soleimani, "Three-dimensional magnetic induction tomography imaging using a matrix free Krylov subspace inversion algorithm," *Progress In Electromagnetics Research*, Vol. 122, 29–45, 2012.
38. Cheney, M., D. Isaacson, J. C. Newell, S. Simske, and J. Goble, "Noser: An algorithm for solving the inverse conductivity problem," *International Journal of Imaging Systems and Technology*, Vol. 2, No. 2, 1990.