INTERACTIVE NMR MAGNETIC FIELD VISUALIZATION
ON GRAPHICS PROCESSORS

by

Biswajit Mishra

A thesis submitted to
Indian Institute of Science
in partial fulfillment of the requirements for the degree of

Master of Technology (Computational Sciences)

Supercomputer Education and Research Centre

Indian Institute of Science

Bangalore
India
June 2006
ABSTRACT

INTERACTIVE NMR MAGNETIC FIELD VISUALIZATION ON GRAPHICS PROCESSORS

Biswajit Mishra

Supercomputer Education and Research Centre

Master of Technology (Computational Sciences)

NMR magnets need to have better than $10^{-8}$ homogeneity over a 10x10x20 mm$^3$ space which is obtained by optimally tuning a set of Shim coils. These shim coil currents are adjusted manually by an operator to achieve a homogeneous magnetic field. Visualizing magnetic field will assist operator in optimal shimming.

A novel method to visualize NMR Magnetic field is presented. The problem poses challenges that are different from computationally intensive general magnetic field computation and visualization. A GPU (Graphics processing Unit) based technique is developed for efficient interactive visualization. This approach exploits the inherent parallelism in the GPUs thus relinquishing CPU from most of the heavy computation. The field equations are transformed into a set of multi-pass GPU shader programs that encapsulates Shim coil current inputs and coil geometry. A geometric deformation based visualization tech-
nique is developed to identify dominant inhomogeneity present in the field to be corrected by adjusting a particular shim current. This implementation delivers required frame rates for real time user interactions.
ACKNOWLEDGMENTS

I would like to express my immense gratitude to my advisor, Dr. P.C. Mathias for his guidance and support. My association with him over past one year has been a greatly enriching experience. I wish to express my profound thanks to my labmate Manohar B.S. for his practical support. My special thanks to IISc authorities and, SERC in particular for providing an un-interrupted computing facility. Finally, I thank all my friends in the Department and the Institute for making my stay here a memorable one.
# Contents

Table of Contents ........................................... v

List of Figures .............................................. vi

1 Introduction .............................................. 1
   1.1 NMR Spectrometer ..................................... 1
       1.1.1 What is Shimming? ................................. 2
   1.2 Graphics Processing Unit ............................... 4
       1.2.1 Why on GPU? ....................................... 4

2 Theory of Magnetic Field ................................ 6
   2.1 Spherical Harmonic Solution of Magnetic Field ......... 6
   2.2 Computation of Harmonic Co-efficient .................. 7
       2.2.1 Taylor Series Expansion and Recurrence Relations . 7
       2.2.2 Circular Arc Field ................................ 9
       2.2.3 Solenoid Field .................................... 10
       2.2.4 Shim Coils Equations .............................. 13
   2.3 The Shim Coil Functions ................................ 13

3 GPU Architecture and Texture Based Volume Rendering ... 17
   3.1 GPU Architecture ..................................... 17
       3.1.1 3D Texture based volume rendering ................. 20
   3.2 Magnetic field computation on GPU ..................... 21

4 Deformation Based Visualization Technique ................ 25
   4.1 Deformation Based Visualization Technique ............ 25
       4.1.1 Implementation of Deformation Based Visualization . 26

5 Conclusion and Future work ................................ 38
   5.1 Conclusion and Future Work ............................ 38

Bibliography .................................................. 40

Index .......................................................... 41
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Coordinate system for field due to solenoid</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Field due to circular arc</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>GeForce 6 Series Architecture</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Slicing Plane in 3D texture based volume rendering</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>Flow diagram of one render cycle</td>
<td>22</td>
</tr>
<tr>
<td>3.4</td>
<td>Volume rendered with slicing plane 3D Textures</td>
<td>23</td>
</tr>
<tr>
<td>3.5</td>
<td>Magnetic field image with no inhomogeneity</td>
<td>23</td>
</tr>
<tr>
<td>4.1</td>
<td>3D meshing volume for deformation based visualization</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>CPU GPU PERFORMANCE STUDY</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>speed up study</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>rendered image with $Z(X^2 - Y^2)$ inhomogeneity</td>
<td>30</td>
</tr>
<tr>
<td>4.5</td>
<td>rendered image with $X^3$ inhomogeneity</td>
<td>30</td>
</tr>
<tr>
<td>4.6</td>
<td>rendered image with $Y^3$ inhomogeneity</td>
<td>31</td>
</tr>
<tr>
<td>4.7</td>
<td>rendered image with $Y$ inhomogeneity</td>
<td>31</td>
</tr>
<tr>
<td>4.8</td>
<td>rendered image with $Z^2X$ inhomogeneity</td>
<td>32</td>
</tr>
<tr>
<td>4.9</td>
<td>rendered image with $Y^2Z$ inhomogeneity</td>
<td>32</td>
</tr>
<tr>
<td>4.10</td>
<td>rendered image with $Z^2 - X^2$ inhomogeneity</td>
<td>33</td>
</tr>
<tr>
<td>4.11</td>
<td>rendered image with $XYZ$ inhomogeneity</td>
<td>33</td>
</tr>
<tr>
<td>4.12</td>
<td>rendered image with no inhomogeneity</td>
<td>34</td>
</tr>
<tr>
<td>4.13</td>
<td>rendered image with $XY$ inhomogeneity</td>
<td>34</td>
</tr>
<tr>
<td>4.14</td>
<td>rendered image with $Z$ inhomogeneity</td>
<td>35</td>
</tr>
<tr>
<td>4.15</td>
<td>rendered image with no inhomogeneity</td>
<td>35</td>
</tr>
<tr>
<td>4.16</td>
<td>rendered image with $ZX$ inhomogeneity</td>
<td>36</td>
</tr>
<tr>
<td>4.17</td>
<td>rendered image with $Z^3$ inhomogeneity</td>
<td>36</td>
</tr>
<tr>
<td>4.18</td>
<td>rendered image with $Z^4$ inhomogeneity</td>
<td>37</td>
</tr>
<tr>
<td>4.19</td>
<td>rendered image with $Z^5$ inhomogeneity</td>
<td>37</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 NMR Spectrometer

Nuclear Magnetic Resonance (NMR) spectroscopy is one of the principal techniques used to obtain physical, chemical, electronic and structural information about a molecule. It is the only technique that can provide detailed information on the exact three-dimensional structure of biological molecules in solution. NMR equipment requires a source of magnetic field with minimal inhomogeneities. No magnet generates an ideal homogeneous magnetic field and therefore a complicated system of coils is required. Coils fed by separate current supplies to remove particular magnetic inhomogeneity \((X, Y, Z, XY, YZ, XZ, X^2 - Y^2, XZ^2, YZ^2, Z, Z^2, Z^3, Z^4)\) is needed. For a solenoid magnet with a cylinder shaped work space, the individual active shims consists of circular and saddle shaped coils symmetrically positioned on the co-axial cylindrical surfaces.

The theory of magnetic field due to a known coil geometry is well established. Significant research has gone into computational aspect of the magnetic field. The theory, design and construction of a magnetically shielded solenoid is described in [1],
[2], [3]. High resolution magnet for NMR and MRI based on the spherical harmonics and unique recursion relations for the co-efficient of most dominant components is described in [4].

1.1.1 What is Shimming?

In the beginning, the field homogeneity of large electromagnets was adjusted by mechanical alignment of the magnet pole faces. The more parallel the pole faces, the more homogeneous the magnetic field. The first step in the process of adjusting magnetic homogeneity was to adjust the position of the magnet’s pole faces by turning three large bolts which held the pole faces. Adjusting these bolts tilted the pole faces relative to each other with the aim of making the pole faces more parallel. If the bolts ran out of range, thin pieces of brass were placed between the magnet yoke and the pole pieces to move the pole pieces as parallel as possible. These thin pieces of brass were also placed in other strategic locations to make the pole faces parallel in a manner not addressed by the three adjustment bolts. The metal pieces were called shim stock and the seemingly endless process of placing and removing pieces of shim stock acquired the name ”shimming”. Because tons of magnetic field pressure existed on the pole faces, the magnet had to be turned off to place and remove the shim stock. When the sample was spinning, the final part of adjusting magnetic homogeneity with these systems was to adjust a ratchet bolt which pulled together or pushed apart the tops of the magnet pole pieces to give a fine adjustment of the Y gradient. All of these processes were mechanical in nature. After these adjustments, the NMR instruments were typically capable of giving better than 0.2 Hz resolution. This is rather impressive when you consider that 0.2 Hz out of 60 MHz represents 3 parts per billion field homogeneity over the volume of the sample.

To increase the performance, reduce the difficulty of adjusting magnetic homo-
1.1 NMR Spectrometer
geneity and reduce the manufacturing difficulty of the magnets, an electronic "shim-
ing" process was developed which used a series of small electromagnets having very
specific magnet field contours. These small electromagnets are placed around the
sample area. Each small electromagnet can be used to adjust the field in the volume
of observation to create more or counteract existing types of magnetic gradients.
A complete series of these electromagnets can be used to adjust the magnetic field
homogeneity to a given level of purity depending on how many types of adjustment
electromagnets are used. The process of adjusting the magnetic field homogeneity by
adjusting the current in each of the small electromagnets retained the name shimming
and the small electromagnets assumed the name "shims".

At first only a few low order (X, Y, and Z) electrical shims were used. As the
fields became higher, magnet production became more difficult, and more and higher
order electrical shims were added to maintain the same level of performance. These
electrical shims are not 100% pure and have interactions with shims of a similar
nature (ZX creates some Z gradient and X gradient in addition to the intended ZX
gradient). Because of these interactions, the number of adjustments necessary to
shim the magnet increases geometrically with the number of shims, not just linearly.
In addition, the raw field encountered in superconducting magnets is usually worse
than in electromagnets, so larger corrections are required. These two facts make
the process of shimming superconducting magnets more difficult and the shimming
process more important to obtain useful NMR spectra.

To obtain 0.2 Hz resolution requires ten times greater magnetic field homogeneity
at 600 MHz than at 60 MHz. Therefore, in addition to the higher field supercon-
ducting magnets being more difficult to shim, shimming becomes more important to
obtain the same results as the magnetic fields increase. Other aspects of an NMR
instrument’s performance are also affected by shimming, such as the NMR signal’s
lineshape, which is critical for achieving good solvent suppression. So the necessary evil of adjusting the small electromagnets, called shimming, remains very important in today’s NMR instrumentation.

\section{1.2 Graphics Processing Unit}

GPUs have evolved as a fully programmable and immensely parallel workhorse for graphics computation. Availability of high-level languages and tools have led to explosion of innovation and creativity. Programmability allows for running any task on GPU as long as an algorithm can be found that fits into the streaming model. GPGPU (General Purpose Computation on GPU) applications \cite{5} ranges from numeric computing operation such as dense and sparse matrix multiplication technique or multi-grid conjugate gradient solvers for system of linear equations to computer graphics processes such as ray tracing and photon mapping usually performed offline in CPU, to physical simulations such as fluid mechanics solvers ,to data base and data mining operations. The physics based simulation on GPU was first seen in \cite{5}, which also demonstrated the "Game of Life" cellular automata and a 2D physically based wave simulation. Several researcher has used GPU to simulate fluid dynamics. Related to fluid simulation is visualizations of flow which has been implemented using graphics hardware to accelerate line integral convolution an Lagrangian-Eulerian advection.

\subsection{1.2.1 Why on GPU?}

Shimming the NMR magnet is a complex task. The most common method of adjusting the homogeneity of an NMR magnet is the observation of an NMR signal. The problem with the shimming process is that the observed NMR signal results from
the integrated signal from the total volume of the observed sample, which may have many different resonant frequencies with different degrees of excitation arising from different positions in the sample. NMR sample can be visualized as a continuum of isolated mini-samples, each of which is infinitesimally small. Each mini-sample then generates a signal whose linewidth is determined by the T2 relaxation time of the sample and whose frequency results from the field value at that point. The intensity of the signal generated by each mini-sample would reflect the amount of excitation at that point. What the NMR operator observes is the sum of signals from all the mini-samples. In other words, the NMR signal is the total integrated signal over the total sample volume times each area’s degree of excitation. It is the integration of the NMR signal response which leads to a major difficulty in the shimming process. Any knowledge as to which part of the NMR sample is experiencing the magnetic inhomogeneity is lost in this integration process. Thus, to overcome above difficulty, instead of signals the magnetic field inside sample volume needed to be visualized. Though, scalar field visualization is solved problem but visualizing magnetic field in the context of shimming poses two challenges:

1: It should be interactive. For an interactive application at least 20-30 frame per second is needed.

2: The type of inhomogeneity present should be identified from the rendered image.

Because of inherent parallelism GPUs are highly efficient for data and compute intensive operations. To achieve required frame rate it is implemented on GPU.
Chapter 2

Theory of Magnetic Field

2.1 Spherical Harmonic Solution of Magnetic Field

The total magnetic induction in a homogeneous region containing no field sources is described by the Laplace equation. NMR applications employ magnets that are axially symmetric [2].

\[ \nabla^2 B_z = 0 \quad (2.1) \]

The dominant component \( B_z \), parallel to the symmetry axis is also described by the Laplace equation in the form Eq.(2.1) The solution of Eq.(2.1) can be expressed in spherical harmonic expansion

\[ B_z(x, y, z) = \sum_{n=0, m=0}^{\infty} r^n m P_n(\cos \theta) \times [a_{m,n} \cos (m\varphi) + b_{m,n} \sin (m\varphi)] \quad (2.2) \]

where \( r, \theta \) and \( \phi \) are spherical coordinates, \( a_{m,n} \) and \( b_{m,n} \) are coefficients and \( m P_n(\cos \theta) \) are associated Legendre polynomials of first kind, degree \( n \) and order \( m \). The first member of Eq.(2.2) is the ideal homogeneous field. Eq.(2.2) can be expanded in Cartesian coordinates as in Eq.(2.3), shown only up to third degree and order.
\[ B_z(x, y, z) = \]
\[ a_{0, 0} + a_{0, 1}z + a_{0, 2} \left( z^2 - \frac{1}{2} (x^2 + y^2) \right) + a_{0, 3}z \left( z^2 - \frac{3}{2} (x^2 + y^2) \right) + \ldots \]
\[ + a_{1, 1}x + 3a_{1, 2}xz + a_{1, 3}x \left( 6z^2 - \frac{3}{2} (x^2 + y^2) \right) + \ldots \]
\[ + b_{2, 1}y + 3b_{2, 2}yz + b_{1, 3}y \left( 6z^2 - \frac{3}{2} (x^2 + y^2) \right) + \ldots \]
\[ + 3a_{2, 2} (x^2 - y^2) + 15a_{2, 3}z (x^2 - y^2) + \ldots \]
\[ + 6b_{2, 2}xy + 30b_{2, 3}xyz + \ldots \]
\[ + 15a_{3, 3}x (x^2 - 3y^2) + \ldots \]
\[ + 15b_{3, 3}y (3x^2 - y^2) + \ldots \]

\[ (2.3) \]

2.2 Computation of Harmonic Co-efficient

2.2.1 Taylor Series Expansion and Recurrence Relations

The dependence of the magnetic field derivatives on the current carrying conductor can be described by Taylor series in cartesian form Eq. (2.4). The expansion is in the neighborhood of the origin.

\[ B_z(x, y, z) = \sum_{i,j,k} \tau_{i,j,k} x^i y^j z^k \]  
(2.4)

\[ \tau_{i,j,k} = \frac{1}{i!j!k!} \left. \frac{\partial^{i+j+k} B_z}{\partial x^i \partial y^j \partial z^k} \right|_{x=0, y=0, z=0} \]  
(2.5)

Applying Laplace’s differential equation Eq. (2.1) in the form Eq. (2.6) to Eq. (2.4)

\[ \frac{\partial^2 B_z}{\partial y^2} = - \frac{\partial^2 B_z}{\partial z^2} - \frac{\partial^2 B_z}{\partial x^2} \]  
(2.6)
we have

\[ B_z(x, y, z) = \]

\[ = \tau_{0,0,0} + \tau_{0,0,1}z + \tau_{0,0,2}(z^2 - y^2) + \tau_{0,0,3}(z^2 - 3y^2) + \ldots \]

\[ + \tau_{1,0,0}x + 2\tau_{1,0,1}xz \]

\[ + \tau_{0,1,0}y + 2\tau_{0,1,1}yz \]

\[ + \tau_{2,0,0}(x^2 - y^2) + 3\tau_{2,0,1}z(x^2 - y^2) + \ldots \]

\[ + 2\tau_{1,1,1}xy + 6\tau_{1,1,1}xyz + \ldots \]

\[ + \tau_{3,0,0}(x^2 - 3y^2) + \ldots \]

\[ + \tau_{2,1,0}(3x^2 - y^2) + \ldots \]

(2.7)

The polynomial in the \( M \)th row and \( n \)th column of the expansion Eq.(2.7) can be denoted by \( ^MT_n \), the following recursion formulae can be adopted.

\[ ^0T_n = z^0T_{n-1} - y^2T_{n-1} \]

\[ n = 1, 2, 3, \ldots \]

(2.8)

\[ ^1T_n = n^1T_{n-1} \]

\[ ^2T_n = y^0T_{n-1} + z^2T_{n-1} \]

\[ ^MT_n = 2n\left(x^{M-2}T_{n-1} - y^{M-1}T_{n-1}\right)/(M + 1) \]

(2.9)

for odd \( M = 3, 5, \ldots, 2n - 1 \) and

\[ ^MT_n = z^MT_{n-1} + x^{M-2}T_{n-1} + y^{M-3}T_{n-1} \]

(2.10)

for even \( M = 4, 6, \ldots, 2n \) where \( ^0T_0 = 1 \) and \( ^2T_{n-1} = 0 \).

Eq.(2.3) and Eq.(2.7) describes the same field. Their members with identical powers of the individual coordinates can be compared and the relationship between
the coefficients can be determined. The coefficients $a_{m,n}$ and $b_{m,n}$ are specified as the linear combination of the derivatives $\tau_{i,j,k}$, $i + j + k = n$.

### 2.2.2 Circular Arc Field

$B_z$ due to a current carrying circular arc of conductor as shown in Fig. 2.2 at a point $(x, y, z)$ is given by Biot-Savart law, Eq.(2.11).

$$B_z(x, y, z) = \frac{\mu I r}{4\pi} \times \frac{(r - x \cos \xi - y \sin \xi) d\xi}{\left[ x^2 + y^2 + (z - d)^2 - 2r (x \cos \xi + y \sin \xi) + r^2 \right]^{3/2}}$$

(2.11)

where $I$ is the current in the circular conductor of radius $r$ and arc angle $2\phi$, $d$ is the $z$ coordinate of the plane containing the circular arc. $\mu$ is the permeability of the medium and the $\xi$ is the integration variable. By evaluating the derivatives of Eq.(2.11) as specified by Eq.(2.5) at the origin, the $\tau_{i,j,k}$ are obtained and are consecutively transformed to spherical harmonic coefficients $a_{m,n}$ and $b_{m,n}$. For the given pose of circular arc $b_{m,n} = 0$. By further analysis $a_{m,n}$ can be found out by exploiting the recurrence relation and rotary symmetry. Coefficients for the zonal members (rotary symmetry, $m=0$) are given by formula

$$a_{0,n} = C_n^0 Z_n \phi$$

(2.12)

Where $C_n$ equals $\mu I / 2\pi cc^n$. Using $\beta = d/c$ for the relative position of the current carrying arc $^0Z_n$ is determined by the recursion formula

$$p Z_q = - \frac{(2q + 1)^p Z_{q-1} + [q^2 - (p - 1)^2]^p Z_{q-2}}{1 + \beta^2}$$

(2.13)

where $p = 0$ and $q = 1, 2, 3, \ldots n$ successively, the starting functions being
2.2 Computation of Harmonic Co-efficient

\[ 0Z_0 = (1 + \beta^2)^{-3/2} \text{ and } 0Z_0 = 0 \]

The Coefficients for the tesseral \((n > m > 0)\) and sectorial terms (independent of \(z\) coordinate, \(n = m\)) are calculated using expression

\[ a_{m,n} = C_n \prod_{i=1}^{m} \left| \frac{(2i - 3)}{n + i} \right| \times \left[ (4m^2 - 1)^m Z_n - m^{-2} Z_n \right] \frac{\sin m\phi}{m} \]  \( (2.14) \)

where \(mZ_n\) and \(m^{-2}Z_n\) are auxiliary functions for the degree \(n = 1, 2, 3, ...\) and the order \(m = 1, 2, 3,...\). They are given by formula \((2.13)\) with generalized starting functions

\[ pZ_{p=q} = (1 + \beta^2)^{-p-(3/2)} \quad \text{and} \quad pZ_{q<p} = 0. \]

An application of these relations to calculate the starting functions is demonstrated for \(m=1\); the determination of the function \(1Z_n\) begins with \(1Z_0 = 0\) and \(1Z_1 = (1 + \beta^2)^{-2.5}\), similarly, that of \(-1Z_{-2} = 0\) and \(1Z_1 = (1 + \beta^2)^{-0.5}\).

2.2.3 Solenoid Field

To describe the spatial variation of the field it is most convenient to use an expansion in spherical harmonics. The field is completely specified if the field on the axis of the symmetry is known [1].

The coordinate system used to describe the field are shown in Fig.2.1. A given point in the solenoid can be described by its spherical coordinates \((r, \theta, \phi)\) or its cylindrical coordinates \((\rho, \phi, z)\). To describe the magnetic field we shall use the components parallel to and perpendicular to the axis of symmetry as in Eq.(2.15).

\[ H = k \cdot H_x(r, \theta) + \gamma H_\rho(r, \theta) \]  \( (2.15) \)

where \(\gamma\) and \(k\) are unit vectors in the \(\rho\) and \(z\) directions, respectively. Since the
2.2 Computation of Harmonic Co-efficient

Figure 2.1 (a): Coordinate system used to describe the field in axially symmetric system.
(b): coordinate used to specify the single layer of the solenoid of length $L$ and radius $a_s$. 
2.2 Computation of Harmonic Co-efficient

system of conductors is axially symmetric, the magnetic field will not be a function of the azimuthal angle \( \phi \). We write the potential inside the system of conductors with cylindrical geometry as Eq. (2.16).

\[
V(r, \theta) = \sum_{n=1}^{\infty} \left( -\frac{H_{n-1}}{n} \right) r^n P_n(\cos \theta)
\]  

(2.16)

where \( P_n(\cos \theta) \) is a Legendre polynomial. The potential function gives the radial and axial components as in Eq. (2.17) and Eq. (2.18).

\[
H_z(r, \theta) = \sum_{n=1}^{\infty} H_n r^n P_n(\cos \theta)
\]

(2.17)

\[
H_\rho(r, \theta) = \sum_{n=1}^{\infty} \left( \frac{1}{n+1} \right) H_n r^n \frac{d}{d\theta} P_n(\cos \theta)
\]

(2.18)

Where the coefficient \( H_n \) are given by the relationship in Eq. (2.19) and Eq. (2.20).

\[
H_n = \left( \frac{1}{n!} \right) \left[ \left( \frac{d^n}{dz^n} \right) H_z(z, 0) \right]_{z=0}
\]

(2.19)

\[
H_z(z, 0) = \frac{4\pi N_s I}{20r_s} \times \left\{ 1 - \left[ \frac{1 - (u_s)^2}{u_s} \right] \times \sum_{n=1}^{\infty} \left( \frac{1}{2n} \right) \left( \frac{z}{r_s} \right)^{2n} P'_{2n}(u_s) \right\}
\]

(2.20)

Here \( N_s \) is the total number of turns, \( u_s \) is for \( \cos \theta_s \), \( I \) is the current, and \( P'_{2n}(u_s) = \frac{d}{du_s} P_{2n}(u_s) \)

The magnetic field is in gauss if \( I \) is in amperes and \( r_s \) is in centimeters. Due to the plane of symmetry through the origin odd powers of \( z \) in the expression Eq. (2.20) is zero. We can rewrite Eq. (2.20) in series form as Eq. (2.21).
2.3 The Shim Coil Functions

\[ H_z(z, 0) = \frac{4\pi NI}{10} (\cos \theta_s) \times \]
\[
\left\{ 1 - \frac{3}{2} \sin^4 \theta_s \left( \frac{z}{a_s} \right)^2 - \frac{5}{8} (\sin^6 \theta_s) (7 \cos^2 \theta_s - 3) \left( \frac{z}{a_s} \right)^4 \\
- \frac{7}{16} (\sin^8 \theta_s) \times (33 \cos^4 \theta_s - 30 \cos^2 \theta_s + 5) \left( \frac{z}{a_s} \right)^6 - \ldots \right\} \]

(2.21)

Here \( N \) is the number of turns per unit length. The successive terms of this expression will be referred to as the zero-order term, second-order term, the fourth-order term, etc.

2.2.4 Shim Coils Equations

Shim coils are designed to produce gradient fields to compensate for the inhomogeneities in the field due to the main solenoid. The currents in the coil are independently controlled to give maximum freedom for tuning. In order to make the field as homogeneous as possible, the derivatives of the axial field \( B_z \) at the origin should be minimized or made zero by using the shim coils. This is one of the design consideration for the shim coils. Detailed geometric design of X-shim coils is given in Table 2.1. For the purpose of further computation and visualization we consider typical geometric design for main solenoid and shim coils and derive the coefficients. The spherical harmonic coefficients \( a_{m,n} \) and \( b_{m,n} \) of typical shim coils are listed in Table 2.2.

2.3 The Shim Coil Functions

In previous sections the theory to compute coefficients of spherical harmonics for different geometry is described. In table 2.2 co-efficients for X shim is described. Ideally
### Table 2.1 The parameters for X shim coil design

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal radius $r$</td>
<td>25 mm</td>
</tr>
<tr>
<td>Conductor cross section</td>
<td>0.7 mm x 0.035 mm</td>
</tr>
<tr>
<td>Nominal current</td>
<td>100.0 mA</td>
</tr>
<tr>
<td>Length of winding</td>
<td>12.4 m</td>
</tr>
<tr>
<td>Resistance at 20°C</td>
<td>8.8 Ω</td>
</tr>
</tbody>
</table>

### Table 2.2 Values of the spherical harmonic coefficients for the coils derived from the coil geometry. The coefficients are in µtesla.

#### Main solenoid

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{0,0}$</td>
<td>7.5 x 10^6</td>
</tr>
<tr>
<td>$a_{0,1}$</td>
<td>4.2 x 10^2</td>
</tr>
<tr>
<td>$a_{0,2}$</td>
<td>4.1 x 10^2</td>
</tr>
<tr>
<td>$a_{0,3}$</td>
<td>3.1 x 10^-1</td>
</tr>
<tr>
<td>$a_{1,1}$</td>
<td>5.2 x 10^2</td>
</tr>
<tr>
<td>$a_{1,2}$</td>
<td>3.2 x 10^2</td>
</tr>
<tr>
<td>$a_{1,3}$</td>
<td>5.1 x 10^-1</td>
</tr>
<tr>
<td>$b_{1,1}$</td>
<td>5.2 x 10^2</td>
</tr>
<tr>
<td>$b_{1,2}$</td>
<td>1.2 x 10^2</td>
</tr>
<tr>
<td>$b_{1,3}$</td>
<td>5.1 x 10^-1</td>
</tr>
<tr>
<td>$a_{2,2}$</td>
<td>4.4 x 10^1</td>
</tr>
<tr>
<td>$a_{2,3}$</td>
<td>2.1 x 10^1</td>
</tr>
<tr>
<td>$b_{2,2}$</td>
<td>1.2 x 10^1</td>
</tr>
<tr>
<td>$b_{2,3}$</td>
<td>5.2 x 10^1</td>
</tr>
<tr>
<td>$a_{3,3}$</td>
<td>9.2 x 10^1</td>
</tr>
</tbody>
</table>

#### X Shim coil

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{0,0}$</td>
<td>0.1 x 10^-4</td>
</tr>
<tr>
<td>$a_{0,1}$</td>
<td>2.1 x 10^-2</td>
</tr>
<tr>
<td>$a_{0,2}$</td>
<td>4.1 x 10^-2</td>
</tr>
<tr>
<td>$a_{0,3}$</td>
<td>3.1 x 10^-2</td>
</tr>
<tr>
<td>$a_{1,1}$</td>
<td>3.4 x 10^2</td>
</tr>
<tr>
<td>$a_{1,2}$</td>
<td>-4.4 x 10^-2</td>
</tr>
<tr>
<td>$a_{1,3}$</td>
<td>5.1 x 10^-1</td>
</tr>
<tr>
<td>$b_{1,1}$</td>
<td>-5.2 x 10^-2</td>
</tr>
<tr>
<td>$b_{1,2}$</td>
<td>-1.2 x 10^-2</td>
</tr>
<tr>
<td>$b_{1,3}$</td>
<td>5.1 x 10^-1</td>
</tr>
<tr>
<td>$a_{2,2}$</td>
<td>2.4 x 10^-2</td>
</tr>
<tr>
<td>$a_{2,3}$</td>
<td>2.1 x 10^-3</td>
</tr>
<tr>
<td>$b_{2,2}$</td>
<td>6.2 x 10^-2</td>
</tr>
<tr>
<td>$b_{2,3}$</td>
<td>5.2 x 10^-2</td>
</tr>
<tr>
<td>$a_{3,3}$</td>
<td>9.2 x 10^-3</td>
</tr>
<tr>
<td>$b_{3,3}$</td>
<td>1.2 x 10^-3</td>
</tr>
</tbody>
</table>
Figure 2.2 Coordinates for the computation of the field $B_z(x,y,z)$ of a current filament in the form of circular arc.

X-shim coil should have only $a_{1,1}$ coefficients other coefficients should be zero. But it is impossible to design totally pure X-shim coil. The various shim coils and their functions are described in table 2.3. The design parameter for this coil is not available in literature, these are mostly patented. So for our simulation purpose we considered that the shim coils field are completely pure.
### Table 2.3 Equation of field generated by shim coils

<table>
<thead>
<tr>
<th>Shim</th>
<th>Equation for Field Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z_0$</td>
<td>1</td>
</tr>
<tr>
<td>$Z_1$</td>
<td>$z$</td>
</tr>
<tr>
<td>$Z_2$</td>
<td>$2z^2 - (x^2 + y^2)$</td>
</tr>
<tr>
<td>$Z_3$</td>
<td>$z[2z^2 - 3(x^2 + y^2)]$</td>
</tr>
<tr>
<td>$Z_4$</td>
<td>$8z^2[z^2 - 3(x^2 + y^2)] + 3(x^2 + y^2)^2$</td>
</tr>
<tr>
<td>$Z_5$</td>
<td>$48z^3[z^2 - 5(x^2 + y^2)] + 90z(x^2 + y^2)^2$</td>
</tr>
<tr>
<td>$X$</td>
<td>$x$</td>
</tr>
<tr>
<td>$Y$</td>
<td>$y$</td>
</tr>
<tr>
<td>$ZX$</td>
<td>$zx$</td>
</tr>
<tr>
<td>$ZY$</td>
<td>$zy$</td>
</tr>
<tr>
<td>$X^2 - Y^2$</td>
<td>$x^2 - y^2$</td>
</tr>
<tr>
<td>$XY$</td>
<td>$xy$</td>
</tr>
<tr>
<td>$Z^2X$</td>
<td>$x[4z^2 - (x^2 + y^2)]$</td>
</tr>
<tr>
<td>$Z^2Y$</td>
<td>$y[4z^2 - (x^2 + y^2)]$</td>
</tr>
<tr>
<td>$ZXY$</td>
<td>$zxy$</td>
</tr>
<tr>
<td>$Z(X^2 - Y^2)$</td>
<td>$[z(x^2 - y^2)]$</td>
</tr>
<tr>
<td>$X^3$</td>
<td>$x(x^2 - 3y^2)$</td>
</tr>
<tr>
<td>$Y^3$</td>
<td>$y(3x^2 - y^2)$</td>
</tr>
</tbody>
</table>
Chapter 3

GPU Architecture and Texture Based Volume Rendering

3.1 GPU Architecture

General purpose computation on GPUs [6] is being advocated in various non graphics areas. Applications have been found in numerical computation, cryptography, signal processing, and many others [5]. GPUs were designed as efficient coprocessors for rendering and shading. The programmability now available in GPUs such as the NVIDIA GeForce series makes them useful coprocessors for more diverse applications. Several programming languages are made available for creatively exploiting the GPU for non-graphic applications. OpenGL Shading Language (GLSL) [7], nVidia’s C for Graphics (Cg) [8], and Microsoft’s High Level Language (HLSL) [9] are some examples. Since the time between new generations of GPUs is currently much less than for CPUs, faster coprocessors are available more often than faster central processors. GPU performance tracks rapid improvements in semiconductor technology more closely than CPU performance. This is because CPUs are designed for high
performance on sequential operations, while GPUs are optimized for the high parallelism of vertex and fragment processing. Additional transistors can therefore be used to greater effect in GPU architectures. In addition, programmable GPUs are inexpensive, readily available, easily upgradeable, and compatible with multiple operating systems and hardware architectures. More importantly, interactive computer graphics applications have many components vying for processing time. Often it is difficult to efficiently perform simulation, rendering, and other computational tasks simultaneously without a drop in performance. Since our intent is visual simulation, rendering is an essential part of any solution. By moving simulation onto the GPU that renders the results of a simulation, it not only reduce computational load on the main CPU, but also avoid the substantial bus traffic required to transmit the results of a CPU simulation to the GPU for rendering. In this way, methods of dynamic simulation on the GPU provide an additional tool for load balancing in complex interactive applications. For the purpose of clarity here a brief model of GPU is presented (Fig.3.1). Detailed information can be found at [10]. It contains a vertex engine, fragment engine, a texture load/filter engine, and a depth-compare/blend data write engine. The vertex and the fragment processors are the user programmable stages of the pipeline. Vertex and fragment blocks operate on data serially and they both support floating point operands. The vertex processor operates on data, passing it directly to the fragment processor, or by using the rasterizer to expand the data into the interpolated values. Each triangle (or point) fed to the vertex processor becomes one or more fragments. The GPU support 16 bit floating point for four color components allowing for high dynamic range. GPUs differ by the shader capabilities and are identified by a particular shader model. nVidia GeForce 6 supports Shader Model 3.0.
3.1 GPU Architecture

Figure 3.1 Block diagram of the GeForce 6 Series Architecture. This GPU has 6 programmable vertex processors and 16 programmable fragment processors. (Reproduced from the book GPU Gems -2, [10])
3.1 GPU Architecture

3.1.1 3D Texture based volume rendering

CPU based scalar field visualization is highly compute intensive and is slow for real time interaction. One particular technique of interest is the one that uses hardware accelerated texture mapping [11]. This technique essentially resamples the scalar field (volume data set) that is represented as a series of 2D textures. A ghost plane normal to the viewing direction slices the scalar field and samples the sectional view, shown in Fig.3.2. GPU supports the necessary sampling techniques including the trilinear interpolation. An advanced technique was proposed by [12] that demonstrates both interactive volume reconstruction and interactive volume rendering with hardware provided 3D texture acceleration. The number of slicing planes or ghost planes is decided by the resolution of the 3D texture and the desired quality of rendering. The order of rendering of the slicing plane decides the alpha blending function. For

Figure 3.2 3D texture and slicing planes orthogonal to the view direction. Higher number of planes sample the scalar field more closely but increases the GPU passes.
3.2 Magnetic field computation on GPU

This section concentrates on the generation of magnetic field on GPU as a function of main coil and shim coil currents. Our first approach involves precomputing the 3D textures for various shim coil geometries and then linearly combining them in the GPU. The total magnetic field is described by \( B_{total} \) is the magnetic field due to a unit current and \( I_i, i = (0, 1, 2...) \) is the current factor. User interface provides the slider bars to change the \( I_i \), which is then encoded into an 'Uniform Parameter' array for the fragment shader program. The basic flow diagram is explained in Fig.3.3. We precompute the magnetic field 3D textures using Eq.(2.7) or Eq.(2.3) and coefficients from Table-2.2. the quad planes are supplied with proper 3D texture coordinates. A vertex shader program is written that transforms the quad plane and its texture coordinates to make it orthogonal to the view direction. The output of the vertex shader is fed to the fragment shader whose pseudo code is presented in Table-3.1.

\[
C_{dest} = (1 - \alpha_{dest}) \alpha_{src} C_{src} + C_{dest} \tag{3.1}
\]
\[
\alpha_{dest} = \alpha_{dest} + (1 - \alpha_{dest}) \alpha_{src} \tag{3.2}
\]

Rendering each plane requires one pass of GPU process. Some GPUs support multiple render targets (MRTs) wherein 4 or more pass can be included in a single rendering cycle. This feature particularly helps to speed up intermediate computation.
3.2 Magnetic field computation on GPU

Figure 3.3 Flow diagram of one render cycle. $I_i$ is encoded into an array and set as uniform parameter to the fragment shader.

texcoord.xyz. GPU generates the texture coordinates by interpolation, in our case it is set to trilinear. GPU allows for simultaneous operations on various data paths. The algorithm that is presented in Table-3.1 is parallelized in the GPU and hence several order of speedup is possible.

Precomputing 3D textures is compute-intensive task and it is best done in the GPU itself. Higher number of shim coils demand that many number of 3D Textures, which directly increases the memory requirement in $O(m \cdot n^3)$, where m is the number of 3D textures and n is the size of the texture. For $n = 256$ and $m = 16$ the memory requirement would be roughly 1024 MB, with 32 floats for each voxel. Latest graphics card doesnot have support for 1024 MB of memory. Thus, texture data will be stored in disk. Thus transferring data from disk to GPU will hurt the performance of the programme. To this end we propose an alternative algorithm that requires less memory but takes up more GPU computation. For each slice plane we compute the magnetic field directly from Eq.(2.3) or Eq.(2.7). This method as described in Table-3.2 still delivers the required frame rate for interactive rendering.
3.2 Magnetic field computation on GPU

Figure 3.4 Volume rendered with slicing plane 3D Textures

Figure 3.5 Image of magnetic field when inhomogeniety is almost removed by shimming
Table 3.1 Fragment shader for summing up the fields due to various shim coils.

```
float Bz0 = tex3D(Bz1Texture, texcoord.xyz);
float Bz1 = tex3D(Bz2Texture, texcoord.xyz);

// Sum up weighted components
BzTotal = I0 * Bz0 + I1 * Bz1 + ...
// Adjust to fit in 0.0 to 1.0
ScaleShift(BzTotal);
float4 color.rgba = tex1D(colormap, BzTotal);
```

Table 3.2 Algorithm for direct field computation. We embed Eq.(2.3) in the fragment shader. The coefficients $a_{m,n}$ are set as uniform parameters to the fragment program.

```
// Evaluate Eq.(2.3)(or Eq.(2.7))
float BzTotal = Evaluate( texcoord.xyz ) ;
// Adjust to fit in 0.0 to 1.0
ScaleShift(BzTotal);

float4 color.rgba = tex1D(colormap, BzTotal);
```
Chapter 4

Deformation Based Visualization Technique

4.1 Deformation Based Visualization Technique

Texture based volume rendering produces an image by mapping color to different type of inhomogeneity. Many different kind of inhomogeneity combine together to produce a single color. Thus, what kind of inhomogeneity has produced the color cannot be made out from the image. Subsequently, which shim coil to be adjusted to remove homogeneity cannot be decided from the image. Thus, to overcome that difficulty we propose an alternate visualization technique which deforms the geometry of visualization volume. The deformation geometry depends on the type of inhomogeneity present in the field. The cube shaped visualization volume is divided into a uniform 3D grid. All point on the grid occupied the original position if there is no inhomogeneity present in the field. The points get displaced from their original position if there is any inhomogeneity present in it. The displaced position of the
4.1 Deformation Based Visualization Technique

Points are given by equation (4.1).

\[ P' = P + c \nabla B_z \] (4.1)

Where \( P' \) is the position of the point \( P \) when a nonuniform magnetic field \( B_z \) is applied in the volume. \( c \) is a constant to be chosen carefully to accentuate inhomogeneity. If small value of \( c \) is chosen then effect of inhomogeneity cannot be visible whereas larger value of it will increase the size of image. As homogeneity is approached by optimally adjusting shim currents the deformation decreases. Thus to make deformations noticeable the parameter \( c \) is increased gradually. If the field is homogeneous then \( \nabla B_z \) vanishes and volume remains undeformed. If a linear inhomogeneity is present then each point in the volume will be displaced by a constant distance which resulted in translation of volume without any deformation. From the direction of translation it is possible to shim either of \( X, Y, Z \) coil to correct it. Other types of inhomogeneities will produce various deformed volume. The deformation of volume due to various inhomogeneity are shown in figure (4.4 through 4.19). The shim current which needs adjustment can be identified easily just looking at the deformed volume.

4.1.1 Implementation of Deformation Based Visualization

To implement it a vertex shader programme is developed to render deformed volume as given in algorithm (3.2). The visualization system being used is Pentium-4 workstation with n-Vidia GeForce 6600 graphics card, running Linux SUSE10. All graphics programmes are written using OpenGL. The volume is meshed to a \( m \times m \times m \) grid, \( m \) being the number of point in each dimension. The grid for is shown in figure (\( m = 84.1 \)). At each grid point the magnetic field and gradient is computed using equation 2.3. From magnetic field gradient the position of each grid point is computed. They are implemented in CPU. Then the data is transferred to GPU for visualization.
Figure 4.1 3D meshing volume for deformation based visualization. The volume is divided into $8 \times 8 \times 8$ grid points. The grid point changes their positions depending on the inhomogeneity present in the field.

However with this implementation when grid size exceeds $64 \times 64 \times 64$, the rendering becomes slower for user interactions. To overcome that difficulty, the magnetic field value and gradient are computed on GPU itself. A vertex shader is written in assembly to optimize the performance. The color and alpha value are computed to emphasize the point having high inhomogeneity. This is done by assigning alpha value to the modulus of gradient. The red, green and blue colors are assigned to the absolute value of each component of the field gradient.
Table 4.1 Algorithm for deformable volume based visualization. A vertex shader programme is developed to compute the position of points inside the deformable volume.

// Evaluate Eq.(2.3)(or Eq.(2.7))
float BzTotal = Evaluate( vertexpos.xyz ) ;

// Compute gradient and save it in 3D vector
vector3d grad.xyz =gradient(BzTotal)

// Compute position
Position.xyz =Position.xyz+c*grad.xyz

// Compute alpha value and color
// alpha value is the magnitude gradient
alpha=MagGradient(grad.xyz)

// color value is absolute value of each component gradient
color.rgb=ABS(c*grad.xyz)
4.1 Deformation Based Visualization Technique

Figure 4.2 The execution time in CPU and GPU, at various grid size is plotted.

Figure 4.3 Speedup = \( \frac{CPU\ time}{GPU\ time} \). The speed up with respect to different grid size is plotted. For lower value of grid size speed up is less than 1.
4.1 Deformation Based Visualization Technique

Figure 4.4 Deformation of volume when $Z(X^2 - Y^2)$ inhomogeneity in magnetic field

Figure 4.5 Deformation of volume when $X^3$ inhomogeneity in magnetic field
4.1 Deformation Based Visualization Technique

Figure 4.6 Deformation of volume when $Y^3$ inhomogeneity in magnetic field

Figure 4.7 Deformation of volume when $Y$ inhomogeneity in magnetic field
4.1 Deformation Based Visualization Technique

Figure 4.8 Deformation of volume when $Z^2X$ inhomogeneity in magnetic field

Figure 4.9 Deformation of volume when $Y^2Z$ inhomogeneity in magnetic field
4.1 Deformation Based Visualization Technique

Figure 4.10 Deformation of volume when $Z^2 - X^2$ inhomogeneity in magnetic field

Figure 4.11 Deformation of volume when $XYZ$ inhomogeneity in magnetic field
4.1 Deformation Based Visualization Technique

Figure 4.12 Deformation of volume when no inhomogeneity in magnetic field

Figure 4.13 Deformation of volume when XY inhomogeneity in magnetic field
4.1 Deformation Based Visualization Technique

Figure 4.14 Deformation of volume when $Z$ inhomogeneity in magnetic field

Figure 4.15 Deformation of volume when no inhomogeneity in magnetic field
4.1 Deformation Based Visualization Technique

**Figure 4.16** Deformation of volume when $ZX$ inhomogeneity in magnetic field

**Figure 4.17** Deformation of volume when $Z^3$ inhomogeneity in magnetic field
4.1 Deformation Based Visualization Technique

**Figure 4.18** Deformation of volume when $Z^4$ inhomogeneity in magnetic field

**Figure 4.19** Deformation of volume when $Z^5$ inhomogeneity in magnetic field
Chapter 5

Conclusion and Future work

5.1 Conclusion and Future Work

The result of our texture based volume rendering is shown in figure [3.4][3.5]. When field is completely homogeneous the rendered image should look like image [3.5]. Though, deformable grid based visualization technique involves an overhead of computing gradient of the field at each grid points, but image produced by this technique can be used to identify the inhomogeneity present in the field. With the vertex programme gradient of magnetic field is computed on GPU itself which provides enough frame rates for user interaction. A comparative study is given in figure [4.3,4.2]. For the lower grid size the speed up is less than 1. However for better quality image we need to have greater grid size. For more than $25 \times 25 \times 25$ grid size the speed up is more than one. The result of deformation based visualization is shown in figure [4.4 through 4.19]

While computing the magnetic field the magnetic properties of the sample is not taken into consideration. The magnetic field distortion due to magnetic properties of sample is considerable at the edge of the sample. There are numerical models available
to compute magnetic field distortion due to sample. Thus, in future this effect can be modelled and implemented.
Bibliography


