To the Graduate Council:

I am submitting herewith a dissertation written by Mohammad Khurram Khan entitled "Use of ^{18}F-BPA PET Images and Image Registration to Enhance Radiation Treatment Planning for Boron Neutron Capture Therapy." I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Laurence F. Miller, Ph.D.
Major Professor

We have read this dissertation
And recommend its acceptance:

George W. Kabalka, Ph.D.

Lawrence W. Townsend, Ph.D.

Ron E. Pevey, Ph.D.

Vice Provost and Dean of
Graduate Studies

USE OF ^{18}F-BPA PET IMAGES AND IMAGE REGISTRATION TO ENHANCE RADIATION TREATMENT PLANNING FOR BORON NEUTRON CAPTURE THERAPY

A Dissertation
Presented for the
Doctor of Philosophy Degree
The University of Tennessee, Knoxville

Mohammad Khurram Khan
May, 2002

Thesis
2002b
.K43

Copyright © Mohammad Khurram Khan, 2002

All Rights Reserved

Dedication

Bismillah Irah-man Nira-him (I begin in the name of God, most gracious, most merciful)

This work is dedicated to my loving parents, Muhammad Nafis Khan and Maryum Un-Nisa Khan, who have stood besides me through all the ups and downs in life, and to my brother and sisters. For, they have continued be a source of encouragement and inspiration in my life.

I also dedicate this work to all the patients who volunteer to undergo revolutionary treatment protocols for the advancement of science. Truly, medical science could not progress without them.

Acknowledgement

I would like to acknowledge Chuck Wimperle and David Nigg at Idaho National Engineering Environment Laboratory (INEEL) for their technical support and for helping me in obtaining the appropriate licenses. I would also like to acknowledge the Department of Energy and INEEL for their financial support of this work.

More importantly, I would like to acknowledge Dr. Trent Nichols, Dr. George Kabalka, and Dr. Larry Miller for their continued guidance and support of this work. I have spent many afternoons discussing the details of this work with them. They have always kept me focused and motivated.

I would also like to acknowledge Dr. Ronald E. Pevey and Dr. Lawrence W. Townsend for their willingness to serve on my committee.

Abstract

Conventional radiation therapy coupled with chemotherapy and surgery has proven unsuccessful for Glioblastoma Multiforme (GBM) patients. A binary treatment therapy such as boron neutron capture therapy (BNCT) may be promising. The Monte-Carlo based simulation environment for radiation therapy (SERA) software is used to simulate the dose administered to a patient undergoing BNCT. The boron concentration used to determine the boron component of the dose has been measured directly during surgery. Point sampling and averaging of the tumor tissue resulted in an estimate of a uniform boron concentration scaling factor of 3.5. This factor is used to scale the boron component of the dose linearly and homogenously within the tumor and target volumes. Current methods should be improved. The average dose to the tumor cells by such a method could be improved by better methods of quantifying the *in-vivo* boron biodistribution. A better method of determining the *in-vivo* ^{10}B biodistribution includes radiolabeling para-Boronophenylalanine (p-BPA) with ^{18}F and imaging the pharmaceutical using positron emission tomography (PET). This biodistribution of ^{18}F-BPA can then be used to better predict the average dose delivered to the tumor and target regions.

To incorporate the PET ^{10}B biodistribution within treatment planning, several patients are selected, and a dynamic PET study is performed. The last frames of the PET images are registered with T1-weighted MRI images using the Statistical Parametric Mapping Algorithm (SPM-99). The registered ^{18}F-BPA PET images are then coupled with SERA results in a new computer software, PET2MRI.m, to linearly scale the boron component of the dose based on the *in-vivo* ^{10}B biodistribution determined from the PET image. A qualititative and

quantitative assessment of the dose contours is presented using the two approaches. Tumor volume, tumor axial extent, and target locations are compared between using MRI or PET images to define the tumor volume. In addition, peak-to-normal brain value at tumor axial center is determined for pre and post surgery patients using ^{18}F-BPA PET images. The differences noted between the registered GBM tumor volumes (range: 34.04 – 136.36 %), tumor axial extent (range: 20-150%), and the beam target location (1.27 – 4.29 cm) are significantly different in using T1 MRI images to define the volumes when compared with using ^{18}F-BPA PET images.

For all GBM patients, the ^{18}F-BPA PET based tumor volumes and tumor axial extents are significantly larger when compared with MRI based tumor and target volumes. The peak-to-normal brain values are also determined at the tumor axial center using the ^{18}F-BPA PET images. The peak-to-normal brain values using the last frame of the pre-surgery study for the GBM patients ranged from 2.05-3.4. The peak-to-normal brain values for time-weighted PET data ranged from 1.71-4. For post surgery time weighted PET data, the peak-to-normal brain value in the residual parts of the tumor ranged from 1.5-1.7. The PET based peak-to-normal brain values are significantly different when compared with the current value of 3.5 used in treatment planning within SERA. Qualitatively, boron dose contours are greatly shifted using PET images when compared with MRI images. Collectively, these differences can lead to significant reorientation of the beam and can significantly impact current BNCT treatment planning.

Table of Contents

I. Introduction 1

1.1 BNCT Basics 1

1.2 Neutron Transport Basics 5

1.3 Monte Carlo Methods 8

1.3.1 History of Monte Carlo Methods 10

1.4 Overview of SERA 11

1.5 Phase I/II Patient Selection Guidelines and Treatment Limits 13

1.6 Radiation Treatment Planning Overview 15

1.7 Treatment Planning Results Overview 19

1.8 Homogeneous Vs. Heterogeneous Assumption 22

1.9 Verification of Linear Scaling Assumption 28

1.10 Significance of Work 33

II. Overview of the Physics and Image Reconstruction Behind MRI and PET 35

2.1 MRI 36

2.2 PET 39

III. Simulated Environment for Radiotherapy Applications (SERA) 50

3.1 Conversion of Images into the Appropriate Format 52

3.1 SeraImage 58

3.2 SeraModel 60

3.3 Sera3d 64

3.4 SeraCalc 66
3.5 SeraDose 72
3.6 SeraPlot 73
3.7 SeraPlan 73
IV. Image Registration 75
4.1 Resampling of Images 75
4.2 Smoothing 77
4.3 Affine Transformations 77
4.4 Optimization 81
4.5 Intra-Modality Registration Theory (SPM Approach) 85
V. Statistical Uncertainty with SPM 92
VI. Dose Adjustment 94
VII. Statistical Uncertainty with SERA 97
VIII. Statistical Uncertainty with PET Pixel Intensity 98
IX. PET2MRI Algorithm 102
X. Results and Discussion 104
XI. Conclusions 125
XII. Future Work 126
References 127
Appendixes 132
A: Example of SERA Monte Carlo Input 133
B : MCNP 4b Code to Verify Linear Assumption 134

C: MRI_Imagereconstruction.m 136

D: Filter_backprojection.m 137

E: TIME_INTEGRATED.m 140

F: ecatfile.m function to read ECAT6.4/7.0 data 142

G: MRI2RAW.m : Converts MRI images into RAW format suitable for SERA 180

H: Input File for MRI2RAW.m 183

I: Read_registration.m 184

J: ECAT2SERA.m 185

K: Example of seraCalc Input File 187

L: Euler_transform8.m program to implement rigid body transformation/translation 189

M: MRI2RAW.m 194

N: PET2MRI.m 197

O:PEAK_TO_AVERAGE.M 201

P: QHD_READ.M 203

Q: CDF_WRITE.m 206

R: FILE_LOCATOR.m 210

S: CDF_WRITE.m 213

T: Dose Contours and 3D Images of Other Patients 216

Vita 246

List of Figures

Figure 1: Cellular Level View of BNCT 2

Figure 2: Molecular Structure of p-Boronophenylalanine (BPA) Compound Currently Used in BNCT Clinical Trials at H/MIT and BNL 4

Figure 3: Boltzmann Transport Equation 6

Figure 3: Region Editing in SERA using SERA MODEL module 17

Figure 4: Volume Rendering in SERA using SERA3D Module 18

Figure 6: Beam Positioning in SERA-Axial, Sagittal, and Coronal Views of the Beam 20

Figure 7: Dose Volume Histogram for Brain and Target (tumor) Regions 23

Figure 8: Dose Depth Profile (Dose Vs. Depth (in cm)). 24

Figure 9: 2D Dose Contour Displays 25

Figure 10: MCNP Setup to Test the Linear And Homogeneous Boron Scaling Assumptions 31

Figure 11: Verification of Linear Assumption (Concentration < 100 pm) 31

Figure 12: Verification of Linear Assumption (Concentration > 100 pm) 32

Figure 13: Dose Depth Profile of the "Homogeneous" Assumption 32

Figure 14: 3-D View of Unreconstructed MRI Data 37

Figure 15: 2-D View of Unreconstructed MRI Data 37

Figure 16: MRI Image Reconstruction Using Fast Fourier Transform (FFT) 38

Figure 21: Reconstructed PET Image Using FBP 42

Figure 22: PET Pixel Intensity for a ^{18}F-BPA Reconstructed PET Image 43

Figure 23: Filtered Backprojection Image Reconstruction Using Convolution Method 46

Figure 24: Filtered Backprojection Image Reconstruction Using Inverse Radon 47

Figure 25: Side by Side Comparison of Individual Frame Image Vs. Time Integrated Image 48

Figure 26: 3D View of Individual Frame Data, Time Weighted/Integrated Data, and the Scaled Difference 49

Figure 27: Overview of MRI2RAW.m Subroutine to Convert MR Images Into SERA Compatible Format 57

Figure 28: Bodies, Image Slices, and Body Slices 59

Figure 29: SeraModel GUI Used to Define Various Volumes of Interest 62

Figure 30: Region Definition Using Univels 64

Figure 31: Patient Geometry Setup Within Sera3d 65

Figure 32: Sera Calc Graphical User Interface 67

Figure 33: Target Location 68

Figure 34: Polar Angle Beam Rotation From Top of the Head to the Right Side of Patient ... 69

Figure 35: Azimuthal Rotation From Back of the Head to the Right Side of Patient 69

Figure 36: Use of Edit Directives To Generate Dose Volume Histogram and a Concatenated Dose Contour (.cdf.sz) File 71

Figure 37: Example of Bilinear Interpolation 76

Figure 38: Statistical Uncertainty With PET Pixel Intensities Using First Order Approximation for Individual Frame Data and Time Integrated Data 101

Figure 39: PET2MRI.m Program Overview 105

Figure 41: SPM Brain Segmentation of T1-Weighted MRI Images for VG 107

Figure 41: SPM Brain Segmentation for ^{18}F-BPA PET Images for Patient VG 108

Figure 42: Verification of Image Registeration for VG: Registered PET Image Contours Plotted on Top of MRI Images 109

Figure 43: Side by Side Comparison of Registered PET/MRI Tumor Volumes (top images) and a Superimposing of the PET Tumor Volume On Top of MRI Image (bottom image) 111

Figure 44: Registered Tumor Volume Differences Along the Z-Axis. A Side by Side Comparison of PET/MRI (top images) and a Superimposing of The PET Tumor Volume on Top of MRI (bottom image) 112

Figure 45: Original Boron Dose Contours 118

Figure 46: A Three Dimensional View of The Original Boron Dose Rate As Calculated by SERA 119

Figure 47: Boron Dose (or Dose Rate) Contours After Applying the Homogeneous Scaling Assumption 120

Figure 48: Three Dimensional View of Homogeneously Scaled Boron Dose Rate 121

Figure 49: Registered ^{18}F-BPA PET Adjusted Boron Dose (or Dose Rate) Contours 122

Figure 50: A Three Dimensional View of the Adjusted Boron Dose Rate (or Dose) Based on Registered ^{18}F-BPA PET Images 123

List of Tables

Table 1: BNCT Eligibility Criteria 13

Table 2: Irradiation Limits Observed During Clinical BNCT Treatments (* reprinted from Coderre[8]) 15

Table 3: Transformation/Translation Parameters Which Register PET Images with MRI Images 106

Table 4: Comparison of Tumor Axial Extent (in cm) Using Registered ^{18}F-BPA PET and T1-Weigted MRI images 113

Table 5: Comparison of Tumor Volumes (in cm^3) Using Registered ^{18}F-BPA PET and T1-Weigted MRI images 114

Table 6: Comparison of (x,y,z)Target Location (in cm) Using Registered ^{18}F-BPA PET and T1-Weigted MRI images 115

Table 7: Tumor Peak-To-Normal Brain (PNB) Value (@ Tumor Axial Center) Using ^{18}F-BPA PET 116

Table 8: Gray-Equivalent Mean Dose Administered to Patients with 50 min Irradiation at @ 2MW, assuming 13 ppm blood boron concentration, a Tumor-to-Blood uptake of 3.5, Tumor-to-Scalp uptake of 1.5 using MRI defined Volumes 116

Table 9: Gray-Equivalent Mean Dose Administered to Patients with 50 min Irradiation at @ 2MW, assuming 13 ppm blood boron concentration, a PET based Tumor-to-Normal Brain, Tumor-to-Scalp uptake of 1.5 using MRI defined Volumes 118

I. Introduction

About half of all tumors of the central nervous system are malignant cerebral gliomas which affect about 7000 Americans each year with only about 3% surviving past 5 years when treated with the most aggressive treatment protocols [1, 2]. With the most aggressive treatments including surgery together with chemotherapy and conventional photon radiotherapy, the survival time after diagnosis of a high-grade glioma may range from 9-12 months [3,4]. Recent advances in new treatment protocols such as boron neutron capture therapy (BNCT) may be promising not only for gliomas, but as well as for metastatic melanomas, and others. For reviews regarding the need for BNCT and its potential in future treatments for multiforme glioblastoma (GBM), multiple melanomas (MM), as well as other forms of cancers, the following references are suggested [1-12].

1.1 BNCT BASICS

Boron neutron capture therapy is a binary form of targeted radiotherapy first suggested by Locher in 1936, shortly after the discovery of the neutron by J. Chadwick in 1932 [5]. BNCT uses pharmaceuticals that contain a stable isotope of boron, ^{10}B, which preferentially localizes within the tumor cells. The cells are then irradiated with low energy neutrons. Because B-10 has a relatively high thermal neutron capture microscopic cross section (3837 barns @ .0253 ev), irradiation results in the capture of a neutron followed by a subsequent disintegration of the B-11 into an alpha particle (^{4}He) and a lithium particle (^{7}Li). As a result of this n-alpha (n,α) reaction and the slowing down of highly charged particles, a large amounts of kinetic energy are locally deposited within the tumor cells. With the combined range of the alpha and the lithium particle being around 14 microns, much of the

energy is deposited within a cell diameter or two [8]. If the particle path traverses the cell nucleus, then the probability of stopping subsequent mitosis and perhaps even killing the tumor cell is high. Thus, BNCT is selective in killing tumor cells while sparing surrounding normal cells. The key to effective BNCT is the preferential accumulation of ^{10}B within the tumor cell followed by adequate irradiation. Shown in equation 1 is the (n,α) reaction and the subsequent release of about 3 MeV of energy.

$$^{10}B + ^{1}n \rightarrow ^{11}B \rightarrow ^{7}Li + ^{4}He + 2.34\ Mev + 0.48\ MeV\ \gamma \qquad (1)$$

Shown in Figure 1 is a pictorial overview of BNCT as it occurs at the cellular and sub-cellular levels.

In 1951, Sweet and his colleagues first demonstrated in animals that certain B-10 compounds concentrated in human brain tumor cells relative to normal brain cells [11]. Shortly after, clinical trials were begun in 1951 and 1952 at Brookhaven National Lab (BNL) and in 1961 and 1962 at the Massachusetts Institute of Technology (MIT) research reactor. The B-10 compound used during these early trials was sodium tetraborate ($Na_2B_4O_7*10H_2O$). Clinical trials were also conducted in Japan in the 1960's.

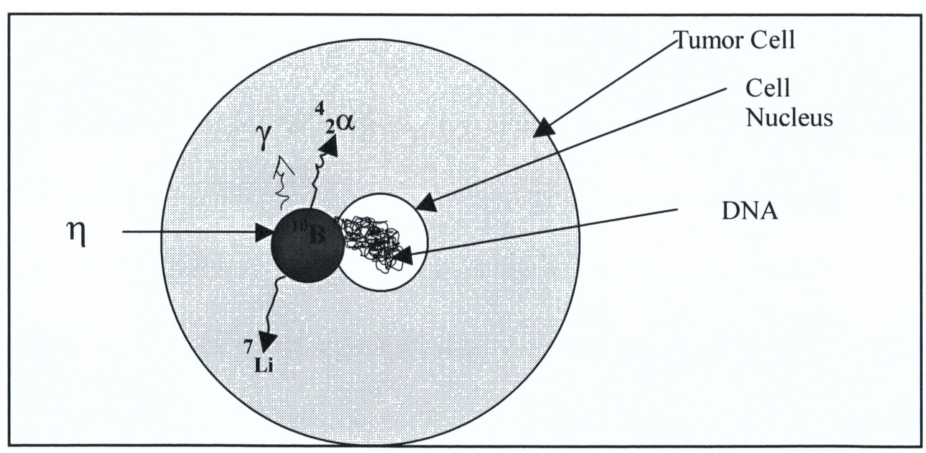


Figure 1: Cellular Level View of BNCT

Unfortunately, early clinical trials within the United States were unsuccessful for two reasons: (1) thermal neutrons used in the early trials only penetrated about 3-4 cm in tissue, thus deep seated tumors were spared, and (2) sodium tetraborate and other early compounds were diffusible, low-molecular weight, and those compounds that did selectively concentrate in the tumor cells had large background blood values. Hence, normal cells were radiated in large amounts along with the tumor cells [2]. Combined with thermal beam impurities, high background photon fluxes, deep situated tumors, and diffusible and low selectivity of the compounds in earlier clinical trials, patient health was jeopardized in some cases. This lead to the shutting down of earlier BNCT clinical trials within the United States in favor of further basic research and understanding of BNCT. Since 1968, over 150 patients with high-grade glioblastomas have been treated with BNCT in Japan. Reports of long-term survival in selected GBM patients in Japan, better beam designs, better pharmaceutical developments, and early success in rats with GBM spurred a keen interest in restarting BNCT clinical trials within the United States in the 1990s.

Recently, BNCT underwent Phase I/II clinical trials at Harvard Medical School and MIT (H/MIT) and at BNL. The objectives of the trials were: to evaluate the safety of a stepwise increase in BNCT dose to the normal brain using p-Boronophenyalanine (BPA) and epithermal neutrons, to evaluate any diverse effects of BNCT at each dose level, and to evaluate the therapeutic effectiveness of each BNCT dose level in patients with GBM. Shown

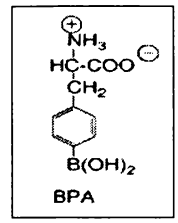


Figure 2: Molecular Structure of p-Boronophenylalanine (BPA) Compound Currently Used in BNCT Clinical Trials at H/MIT and BNL

in Figure 2 is the molecular structure of *para*-Boronophenylalanine (BPA), which was used within these Phase I/II clinical trials at BNL and H/MIT.

BPA is an amino acid analog of the naturally occurring essential amino acid, phenylalanine, needed for building of proteins. It is thought that since BPA is molecularly similar to an essential amino acid needed for protein synthesis, the growing tumor cells will readily take up this compound in larger concentrations when compared with normal non-proliferating brain cells. It has been shown in several studies that BPA concentrates in tumor cells (viable dense zones of tumor cells) by a factor of about 3-4 times relative to normal cells within the brain. However, this ratio of 3-4 is very variable *in-vivo*. This variability is seen within the tumor from the same patient as well as across patients [8-10]. Currently, there is no precise method for determining the *in-vivo* boron biodistribution within tumor cells. The best methods tend to average the concentration as determined from histological sampling during debulking.

Results from the initial dose escalation studies within the US demonstrated that BNCT doses used to date are safe. Patient survival after a single BNCT dose is slightly better than survival after six weeks of standard radiation therapy with or without chemotherapy, and patient quality of life after BNCT is better than with standard therapy [8,12]. Current results

from the phase I/II clinical trials have shown that the mean survival time with BNCT are around 13.5 months with an average recurrence time of about 6 months post debulking [13]. Based on initial phase I/II results and lessons learned, there is a continued need for improvement within BNCT if BNCT is to become fully acceptable as a treatment alternative for GBM and other cancers in the future. Better beam filter designs, purification of the neutron beam, a higher epithermal flux, coupling of epithermal BNCT with fast neutron therapy, better pharmaceuticals and understanding of the kinetics associated with these new pharmaceuticals, a better understanding of the macro- and microdosimetry, and a better understanding of the underlying biological mechanisms are just some of the basic research which continues to drive BNCT. Perhaps the biggest setback to BNCT today is the cost of having BNCT relative to standard chemotherapy and irradiation. However, this may change as more individuals opt to undergo BNCT, and if BNCT is shown to be successful in other cancers besides GBM.

1.2 NEUTRON TRANSPORT BASICS

The boron-neutron interactions at the cellular level can be modeled as an engineering system with various parameters and mathematical equations that describe the transport of neutrons through the materials before interacting with the boron atoms (and depositing some or all of their energies in the process). The general framework of the physics of such a process and a detailed mathematical description of neutron transport and their interactions with various materials can be found elsewhere [14, 15]. General neutron transport mathematics and physics can be readily implemented on biological systems with some modifications.

$$\frac{1}{v}\frac{\partial \phi}{\partial t} + \vec{\Omega} \bullet \nabla \Phi + \Sigma_t(\vec{r}, E)\phi(\vec{r}, E, \vec{\Omega}, t) - \iint \Sigma_s(\vec{r}, E' \rightarrow E, \vec{\Omega}' \rightarrow \vec{\Omega})\phi(\vec{r}, E', \vec{\Omega}', t)dE'\,d\Omega'$$

$$= \frac{1}{4\pi}\chi(E)\iint \nu(\vec{r}, E')\Sigma_f(\vec{r}, E')\phi(\vec{r}, E', \vec{\Omega}', t)dE'\,d\vec{\Omega}' + S(\vec{r}, E, \vec{\Omega}, t)$$

Where,

E = Neutron energy (J)
$\vec{r}$ = Position of Neutron (cm)
t = Time (s)
v = Neutron Velocity (cm s^{-1})
S = External Neutron Source (cm^{-3} s^{-1})
ν = Mean Number of Neutrons Produced Per Fission
Σ_a = Macroscopic Fission Cross Section (cm^{-1})
Σ_s = Macroscopic Scattering Cross Section (cm^{-1})
Σ_f =Macroscopic Fission Cross Section (cm^{-1})
Σ_t = Macroscopic Total Cross Section (cm^{-1})
χ = Normalized Fission Spectrum
ϕ = Forward Neutron Fluence Rate (cm^{-2} s^{-1})
$\vec{\Omega}$ = Unit Direction Vector
∇ = Gradient Operator

Figure 3: Boltzmann Transport Equation

The roots of transport theory date more than a century to the Boltzmann equation first formulated for the study of the kinetic theory of gases [14]. In 1940s, the Boltzmann transport equation was used to model neutron interactions as applied to nuclear reactors. The neutron transport equation that describes interaction between neutrons and materials and the physics behind such processes is derived from these kinetic equations developed earlier by Boltzmann, and hence the neutron transport equation is more commonly referred to as the Boltzmann transport equation. The Boltzmann transport equation is shown in Figure 3.

Several assumptions are made during the derivation of the time-independent non-multiplying neutron transport form of the Boltzmann equation. These assumptions are described in detail

elsewhere and briefly discussed here [14]. The assumptions include treatment of particles as points, particles travel in straight lines between collisions, particle-particle interactions are neglected, collisions are considered instantaneous, material properties are assumed to be isotropic, properties of nuclei and material compositions are generally assumed to be known and time-independent, and it is only the mean value of the particle density distribution that is considered.

With an understanding of such concepts as cross sections, particle distributions and statistics, fissioning of neutrons, primary and secondary (and scattered) particle streaming in three dimensions and the mathematics which describe this streaming in various coordinate systems, and applying certain boundary conditions, one can arrive at the time-independent neutron transport equation:

$$[\hat{\Omega}\bullet\vec{\nabla}+\sigma(\vec{r},E)]\psi(\vec{r},\hat{\Omega},E)=q_{ex}(\vec{r},\hat{\Omega},E)+\int dE'\int d\hat{\Omega}'\,\sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\bullet\hat{\Omega})\psi(\vec{r},\hat{\Omega}',E') \quad (2)$$

where,

$\hat{\Omega}'$ represents the direction in which particles are traveling,
$\hat{\Omega}$ represents the direction in which particles are scattered,
$d\hat{\Omega}'$ represents the solid angle from which the particles are scattered into,
$\vec{\nabla}$ represents the gradient operator which describes the movement of particles in and out of a particular volume element in three dimensional space,
$\psi(\vec{r},\hat{\Omega},E)$ represents the angular flux,
$\vec{r}$ is the three dimensional position in space,
E' represents the initial energy of the particles,
E represents the energy of the particles after collision,
$q_{ex}(\vec{r},\hat{\Omega},E)$ represents an external source,
$\sigma(\vec{r},E)$ represents the total cross section,
$\sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\bullet\hat{\Omega})$ represents particles which scatter into energy E and angle $\hat{\Omega}$.
$\hat{\Omega}\bullet\vec{\nabla}$ represents the loss of particles due to streaming,

$\sigma(\vec{r}, E)\psi(\vec{r}, \hat{\Omega}, E)$ represents the reaction rate (or collision density),

$q_{ex}(\vec{r}, \hat{\Omega}, E)$ represents the external source, and

$\int dE' \int d\hat{\Omega}' \sigma_s(\vec{r}, E' \rightarrow E, \hat{\Omega}' \bullet \hat{\Omega}) \psi(\vec{r}, \hat{\Omega}', E')$ represents particles traveling with energy E' about dE', at an angle $\hat{\Omega}'$ about $\partial\hat{\Omega}'$, and scatter into energy E and angle $\hat{\Omega}$.

Thus, balancing loss of particles with gain of particles within a given 3D volume element and applying various assumptions, one arrives at the time-independent non-multiplying form of the Boltzmann transport equation listed above.

1.3 MONTE CARLO METHODS

The neutron transport equation can be computationally simulated in at least three ways, using deterministic methods, integral transport methods, or Monte Carlo methods. In deterministic methods, the energy variable is generally discretized and hence is commonly referred to as the multi-energy-group approximation. The discrete ordinates methods deal with space and angular variable in such a way as to yield systems of simultaneous equations suitable for solution on a computer by discretizing the space and angular variables. Details of the discrete ordinates methods (and certain anomalies such as ray effects which can arise as a result of discretizing space and angle) and the integral transport methods used to solve the Boltzmann transport equation can be found elsewhere [14,15].

The most common and preferred method for complex three-dimensional geometries encountered in medicine is the Monte Carlo method. There are several advantages to using Monte Carlo methods for medical purposes. For one, complex geometries can by easily represented within Monte Carlo. Monte Carlo methods also avoid other systematic errors that

arise in discrete ordinates methods by making no discretization approximations regarding the treatment of energy, space, and angle. Errors with Monte Carlo tend to be stochastic in nature.

Monte Carlo methods consist of simulating a finite number of particle histories (N), through the use of a pseudo random number generator. In each particle history, random numbers are generated and used to sample appropriate probability distributions for scattering angles, track length distances between collisions, and so on. A history of the particle is terminated when the particle being tracked is captured or escapes the geometry of interest. However, if a particle is scattered, then a new energy is determined by conservation of energy and momentum principles. This process can be repeated until either the particle is absorbed or escapes from the medium (like in a void) or if the weight of the particle drops below some threshold value (the Russian roulette cut off weight). Because Monte Carlo methods tend to be stochastic in nature, large number of particle histories and batches must be run in order to assure reasonable uncertainty in the results. In most cases, the uncertainty decreases in proportion to $N^{-1/2}$. For example, in a BNCT simulation, if one used $2X10^6$ particles, and 1% of them, or $2X10^4$ are detected by a detector, then the error will be on the order of 0.7 % (i.e, $1/(2E4)^{-.1/2}$. Because of the stochastic nature of the process and for large geometries with small detector sizes, one may find it useful to use certain variance reduction techniques.

Advantages of the stochastic methods like Monte Carlo is apparent when modeling complex three- dimensional objects which may be difficult to represent in discrete ordinate methods. This is especially true for the medical images in which tumor regions, certain brain objects like the optic chiasma, and other regions of interest that might be too complex to

represent in discrete ordinates methods. However, Monte Carlo suffers in the fact that detailed distribution of a dependent variable, such as spatial profiles of flux or power may not be easily obtained since these quantities are generally not calculated at a point. They are only estimated from the number of collisions, particle track lengths, or from some other quantity in some incremental volume or phase space. Thus if one needed detailed spatial profiles, the problem would have to be divided into many small volume elements and the flux estimated in each of these cells. (Note that by reducing the size of the volume elements increases the amount of histories needed to get similar statistics). Thus, the statistical uncertainty grows in relation to the size of the volume element in Monte Carlo methods. A description of the Monte Carlo as applied within treatment planning and the statistical uncertainty with the estimated doses within treatment planning is described in a later section.

1.3.1 History of Monte Carlo Methods

The Monte Carlo technique is a statistical approach. It was first used in the early 18^{th} century. The development of the method is contributed to the French historian and mathematician, Georges Louis Leclerc Comte de Buffon, who in 1768 performed an experiment in his house. In the experiment he took loaves of bread of length L and threw them over his shoulder. He then looked to see if they crossed the lines on his tile floor, which had a spacing D. For L less than or equal to D, he obtained that the probability that the loaf intersected the tile is given by:

$$\mathbf{P = 2L/(\pi * D)} \qquad (3)$$

If the loaves are dropped N times and R intersections are counted then the total probability is estimated to be:

$$P = R/N; \quad (4)$$

With this procedure, one can estimate π through statistical methods:

$$\pi = 2LN/(RD) \quad (5)$$

A key feature of this method is the fact that the absolute error of the estimate decreases as $N^{-1/2}$, where N is the number of times that the procedure is repeated. Thus a good estimate for π can be estimated if one repeats the experiment enough times. Perhaps the real use of the Monte Carlo technique was shown in 1948 when Fermi, Metropolis, and Ulam obtained a solution of the Schroedinger wave equation with this method. The Monte Carlo technique is also used for macrodosimetry simulations and patient treatment planning within this work. Thus a basic understanding of the Monte Carlo process is essential to be able to use the treatment planning software in a meaningful manner and to be able to produce results that are statistically acceptable.

1.4 OVERVIEW OF SERA

Use of neutron radiation in clinical BNCT necessitates the use of computational software that allows for detailed 3D-volume reconstruction from medical images followed by radiation transport calculations that simulate the neutron transport equation to determine a certain quantity of interest. This quantity of interest may include the dose, the dose rate, the flux, reaction rate, or other quantities of interest. Investigators have utilized various approaches for 3D-volume rendering from medical images so that the geometry could be incorporated into Monte Carlo. At least three have been discussed in a recent paper [16]. At least two approaches are available to simulate the Boltzmann transport equation that describes transport of particles and their interactions in a given volume of interest. The resulting fluxes

and doses to the tissues can be determined by either discrete ordinates methods (DORT/TORT) or via the use of statistical methods like Monte Carlo.

Idaho National Engineering Environment Lab (INEEL), in conjunction with the Montana State University (MSU) computer science department, has developed a BNCT specific computational package (Simulation Environment for Radiotherapy Applications, or SERA) to simulate the solution to the Boltzmann transport equation using Monte Carlo techniques. SERA uses a modified non-uniform rational B-Spline (NURBS) combinatorial geometry (CG) method to reconstruct the detailed 3D uniform volume element (univels) geometry from 2D medical images followed by Monte Carlo methods to determine the dose to particular volumes of interest. The older version of SERA, or BNCT radiation treatment planning environment (BNCT_RTPE), did not allow for three-dimensional image rendering and visualization options as is currently implemented in SERA. However, the basics of transport calculations using Monte Carlo are the same in both simulation codes. Treatment planning and the resulting doses as described in various publications which used BNCT_RTPE for modeling is essentially the same since the Monte Carlo module of BNCT_RTPE did not significantly change with recent migration to SERA. The earlier BNCT_RTPE module has been benchmarked and tested against the MIT version of BNCT treatment planning software elsewhere [17]. It is SERA, which is the latest version of BNCT_RTPE that is used within this work to define the appropriate parameters and to simulate the transport equation to determine the doses to various regions of interest like the tumor, target, brain, scalp, and skull.

1.5 PHASE I/II PATIENT SELECTION GUIDELINES AND TREATMENT LIMITS

Selection of patients for BNCT was based on certain FDA approved guidelines for BNL's phase I/II clinical trials before radiation treatment planning could be administered. The guidelines by which patients were selected for Phase I/II clinical trials are shown in Table 1. These guidelines have been followed rigorously with treatment planning presented in this work. Further details pertaining to these guidelines can be found in the reference published by BNL physicists and medical personnel who conducted the BNCT treatments [8].

During phase I/II BNCT treatment, the administered dose of BPA-F is about 250 mg BPA/kg delivered as a 2 hour intravenous infusion. After infusion, irradiation is started approximately one hour after complete administration of F-BPA [8]. Blood boron concentration at the beginning of irradiation and during the midpoint of irradiation is measured to determine the duration of irradiation. The tumor-to-blood boron ratio is measured histopathologically by point sampling during debulking. This tumor-to-blood ratio along with the blood boron concentration is used to determine the boron component of the dose to the tumor regions. Generally, neutron irradiation is administered as a single

Table 1: BNCT Eligibility Criteria

1. Supratentorial lesion for which most likely clinical diagnosis is glioblastoma multiforme and diagnosis confirmed histopathologically.
2. Tumor must be unilateral and unifocal
3. Deepest portion of tumor margin <= 6.0 cm from skin surface
4. No prior radiation therapy (immunotherapy and or itrathecal chemotherapy excluded)
5. >18 yrs.
6. No pacemaker, metal prostheses, or metal implant in the head/neck region. (Dentures are allowed)
7. Karnofsky Performance Status >70.

fractionation/single field that lasts around 40-50 minutes (@ 2 MW thermal) using the BNL reactor [8]. Duration of irradiation is determined from the treatment setups and alignment of the beam to insure that certain FDA dose criteria are fulfilled with a particular treatment setup. Some technical considerations during beam alignment and duration of irradiation include insuring adequate dose to the tumor regions while minimizing dose to healthy tissue.

In addition to the patient selection guidelines, several dose limits were also set under the FDA phase I/II approved clinical trials. These dose limits were observed during the treatment session and served as key indicators for the validity of the treatment plan setup. These dose limits helped to determine the duration of the irradiation for a given treatment setup. For one, the irradiation time was adjusted such that no more than about 10.5 Gy-Eq would be delivered to the peak-dose volume within the normal brain endothelium. However, in recent publications and in some dose escalation studies, this limit was increased to 12.5 Gy-Eq. Other conditions that were observed are listed in Table 2 and are reprinted from Coderre [8].

When determining the dose to patients using BNCT, a special boron "Gray-Equivalent" quantity is defined. It uses a relative biological effectiveness of 3.2 for fast and thermal neutrons (^{14}N (n,p) ^{14}C reaction) , 0.5 for photons since a long irradiation time is involved, 1.3 for B-10 within normal brain tissue, and 3.8 for brain tissue. These values are used for treatment planning at BNL [13, 18, 19]. These RBE values are derived from cell survival data using the 9L rat gliosarcoma cells that were irradiated at BNL [19].

Table 2: Irradiation Limits Observed During Clinical BNCT Treatments (* reprinted from Coderre[8])

1. Prescribed absolute peak dose (to a 1 cm^3 volume) will be 10.5 Gy-Eq (12.5 in recent studies)
2. The peak dose rate will be kept below 27 cGy-Eq/min so as not to exceed that used by others for photon irradiation of the whole brain.
3. The average dose to the whole brain volume will not exceed 7.5 Gy-Eq.
4. The maximum dose to the scalp will not exceed 20 Gy-Eq.
5. The total radiation dose to the deepest part of the contrast-enhancing region of tumor should be >20 Gy-Eq.

1.6 RADIATION TREATMENT PLANNING OVERVIEW

Treatment planning using SERA is accomplished by providing patient specific 2D medical images in various formats, preferably in the RAW 8 bit 256X256 format or the QSH format. QSH is the standard internal format supported by SERA and developed by Macguire and Noz [40]. However, it will allow for other American Association of Physicists in Medicine (AAPM) standard formats (like Digital Imaging and Communications in Medicine (DICOM) format) to be converted to QSH. Once medical images are inserted into SERA, one can go about picking various points on the image to demarcate the boundaries of a given volume of interest. After the volumes are created, one can specify various materials that compose those volumes of interests. The materials and tissue properties are obtained from International Council or Radiation Units (ICRU) publication number 46 for brain, tumor, target, scalp, skull, optic chiasma, sinus, as well as other tissues of interest.

Figure 3 demonstrates the step by step picking of points in the SERA Model by using various threshold techniques or by manually painting a region of interest. Once individual points or regions are picked, one can volume render the regions of interest. Results of demarcating a volume of interest and using the modified NURBS combinatorial geometry method to volume render specific regions of interest are shown in Figure 4.

After volume rendering and generating a patient specific combinatorial geometry (CG) file, one can setup a Monte Carlo input using the SERA MC module wizard or enter it in manually. An example of a Monte Carlo input (described in detail in a later section) is included in the appendix section (Appendix A). In addition to the CG geometry file, one must also specify the source and the position of the beam within the input file. The source file used for BNCT treatment planning at BNL is a 12 cm disk source that consists of 47 neutron groups and 20 gamma groups. The source is predominantly an epithermal neutron beam with a maximum intensity of 4.94 X 10^9 neutrons/ cm^2/MW-Min occurring between 10 and 30 electron Volts, as is shown in Figure 5. The advantage of an epithermal neutron source is that one can get further penetration in tissue while preventing thermal flux depression in deeper parts of the tumor caused by attenuation in scalp and other superficial tissue.

Positioning of the beam is accomplished automatically or manually by providing the z-separation between the scalp and the beam aperture and specifying the azimuthal and polar angles of rotation. The azimuthal angle and the polar angle of rotation correspond to the rotation from the top of the head to the back of the head, and from the right side of the head to the left, respectively. An example of beam positioning as visualized within SERA is shown in Figure 6. The saggital, coronal, and axial views of patient geometry coupled with the beam geometry can be used to determine if the beam overlaps or conflicts with patient geometry. In the case where the beam overlaps certain boundaries of the patient geometry, erroneous results can be expected.

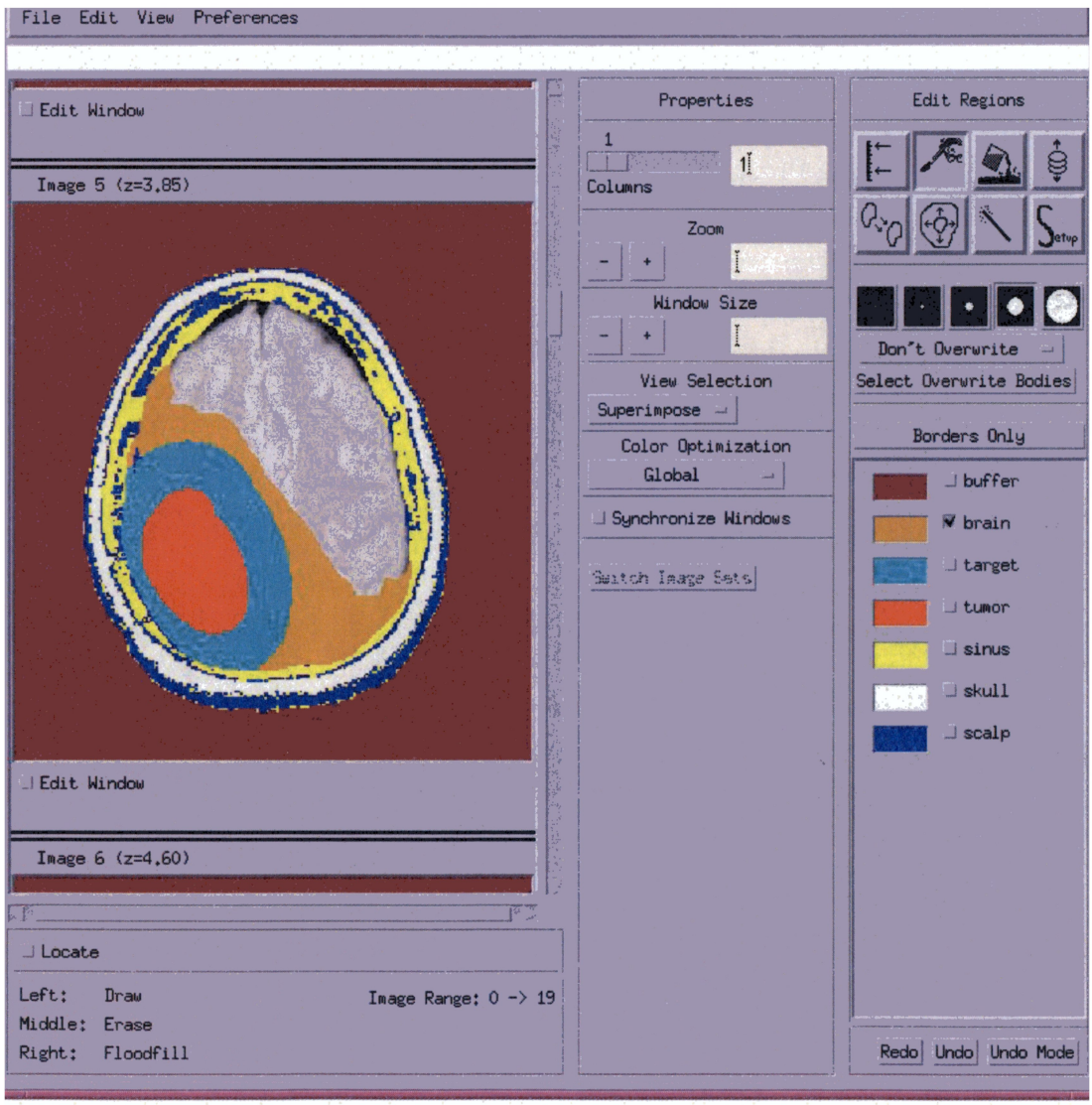


Figure 3: Region Editing in SERA using SERA MODEL module

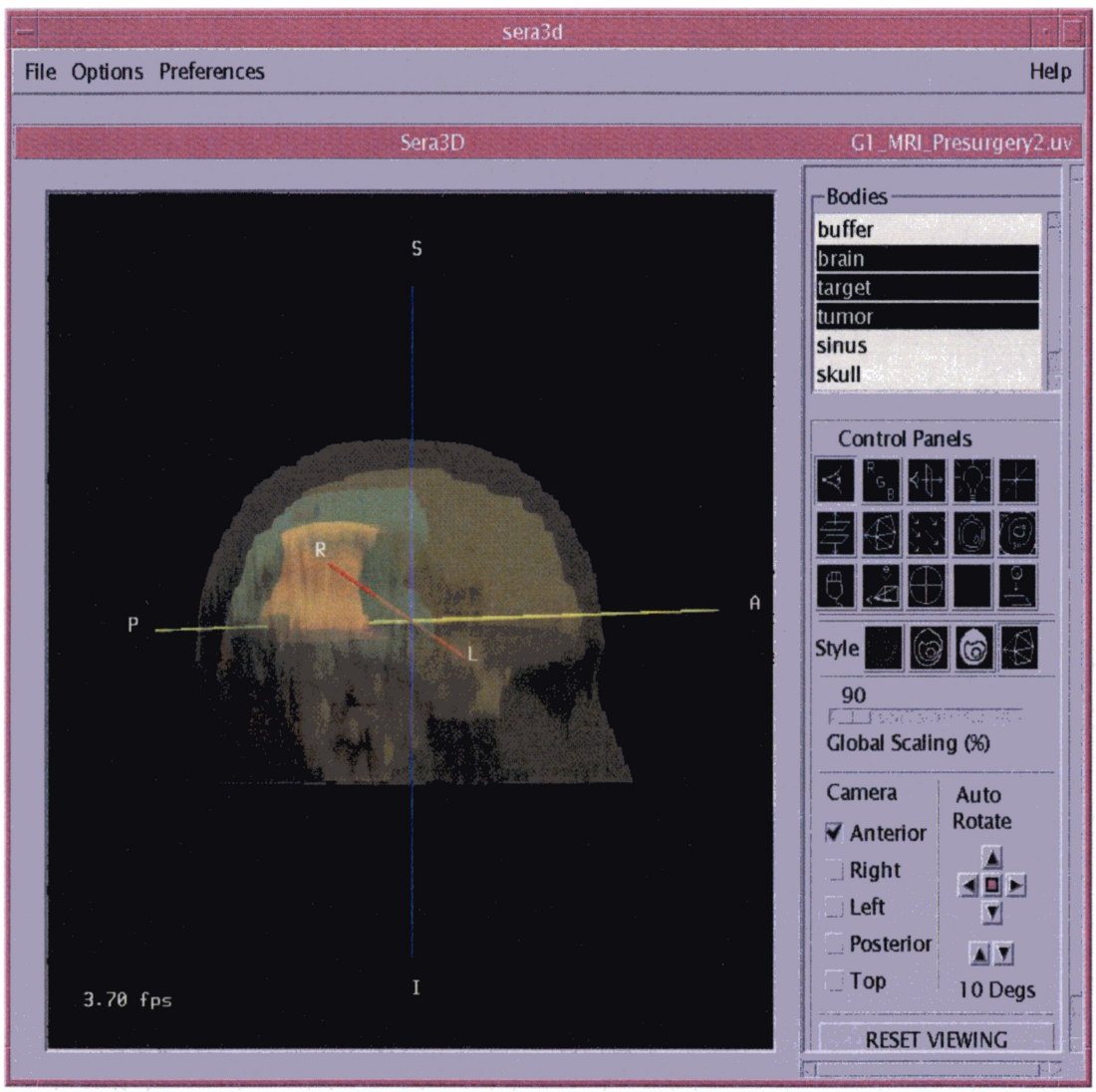


Figure 4: Volume Rendering in SERA using SERA3D Module

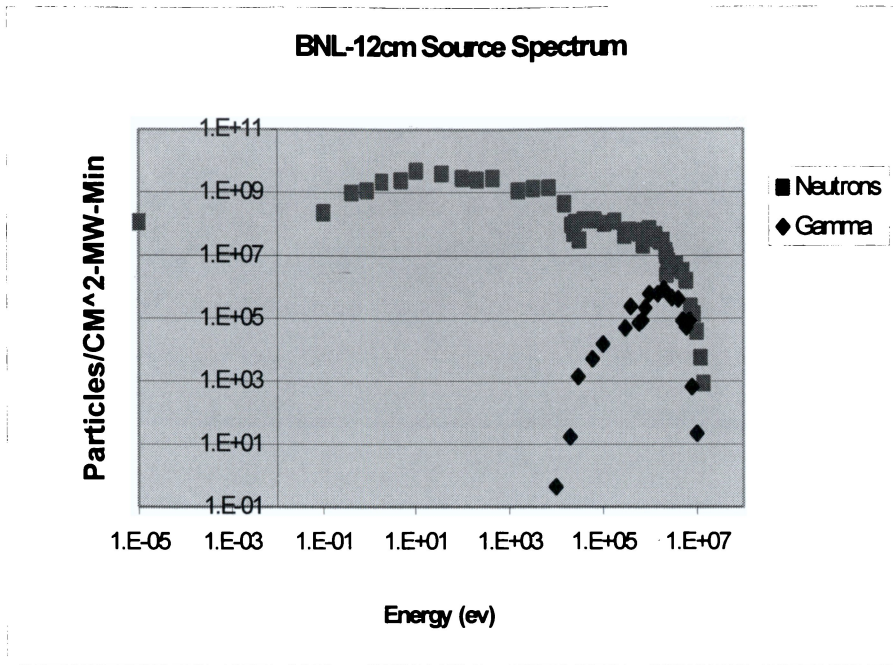


Figure 5: BNL 12 cm Neutron & Gamma Source Spectrum as used for BNCT

In most cases, too many particles will tend to get lost since there are overlapping boundaries. This can lead to increased statistical uncertainty in the dose estimates since many of the particles in overlapping regions will not be tallied. Thus beam alignment plays a critical role in the treatment planning and one should exercise careful examination of Figure 6. In addition to the losing of particles, alignment of the beam also plays a significant role in generating dose contours such that most of the 80% and 90% dose contours cover the center parts of the tumor while allowing for sparing of the normal brain.

1.7 TREATMENT PLANNING RESULTS OVERVIEW

Once the position of the beam and the source spectrum are specified, this information can be coupled with the geometry data, materials data, and cross section data to generate the

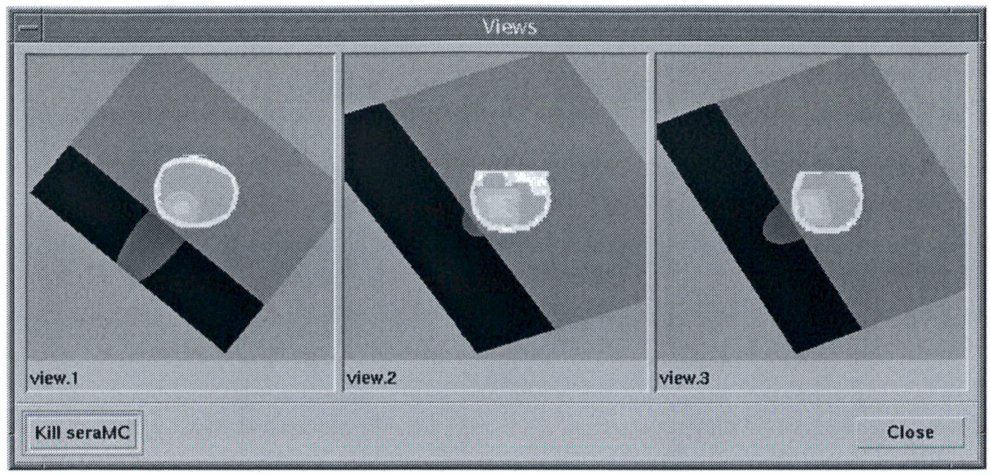


Figure 6: Beam Positioning -Axial, Sagittal, and Coronal Views of the Beam

minimum amount of input data needed to complete a Monte Carlo simulation. **SeraMC** module allows for these files to be used in combination and to simulate the boron neutron interaction to determine a statistically simulated solution of the Boltzmann transport equation. Results of the transport calculation include dose volume histograms that demonstrate the percentage of a given volume receiving more than a specified dose. In addition, the results of **SeraMC** also include individual dose contours (in 40X40, 128x128, or some other user specified grid size) that overlay the individual image planes. Shown in Figures 7 and 8 are dose volume histograms and a dose depth profile, respectively, for a treatment setup using one of the MRI patient data. The results shown in Figures 7 and 8 are for the BNL 2 megawatt reactor with a 50 minute exposure time and a maximum allowable limit to normal brain set at 12.5 Gray-Equivalent. Figures 7 and 8 can be used to assess the efficacy of the treatment plan to ensure that any treatment plan set follows the FDA approved guidelines for the BNCT clinical trials. Doses and dose rates shown in Figures 7 and 8, respectively, include relative biological effectiveness (RBE) factors and a boron concentration factor. Thus, the doses and

the dose rates displayed are more commonly referred to as "boron dose equivalent" or "boron dose rate equivalent" and can be used to compare with conventional photon therapy. The terms, "boron dose equivalent" and "boron dose rate equivalent" are a departure from the traditional sense of dose equivalent and dose rate equivalent since the relative biological effectiveness (RBE) of 3.8 and the boron concentration factors of 3.5 are used to compute the "boron dose equivalent" or the "boron dose rate equivalent". The terms "dose equivalent" and "dose rate" equivalent used throughout this work only refer to the dose derived from using BNCT to reach some end objective in mind. Since BNCT is tumor specific, there tends to be a larger amount of dose delivered to the tumor relative to normal brain when compared with conventional photon therapy.

The dosimetry data can also be shown as either a contour plane that overlay an individual plane or three dimensionally with volume rendering with SERA, as is shown in Figures 9 and 4, respectively. However, note that the dose contours in Figures 4 and 9 do not include the boron scaling factors, thus the dose contours are seen as smooth oval like in nature. However, after applying "correction" factors such the boron concentration factor, Relative Biological Effectiveness (RBE) factor, and the normalization factor, the dose would be "scaled" in a fashion as that shown in Figures 7 and 8. More specifically, one can notice that as a result of the "homogeneous" scaling of the dose, one gets a sharp discontinuity in the dose depth profile for the boron component of the dose at the target/normal brain boundary, as is evident from Figure 8. Such discontinuity in the dose in the tumor and target volumes may not represent the actual tissue level boron distribution and hence the dose within those regions. Thus, the dose shown in these Figures for the tumor and target regions may not be patient

specific and non-representative of the macroscopic heterogeneous localization of B-10. One of the ways to resolve such discrepancy and to get patient specific boron dose which accounts for this heterogeneous localization of the dose is to utilize registered ^{18}F-BPA PET images during treatment planning and dose prediction.

1.8 HOMOGENEOUS VS. HETEROGENEOUS ASSUMPTION

During the "correction" factor stage of the treatment planning, the radiation dose that is predicted to be delivered to the tumor and target volumes is linearly and homogeneously scaled based on a B-10 concentration factor. This concentration is estimated from histopathological studies conducted during surgery [9,10]. Based on these studies, it is estimated that the boron concentration within all individual tumor cells is about 3.5 times greater than that in the blood. Similarly, it is estimated that the boron concentration within the normal brain regions is about the same as that in the blood, and the boron concentration in the scalp is about 1.5 times higher than the blood [13]. Based on the first 10 patients that underwent treatment at BNL, the average blood ^{10}B concentration was 13 ± 1.5 ug ^{10}B/g (range 11.2 to 15.4). Thus, based on the boron blood concentration, the doses to the tumor, target, brain, and scalp volumes is linearly and homogeneously scaled by 3.5, 3.5, 1, and 1.5, respectively. It is therefore assumed, that every tumor cell within the tumor volume, every tumor cell within the target volume (tumor volume + 2cm), and every scalp cell within the scalp region will contain 45.5 ppm, 45.5 ppm, and 19.5 ppm, respectively (where 1 ppm = part/million = 1 ug/g of B-10).

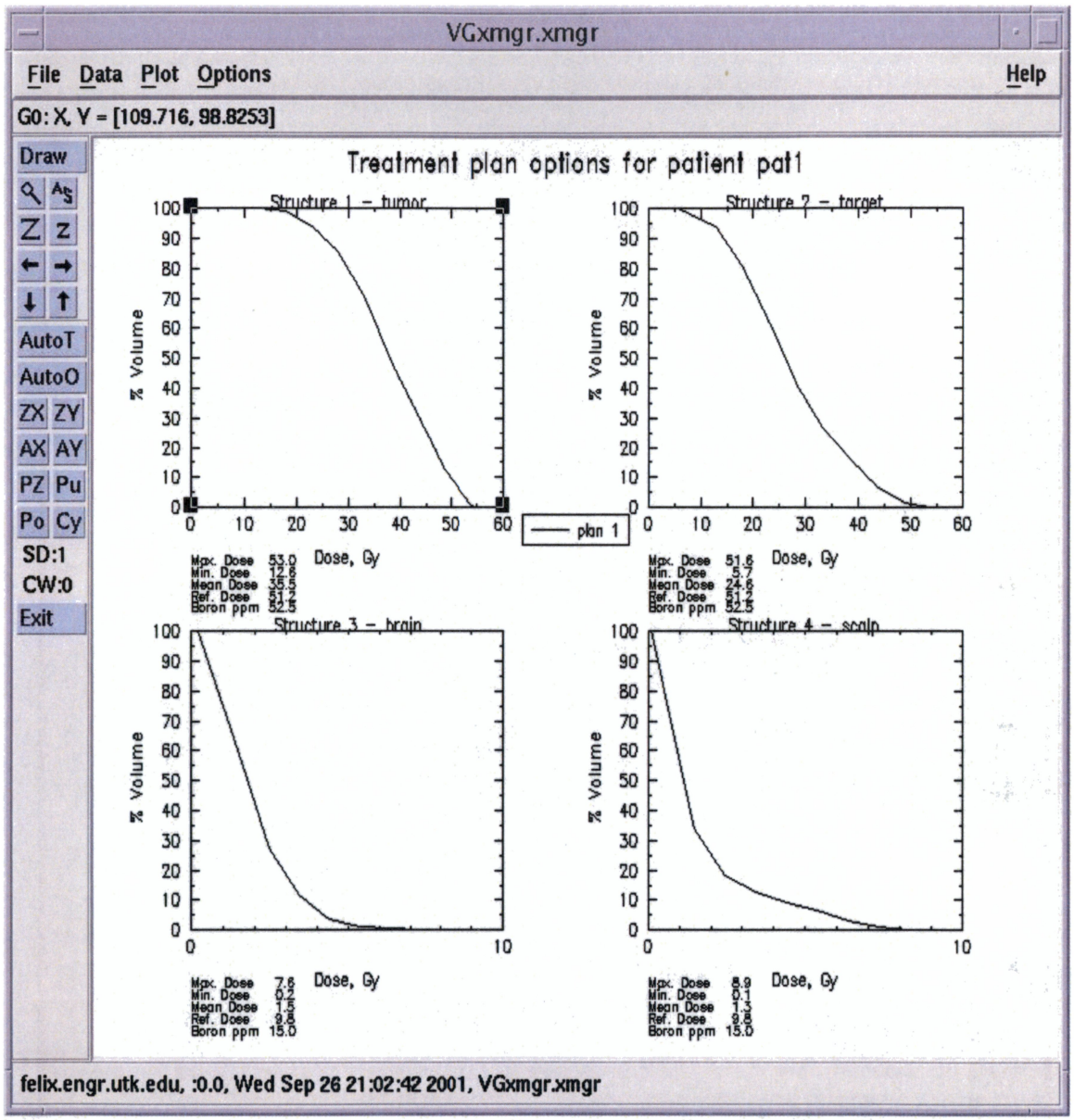


Figure 7: Dose Volume Histogram for Brain and Target (tumor) Regions

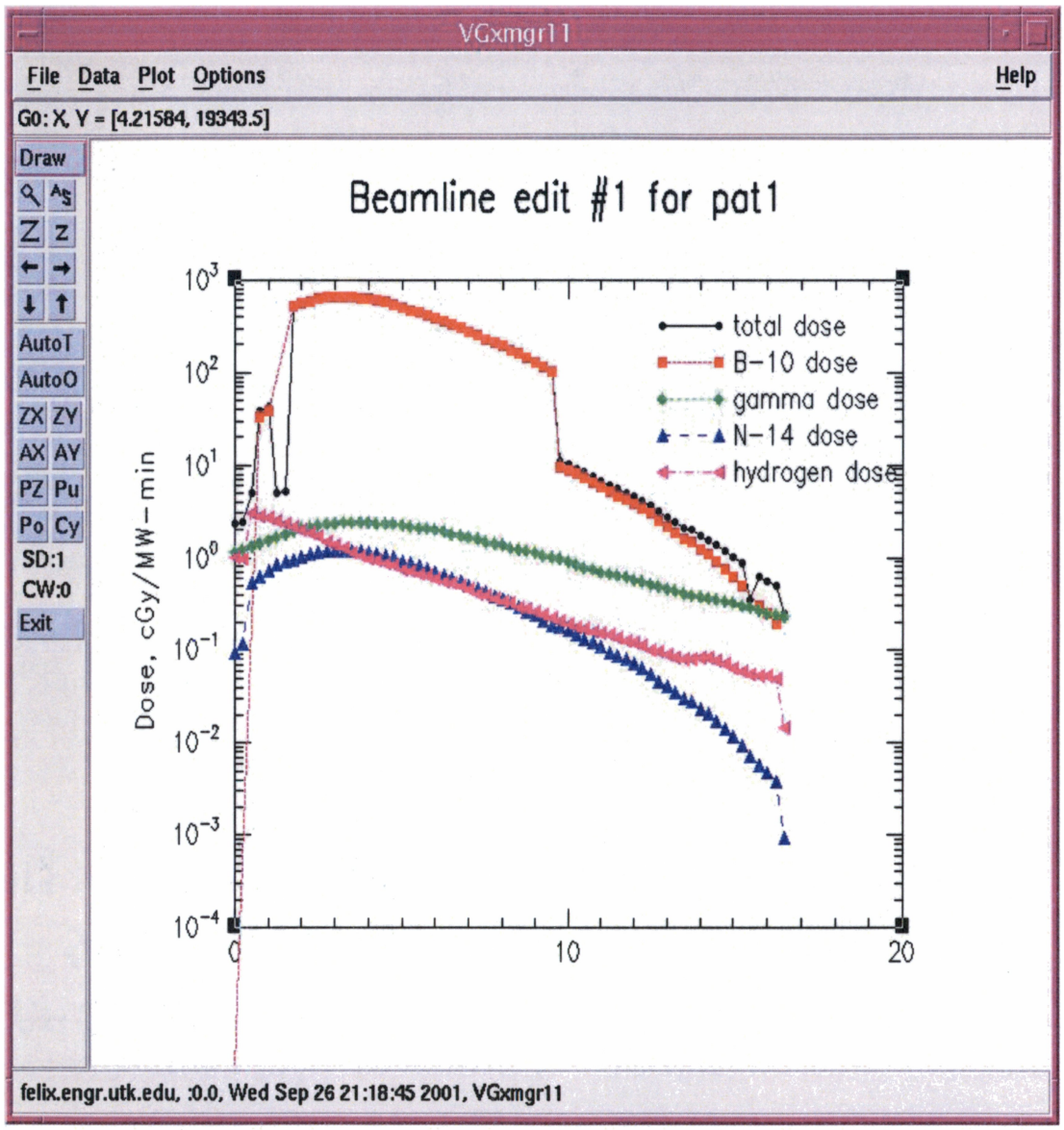


Figure 8: Dose Depth Profile (Dose Vs. Depth (in cm)).

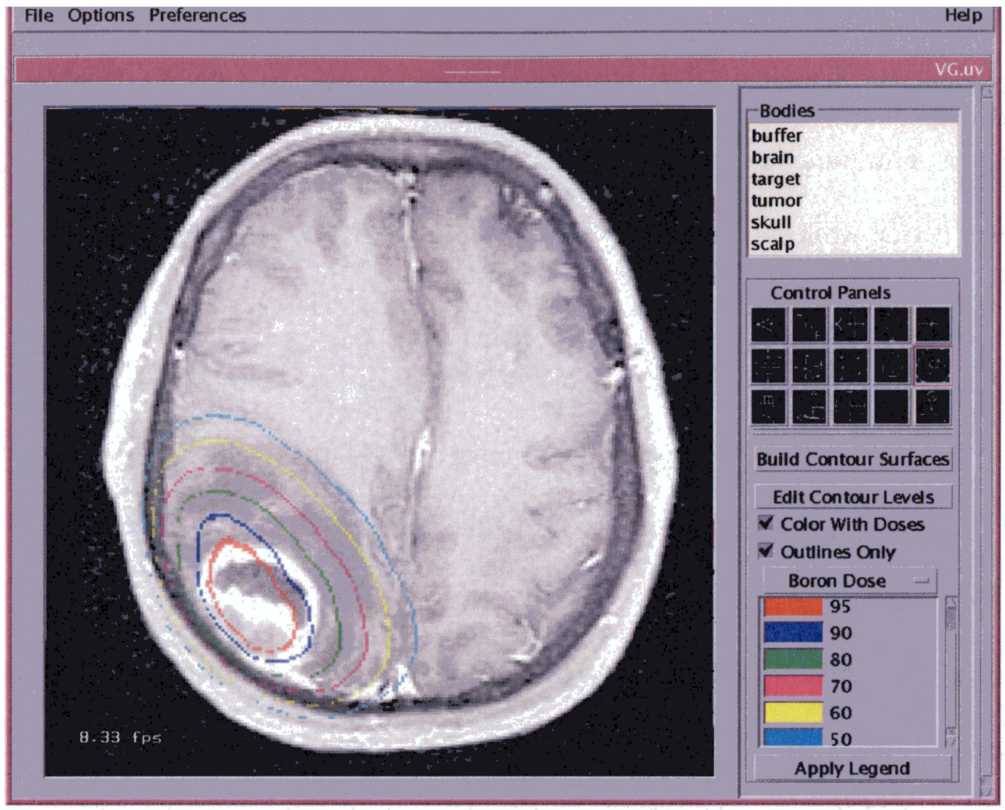


Figure 9: 2D Dose Contour Displays

The reason for defining a target volume that is 2 cm beyond the visible tumor volume in a MRI image is due to the fact that GBM tends to be proliferative in nature. The central tumor mass is known to have tumor cells or extensions of tumor cells which migrate beyond the main tumor mass. This extended region of tumor cells is generally believed to be about 1-3 cm for the GBM tumor [20]. Because of this proliferative nature of GBM, medical physicists tend to define a 2 cm target region beyond the tumor mass. The 2cm is chosen since it averages the 1-3 cm regions found in various studies. Also, more studies favor using the 2cm region when compared with the 3 cm region [20]. This region is also seen as the edema

region in an MRI image and surrounds the main tumor mass region. Within the BNCT community there is great debate about the exact distribution of boron and if the *in-vivo* boron biodistribution is actually homogeneous. One recent study concludes that histopathological determination of the tumor-to-blood or tumor-to-brain is highly variable [10].

Another debate while working with GBM tumor is due to the actual extent of the tumor beyond the tumor mass. What the exact extent of the tumor region beyond what is visible in an MRI image and the metabolic activity of those tumor cells is not known with any great certainty. Thus, this extended tumor region known as the target region is never precisely defined within SERA, and it is only assumed that this region will not extend beyond about 2 cm.

The linear scaling assumption is discussed in the next section and is justified as long as the boron concentration in a given region is below about 50 ppm. However, the homogeneous scaling of the dose within a given volume is not justified. There are several problems with assuming that every cell within a tumor volume and the target volume will exhibit similar metabolic characteristics in terms of taking up the boron (i.e., every cell within the tumor and target volumes will not have 45 ppm). In addition, the macroscopic dose being delivered to that volume cannot be homogeneously scaled. Some of the problems associated with this "homogeneous" scaling include the following:

1. Homogeneous scaling ignores near neighbor effects in which one alpha particle emitted in one cell may influence killing of nearby cells. This case was demonstrated in a recent study [21] that looked at irradiating individual cells and bombarding them with alpha particles and measuring changes in nearby cells. It was concluded that some factors

(perhaps mutated genetic material or other macromolecules) traveled between cells and promoted killing of nearby cells. This phenomenon is also referred to as the bystander effect.

2. The combined range of the alpha and lithium particles is about 14 microns. Thus, it is possible for an individual event to occur in one cell while having its impact on a neighboring cell (i.e., this more commonly referred to as the "cross-talk" effect in radiation biology textbooks).
3. Surgery may reduce availability of boron to peripheral tumor cells within the target region.
4. Individual cells communicate with one another through the use of gap junctions (and hence exchange intracellular material). There is also a possibility of having certain oxygen radicals (or other secondary messengers) which result from the irradiation traveling across cells and causing effects in neighboring cells.
5. Cells themselves are heterogeneously distributed throughout a volume. The tumor and target volumes contain ranging pockets of very densely packed tumor cells to parts where the tumor cells are loosely packed. Hence, the dose will be a function of this cellular density (and the boron contained within the cells).
6. Cells undergo different cell cycle stages. There is a random distribution of cells in various stages within the tumor and target volumes. There will be a cluster of cells which maybe in the S stage, while others will be in the G1 or G2 stages. The stage a particular cell is in can influence the B-10 concentration within the cell. Thus, in the tumor/target volume, not all cells will have equal B-10 uptake.

7. Tumor killing is a tissue level response. The killing of tumor is a collective tissue/macroscopic response rather than an individual cell killing response.
8. Based on *in-vivo* ^{18}F-BPA Positron Emission Tomography images, the boron biodistribution is heterogeneously distributed through out the volume. The heterogeneous distribution as seen in PET is better related to the cellular density and the metabolic function of the tumor cells and provides for a better *in-vivo* biodistribution than point sampling and averaging of various locations through out the brain tumor volume.
9. Current methods use point sampling and sample only densely packed and highly metabolic regions of the tumor to determine the average scaling factor. Point sampling is highly subjective to error and large variability [10].
10. The homogeneous assumption ignores patient tumor variability and staging of the tumor. Each patient's dose is scaled with the same "average" estimate.
11. The homogeneous assumption also ignores the dose "shadowing" effect caused by the depression of the epithermal/thermal flux within the tumor volume. This phenomenon is described in the next section.
12. Others have also favored use of functional PET images in treatment planning [22-24].
13. Some of the issues listed above are resolved by the use of registered ^{18}F-BPA PET images, others are not.

1.9 VERIFICATION OF LINEAR SCALING ASSUMPTION

Boron Neutron Capture Therapy Radiation Treatment Planning Environment (BNCT_RTPE) software and its latest version, Simulated Environment for Radiotherapy Applications (SERA) are used to determine the dose to the tumor, target, scalp, brain, and

other volumes of interest. It is assumed that one can place 1 ppm of boron within a particular volume to perform a dose calculation. Later, it is assumed that the dose based on 1 ppm of boron can be linearly scaled as a function of the actual boron concentration within that particular volume so long as the concentration within the volume is below 50 ppm. This is an assumption that is built into these computer programs and is referred to as the "linear" assumption.

This linear scaling assumption has been shown to be accurate so long as the boron concentration within a region is below about 50 ppm. This is generally the case for current clinical studies using BPA which accumulates in the tumor and target regions on the order of about 40-50 ppm. Thus, the linear assumption is valid for this type of treatment planning. However, this may change in the future with new compounds that may accumulate in larger amounts within the tumor and target volumes. A separate Monte Carlo code (MCNP 4b by Los Alamos National Laboratory) was run to verify this assumption (Appendix B). Essentially, the boron reaction rate is determined at various locations around the tumor and for varying boron tumor concentrations. An 8 cm disk source using the BNL source spectrum used in SERA was placed 20 cm to the left of a 20X20X20 cm^3 volume of water representing human tissue. The 5X5X5 cm^3 tumor volume containing varying concentrations of boron was placed 2 cm inside the left surface of the larger 20 cm cube water volume. A schematic diagram of the geometry is shown in Figure 10.

Shown in Figure 11 and 12 are the results for the MCNP calculations which were used to assess the linear scaling assumption. As can be seen from Figures 11 and 12, the boron-

neutron reaction rates are linear so long as the concentrations within the tumor (and target) volumes does not exceed 50 ppm. This allows one to place a few ppm of boron within a tumor, target, scalp, and brain volumes and post process the results based on the actual distribution of boron within these volumes.

Using the linear scaling assumption, one can plot the results in Figures 11 and 12 slightly differently. If one plots the reaction rate as a function of depth (while taking out the homogeneous concentration scaling factor), one will get a plot similar to that shown in Figure 13 where position 0 corresponds to the center of the 5X5X5 cm tumor volume. One can note from this Figure that there is a depression of the thermal and epithermal flux in deeper parts of the tumor (as can be seen at detector position 1 which corresponds to being 3.5 cm deep into the tumor). This results in under predicting the dose to deeper parts of the tumor by about 7.9 % for 50 ppm, regardless of the distribution of the boron within the volume. This percentage will vary depending on the spatial depthness of the tumor volume and the concentration of boron within the tumor. Although this study did not explore placing the tumor at various depths within the larger water volume, these differences could have been as large as 20 % [25].

One of the effects caused by the linear "homogeneous" scaling assumption is also referred to as the dose "shadowing" effect that results from the depression of the epithermal/thermal flux as one penetrates deeper and deeper into the tumor. The current linear homogeneous scaling assumption that assumes that the dose to the entire tumor volume would be the same is not valid, even if the tumor concentration throughout the entire volume was the same (as is the

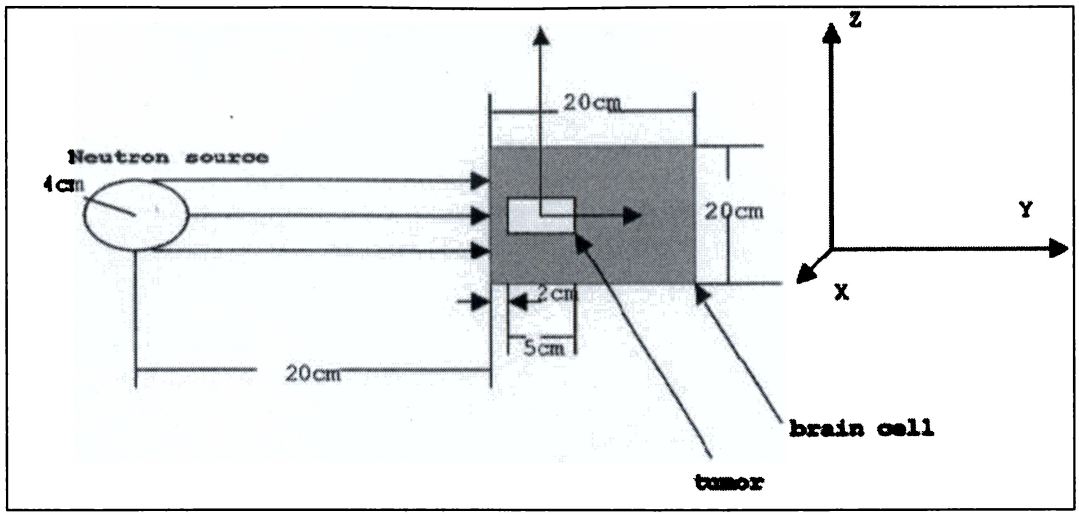


Figure 10: MCNP Setup to Test the Linear And Homogeneous Boron Scaling Assumptions

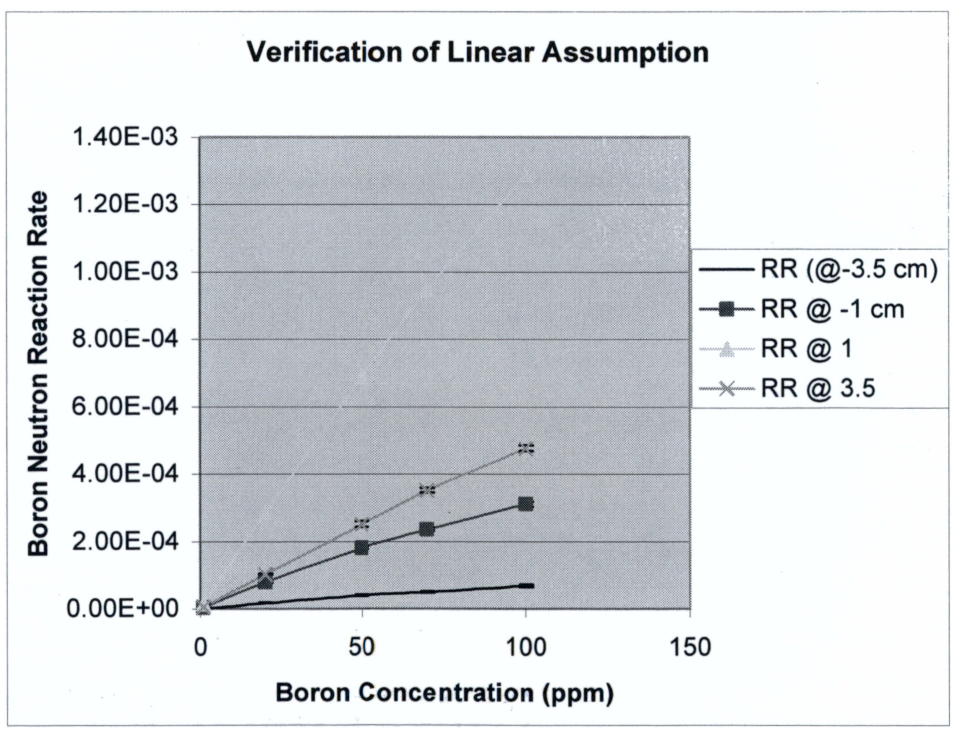


Figure 11: Verification of Linear Assumption (Concentration < 100 pm)

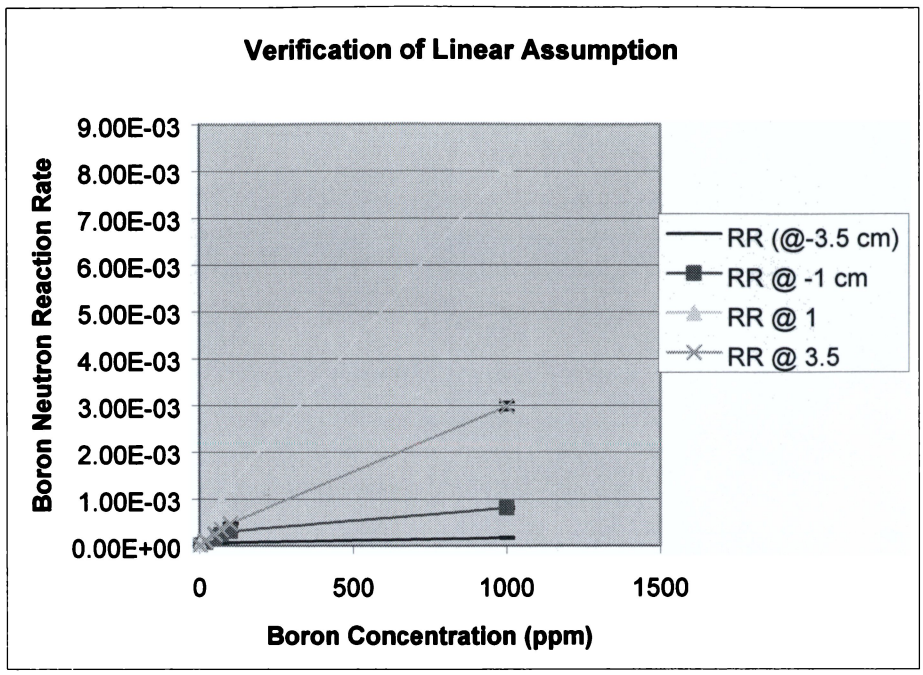


Figure 12: Verification of Linear Assumption (Concentration > 100 pm)

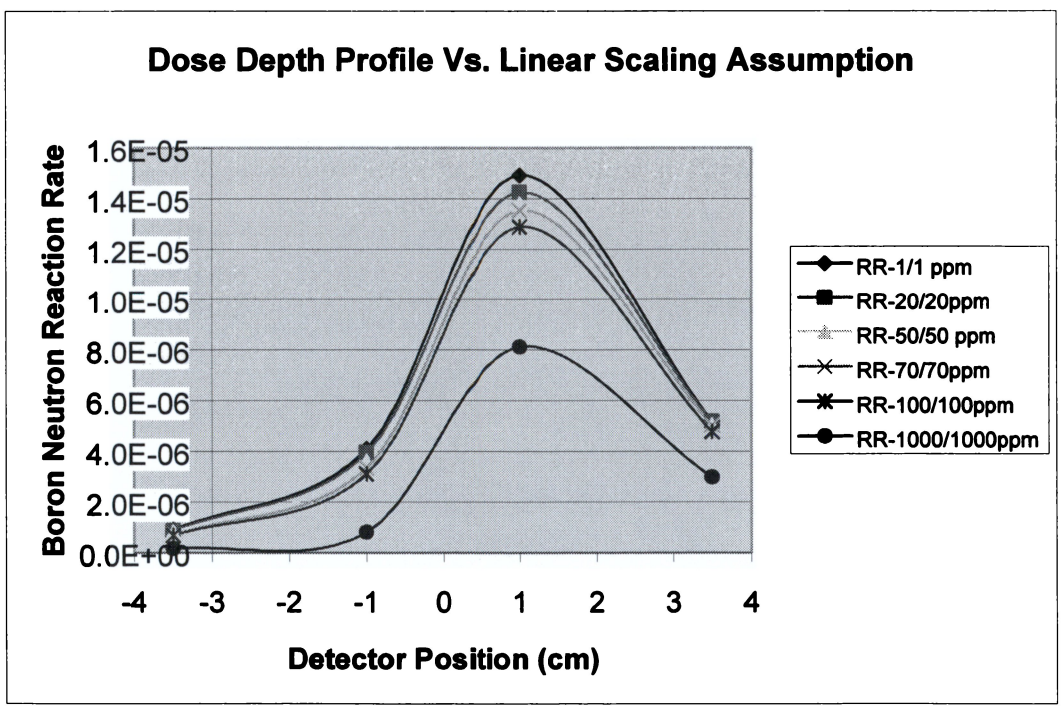


Figure 13: Dose Depth Profile of the "Homogeneous" Assumption

case in this simple Monte Carlo simulation in which the concentration throughout the entire tumor volume was kept the same.

If one couples this dose depth shadowing effect along with the fact that the biodistribution of the pharmaceutical within the tumor and target volumes is not homogeneous, one would expect greater discrepancies between what is predicted and the actual dose being delivered to the target and tumor volumes. Thus, ideal treatment planning for BNCT would couple the actual boron biodistribution as determined from a PET image along with PET/MRI image registration prior to the Monte Carlo simulation. Currently, SERA does not allow for incorporation of the boron biodistribution prior to treatment planning. One is limited to post processing the dose based on the linear assumption. This may be of greater importance in the future if new pharmaceutical accumulate at concentrations well beyond the 50 ppm range or if the administered boron concentration is optimized to allow for greater accumulation within the tumor regions.

1.10 SIGNIFICANCE OF WORK

Currently, no one has addressed the use of ^{18}F-BPA PET images for the purpose of improving BNCT radiation treatment planning for GBM patients. No methodology or computer codes are available which may allow for such an analysis. The author has developed a methodology and a computer algorithm to allow for such detailed incorporation of the ^{18}F-BPA biodistribution within current BNCT treatment planning using PET images. The methodology and the computational software are discussed within this work and are well supported by various references. The current methodology and computational software is

shown to greatly improve treatment planning. In addition to improving existing treatment planning, the author has also developed a technique that allows for better visualization and reconstruction of PET data using the time integrated PET data.

Besides developing a methodology to improve existing treatment planning using the heterogeneous boron distribution assumption and for better PET image visualization, the author has also used the registered ^{18}F-BPA PET images to compare differences between a metabolic view of the tumor as shown in the PET images with that of an T1-weighted MRI images. These differences are analyzed to determine the beam misalignment which can result between using MRI images or PET images to determine the tumor volume. Finally, the author also uses the ^{18}F-BPA PET images to determine the peak-tumor-to-normal brain values and compares that with the homogeneous scaling factor of 3.5. Although this kind of work is critical to BNCT treatment planning, this work has not been carried out by anyone.

II. Overview of the Physics and Image Reconstruction Behind MRI and PET

To better understand the need for Positron Emission Tomography (PET) images within BNCT treatment planning and the differences between T1 and T2 weighted MRI images in treatment planning, a basic overview of image reconstruction and the underlying physical processes governing these imaging modalities is helpful. The ^{18}F-BPA PET images represent physiological/functional information, whereas T1 MRI images represent anatomical information. The reconstructed PET and MRI images are used in treatment planning to set up the volume and to compute the appropriate dose to these regions. The intensities and the locations of the objects within these reconstructed images are also used to register PET and MRI images and to linearly adjust the dose based on the heterogeneous boron distribution. Thus, a basic understanding of image reconstruction and the physical processes underlying these images can provide useful insight to the inherent differences between the two modalities and how the two modalities can complement one another during treatment planning.

Generally, PET and MRI data retrieved from the University of Tennessee Medical Center exists either in raw (unreconstructed) format or in reconstructed format. Sometimes it is useful to be able to take the raw unreconstructed data and to create the object that is being imaged. For example, this may be the case, especially since much of the data at UTMCK is in proprietary machine specific format. Generally, the MRI data is in a General Electric Signa 5.0 format while the PET data is in either the CTI ECAT 6.4 or ECAT 7.0 data formats. These formats are cryptic and make it difficult for individuals to easily work with the data

(unless appropriate licenses and software are available). Thus, one would like to have separate algorithms that can use the raw data directly instead of having to obtain various licenses. Another case, where this maybe appropriate is if one would like to experiment with reconstructing data to remove artifacts from the image, to reduce noise, filter data using a different filter as opposed to the one used by the manufacture, remove blurring, enlarge images, change pixel size, etc.

2.1 MRI

Several 2D and 3D image reconstruction algorithms have been developed for PET and MRI images. Objects visualized in PET images tend to be quite different from those visualized in MRI images. MRI images generally rely on the differential decay and recovery characteristics of the proton NMR signal. Signal processing lies at the heart of MRI and data acquisition and manipulation occur naturally in the Fourier transform domain of the image. Hence, the images are easily reconstructed in the Fourier domain. A Matlab^{tm} code to reconstruct MRI images using the properties of the Fast Fourier transform is attached as Appendix C. The original data spectrum (fspec) in the Fourier domain is reconstructed using a 2 dimensional fast Fourier Matlab function (FFT) by taking the FFT of the rows of the image followed by taking the FFT of the columns. The raw MRI data as acquired by the G.E. Signa 5.0 scanner in the frequency domain is shown in Figures 14 and 15. Figure 14 shows a three dimensional view of the two dimensional image data in the frequency domain. Figure 15 shows a two dimensional view of the frequency data after taking the log of the data. Figure 16 shows the reconstructed MRI data in the space domain using the properties of the Fourier transform.

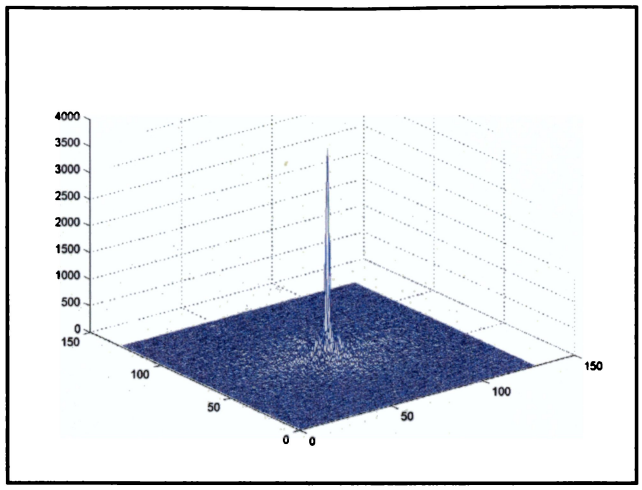


Figure 14: 3-D View of Unreconstructed MRI Data

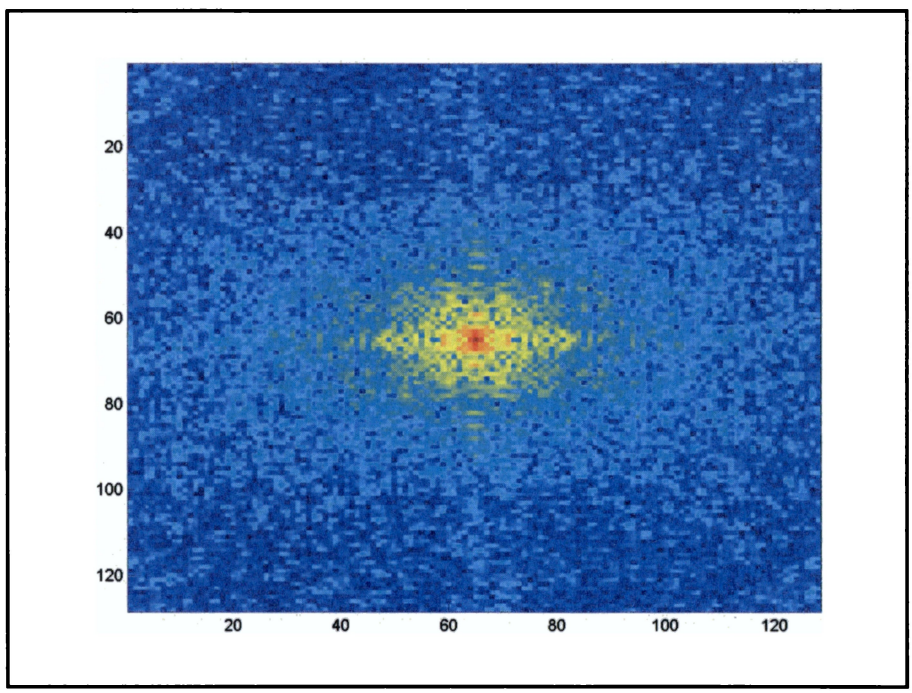


Figure 15: 2-D View of Unreconstructed MRI Data

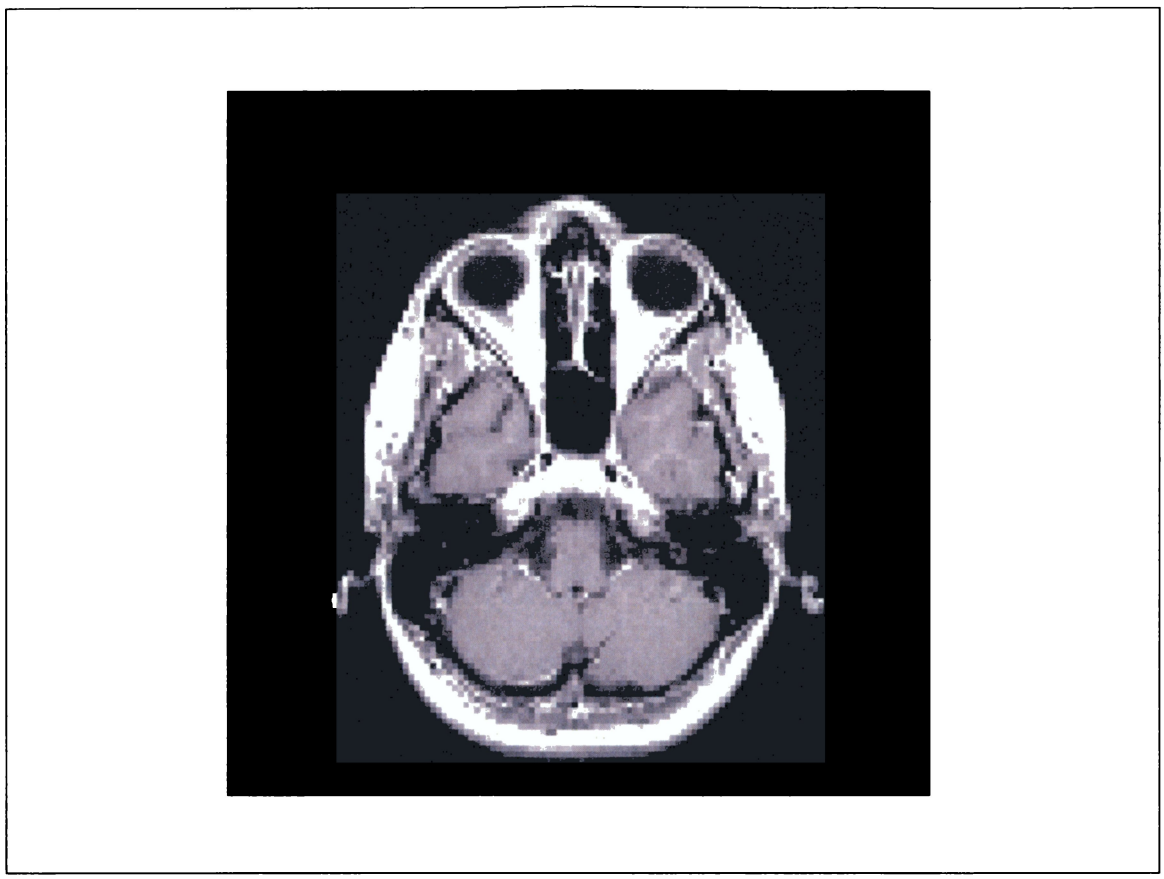


Figure 16: MRI Image Reconstruction Using Fast Fourier Transform (FFT)

As can be seen from the reconstructed MRI data, there is good discrimination between hydrogen dense areas (skull, eyes, fat), to less dense areas (brain, etc.). Also, note that several anatomical structures such as the brain, skull, eyes, etc. are readily apparent in MRI images and the resolution of the MRI data as retrieved from UTMCK is on the order of .8953 mm X .8953 mm in the X and Y dimensions and 7.5 mm in the Z dimension. The structures that become visible on an MRI image depend on the decay characteristics of the proton. The local environment of the protons can greatly influence the decay properties of the photon. The differences in the decay properties of the proton can be taken advantage of by selecting T1 or T2 weighted image. The physics of MRI images and the use of T1 or T2 signals to image

different anatomical structures is described elsewhere [26]. T1 weighted MRI images have a short repetition time (TR) and a short echo time (TE) [26]. This makes them ideal for visualizing the anatomical details needed for treatment planning. It is also ideal for imaging tumor, muscles, infections, cortical bone, ligaments, tendons, and other anatomic subtleties. Thus, T1 weighted MRI images are suitable for treatment planning purposes to delineate the tumor and target regions. T2 weighted MRI images have a long repetition time (TR) and a long echo time (TE) [26]. Thus, this makes them useful in distinguishing normal from abnormal soft tissue. T2-weighted MRI images are suitable for studies looking at inflammation and presence of fluids in the brain. The T2-weighted MRI images are not used in treatment planning or in image registration within this work.

2.2 PET

Others have recently noted considerable benefits in using PET images for revealing cancer metastases in cervical cancer patients, breast cancer patients, as well as in lung cancer patients undergoing conventional treatments and therapies [27,28,29]. In these recent studies, PET was shown to significantly alter the conventional treatment protocol and in some cases even the amount of the dose prescribed and the location at which it was delivered. Several others have also addressed the importance of using PET images in treatment planning [4,20, 23,24,30]. An understanding of the nature of how PET images are reconstructed is necessary to understand how they can potentially improve treatment planning. Essentially, a PET image is an *in-vivo* look at the biodistribution of a radiolabeled tracer in which the pixel intensity of the PET image is directly proportional to the concentration of the tracer present within that pixel.

For BNCT purposes, a tracer, such as *para*-Boronphenylalanine (BPA) is radiolabeled with isotopes like F-18 (most common), C-11, N-13, and O-15 that all have unpaired protons and decay by an emission of a short-lived positron. Once emitted, the positron has a limited range (on the order of a millimeter or so) before annihilating with an electron to produce two photons emerging from the point of annihilation at approximately 180 degrees apart, in the laboratory. Coincidence detectors along the line of response can then detect the emitted photons. These lines of response can be rotated about the object to view the resulting photons from various projection angles. The integral sum of all the lines of responses for a particular location on the image is directly proportional to the amount of radiotracer present within that particular point of interest on the image. The resulting images which result from tallying up the lines of response for various projection angles, θ, and various increments, s, between the lines of response as the detector gantry rotates about theta results in what is commonly known as a sinogram image. The sinogram image can be backprojected to represent the actual image in the x,y coordinate system (as opposed to the (s,θ) coordinate system which represents the sinogram data). Various algorithms have been proposed to reconstruct the images from these sinogram images. The most common algorithm currently used is the filtered backprojection (FBP) algorithm [31]. Other algorithms include the ART (Algebraic Reconstruction Algorithm) and the EM-MLE (Expectation Maximization/Maximum Likelihood Estimate). The resulting sinogram data in the space domain (as shown in Figure 19) can be reconstructed using one of the various algorithms like EM-MLE (as shown in Figure 20) or the more common, like FBP (as shown in Figure 21) in which the data is back-projected to reconstruct the image.

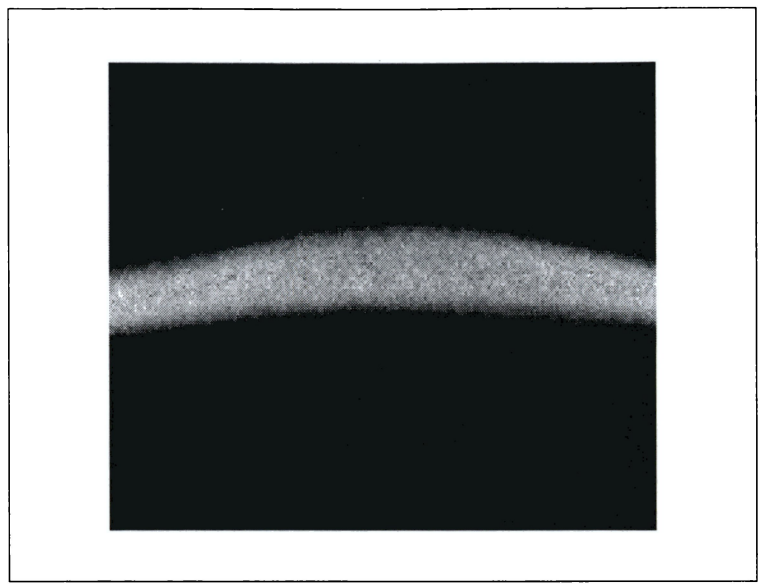


Figure 19: PET Sinogram Data

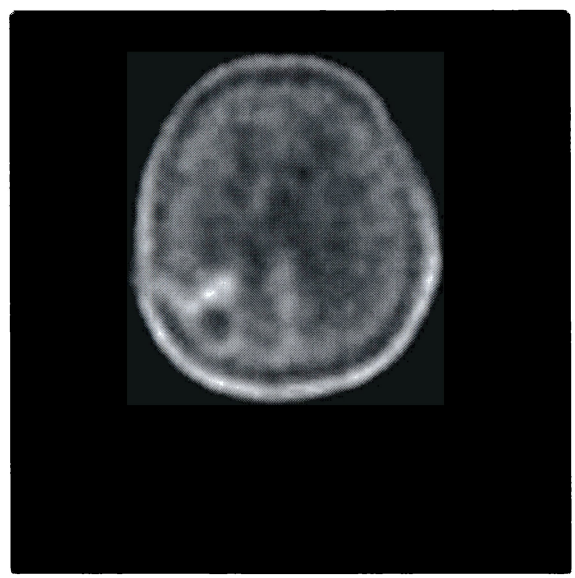


Figure 20: EM/MLE Reconstruction (implemented using Gregor's algorithm [32].)

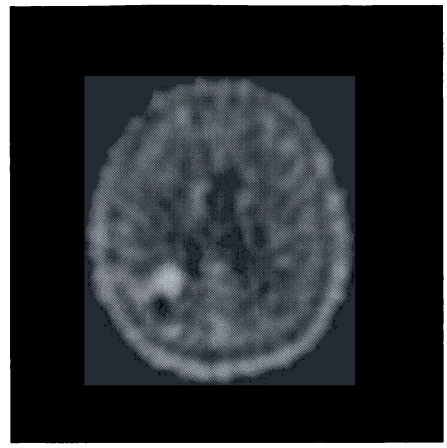


Figure 21: Reconstructed PET Image Using FBP

Generally, the PET image reconstruction process involves normalization, back projection, decay correction, scatter correction, and filtering to produce the standard image. The filtered back projection method, as derived by Radon in 1917, can be described as:

$$f_{bp}(x, y) = \int_0^{\pi} m(x \cos\theta, y \sin\theta) d\theta \qquad (5)$$

where

f_{bp} is the filtered back-projected image in the spatial domain,

x and y are the location of the annihilation event,

θ is the angle at which the detector is located along the line of response,

and m is the mapping function that maps the annihilation event back to the location on the image.

After reconstructing a PET image, the pixel intensity of the image is proportional to the distribution of the tracer within the volume. Distribution of ^{18}F-BPA as is visualized from a PET image for one of the University of Tennessee Medical Center Knoxville (UTMCK)

BNCT patients is shown in Figure 22. Others in the field have used 18F-BPA to determine the boron biodistribution within the brain using the FBP reconstructed images [33].

As is evident from the PET pixel intensity plot in Figure 22, the boron distribution within any volume of interest is not homogeneously distributed, as is assumed in current treatment planning within SERA. This is especially true within the tumor volume and the target volumes where the tumor tends to have a larger concentration of boron within the main mass of the tumor and lesser concentration further out on the peripheral boundaries of the tumor (i.e. the target region). Thus, erroneous results may be obtained by assuming a homogeneous distribution within the entire tumor and target volumes during the scaling portion of treatment planning in SERA.

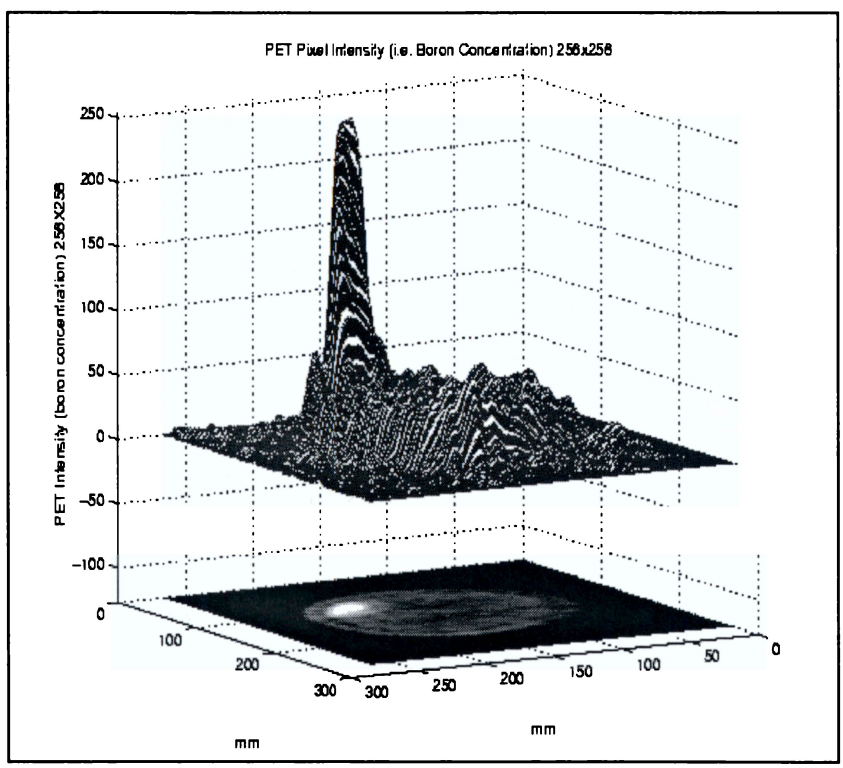


Figure 22: PET Pixel Intensity for a ^{18}F-BPA Reconstructed PET Image

Better treatment planning would allow for registered MRI/PET images to be used to determine the dose and to incorporate the boron biodistribution within the treatment planning. This is especially true since the dose is a function of the product of the neutron fluence at a particular location and the concentration of the boron within that volume. In other words, the reaction rate of the boron with neutrons is as follows:

$$RR = \int \phi(r, E, \Omega) * R(E) dE$$

where,

R(E) is the boron absorption response function at some volume element

ϕ is the flux at a particular volume element,

r, E, and Ω are quantities described in an earlier section.

Thus, the reaction rate of the boron with the neutrons depends on the spatial distribution of the boron within a given volume. Thus if one, is able to get a better biodistribution of this boron with in-vivo ^{18}F-BPA PET images, one is likely to get a better estimate of the dose being delivered within the volume.

Even though PET provides metabolic information (and hence the boron concentration information), it lacks significant anatomical information for precise definition of certain anatomical structure for volume reconstruction. The boron concentration (# Boron Atoms/pixel) can be computed as follows:

$$\frac{[x \ \ ECAT \ \ cps] * [20.5 \ uCi/ml] \ \ [3.7x10^4 \ di\sin tegrations/\sec ond] \ [1s] \ \#ml}{[ECAT \ pixel] \ * \ [uCi] \ \ [1.053*10^{-4}]} = \frac{\#\ Boron\ Atoms}{pixel}$$

where,

x ECAT cps are the ECAT counts per second determined directly from the 16 bit PET pixel intensity multiplied by the Quantity Scale factor,

20.5 uCi/ml/ECAT cps/ECAT pixel is the conversion factor that will convert the PET pixel intensity in ECAT cps into uci/ml,

$3.7 \text{ X } 10^4$ is a factor that will convert uCi into disintegrations/second,

ml is the total tracer volume (10 ml) that is administered to the patient prior to scanning,

and $1.053*10^{-4}$ is the decay constant of ^{18}F.

Since it is known that ^{18}F-BPA does not associate *in-vivo*, each annihilation corresponds to 1 atom of boron within ^{18}F-BPA or the soluble form it, ^{18}F-BPA-Fr. One can also take the #B atoms/pixel as determined in the above equation and convert that directly into a more usable form (in ppm/pixel) as follows:

$$\frac{[\#\, B\, atoms]}{[pixel]} * \frac{[1mole]*10.8g/mole}{6.022*10^{23}\, atoms} * \frac{1*10^6\, \mu g}{g} = \frac{ppm\ ^{10}B}{pixel}$$

where 10.8 g/mole is the molecular weight of boron

Thus, use of PET within treatment planning should be coupled with MRI or CT image registration for better treatment planning since it can allow for direct measurement of the boron concentration *in-vivo*.

A Matlab program has been developed to reconstruct PET images using the convolution backprojection method (Appendix D). Shown in Figure 23 is an original phantom image of the thorax, the radon transform of it and the associated sinogram data, the ram-shak reconstruction filter used to remove 1/r blurring of the image, and the reconstructed image using the convolution backprojection algorithm. The phantom is a generic phantom

built into Matlab[tm]. A discussion of the phantom can be found in the references [34,35]. Shown in Figure 24 is the same image reconstructed using the radon inverse function within Matlab[tm].

Dynamic PET images consist of several image frames spaced out over about 2 hours. Each image frame consists of 47 image planes with 128X128 pixels per image for the CTI ECAT 921 scanner. The image planes are axial planes starting from the top of the head to the bottom of the head with an axial separation of about 3.5 mm between the planes. Each frame has a period during which the counters count the number of photons being emitted. This can range from 10 seconds to 15 minutes as specified by the user. Since the entire dynamic PET scan collects counts over a period of 2 hours, the images can be time weighted and integrated to get a better anatomical view of the brain than is currently available using just a single frame.

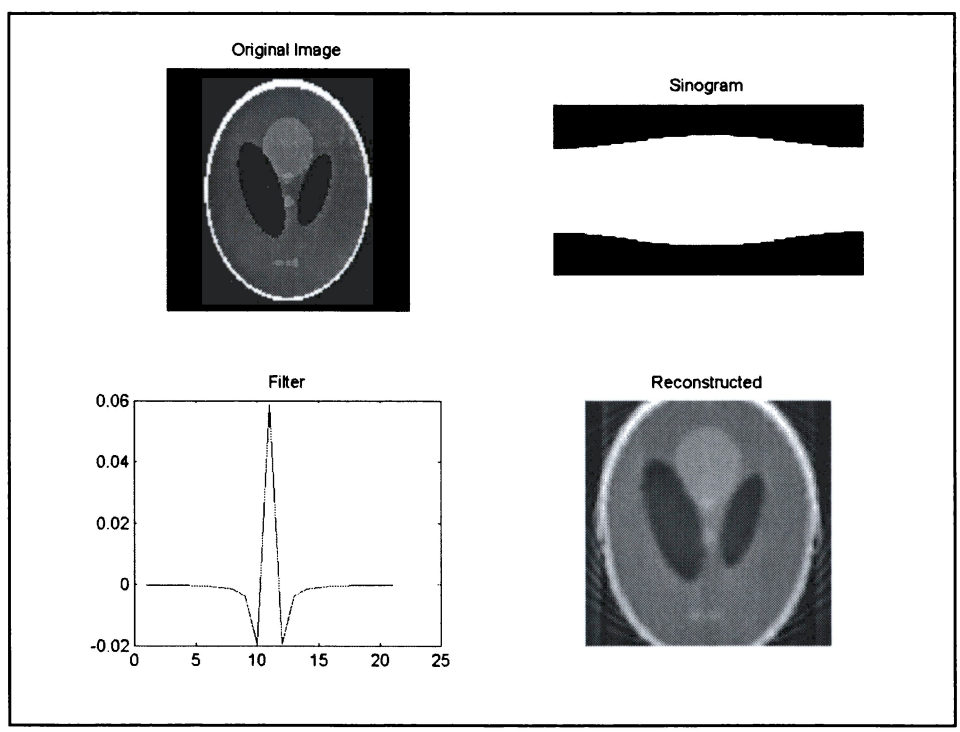


Figure 23: Filtered Backprojection Reconstruction Using Convolution Method

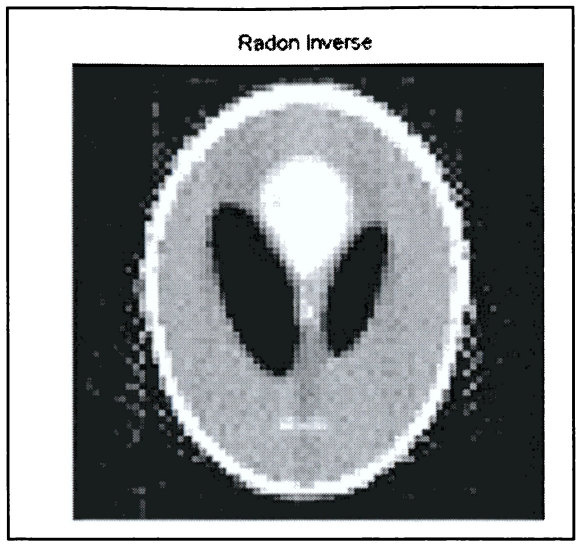


Figure 24: Filtered Backprojection Reconstruction Using Inverse Radon

A Matlab[tm] program has been written which time weights the data and integrates each image to maximize the number of counts corresponding to a given pixel (Appendix E and F). This can be accomplished as follows:

```
For k=1:number of frames,
 For i = 1:number of planes,
     Image(i,k) = Image(i,k)*QS(k)*frametime(k)/sum(frametime) + Image(i,k-1)
End
End
```

where,

k is the counter over the total number of time frames.

i is the counter over the total number of axial planes.

image is the individual 128X128 axial image, i, and at some time frame, k

QS is the quantity scale factor that converts the 16 bit PET pixel intensities for a given frame into ECAT counts/second.

Frametime(k) is total count time for a particular image frame.

This results in PET images that are anatomically and statistically more accurate than the individual images. Shown in Figure 25 is a side by side comparison of an image in a particular frame and the time integrated image. Visually, the time integrated PET image is more descriptive in that the midline fissure of the brain, the scalp, the brain, and other regions within the PET image are better defined. In addition, streaks and noise are further reduced in the time integrated PET data. Shown in Figure 26 is a three dimensional view of the PET pixel intensities along with a comparison of the normalized difference between the two images. One can note that in regions around the tumor, there is increased uptake in the time integrated data in some cases. This may indicate better visualization of the tumor regions in the time integrated data as opposed to the individual frame data. This can perhaps be attributed to increased signal-to-noise ratio by summing up the total accounts in the tumor and target regions.

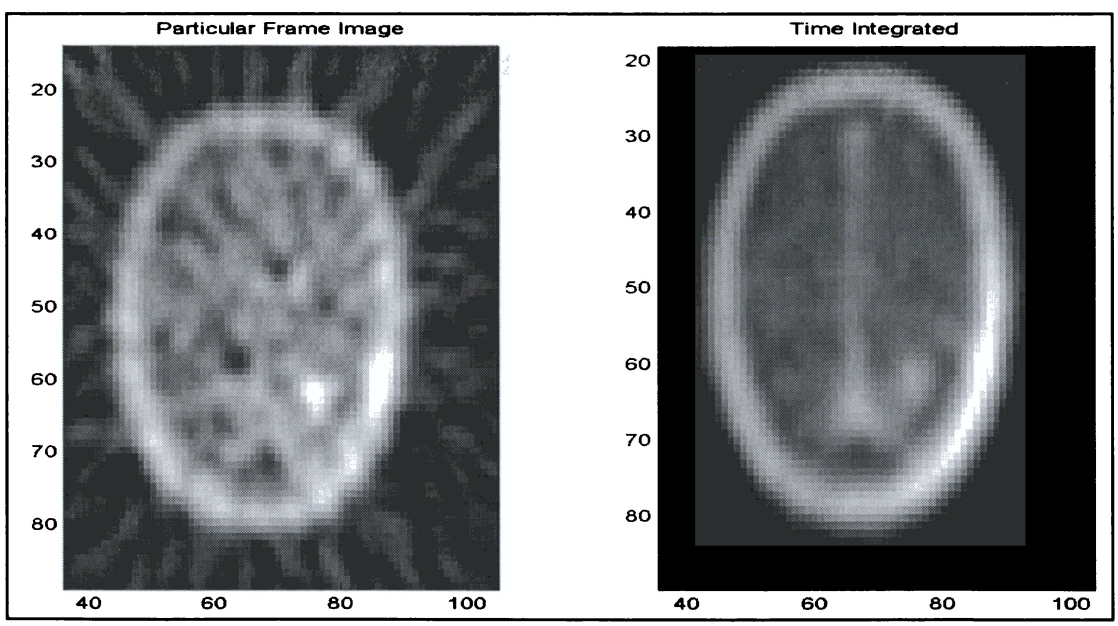


Figure 25: Side by Side Comparison of Individual Frame Image Vs. Time Integrated Image

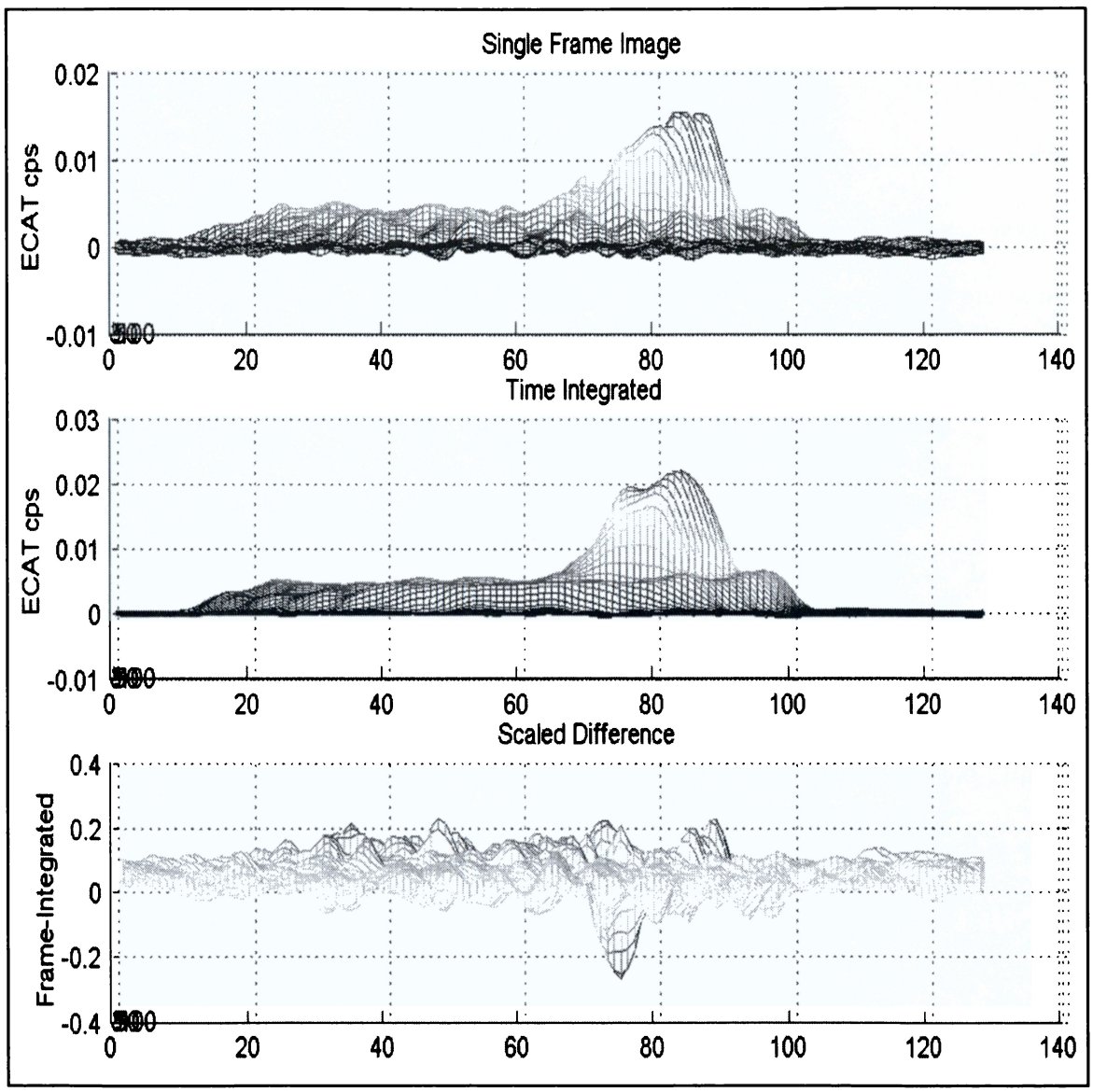


Figure 26: 3D View of Individual Frame Data, Time Weighted/Integrated Data, and the Scaled Difference

III. Simulated Environment for Radiotherapy Applications (SERA)

SERA is a high performance three-dimensional planning system for neutron capture therapy treatment plans developed jointly by Idaho National Engineering Environment Laboratory (INEEL) and the Montana State University (MSU) computer science department. It has been tested and validated at Brookhaven National Laboratory (BNL) [36,37]. SERA allows for patient geometry to be reconstructed based on highly efficient uniform volume elements (univels) in which the model resolution is only limited by the image resolution. In typical use, SERA consists of three main components: manual and semi-automated geometric modeling of treatment objects derived from PET, MRI, or other medical imaging modalities, dose computation for these geometries with **SeraMC** (the INEEL radiation transport code), and dose contouring that overlays dose contours derived from the dose computation onto the original image planes to determine healthy tissue limits. A license for SERA was obtained from INEEL with the help of Dr. George W. Kabalka, Chuck Wimperly, and David Niggs.

The author has installed a current version of SERA on the nuclear engineering workstation on **Felix.engr.utk.edu** under the **/ne5/codes/sera** directory. To execute SERA, a path to **/ne5/codes/sera/sera1/Target/bin** must be added to the .cshrc file. In addition, one must copy the .serarc file within the main directory and this file must be sourced in the .cshrc file. The author has also kept SERA updated with the latest patches. Latest patches of SERA can be downloaded from http://www.cs.montana.edu/~bnct/ and following the link under updates. An appropriate login name and a password should be obtained from INEEL to keep SERA updated.

The first part of the task, manual and semi-automated geometric modeling of the treatment objects, are implemented using the **seraImage** and **seraModel** modules of SERA. These modules can be launched independently by issuing the **seraImage** or **seraModel** commands at the prompt, or by selecting the appropriate icons from the SERA Graphical User Interface (GUI). Sera GUI can be launched by issuing the **seraMenu** command at the prompt.

The solid models generated by **seraImage** and **seraModel** are comprised of volumetric automatons and are referred to as univels. It is the combination of these univels that define particular volumes and regions of interest. Because of the uniformity of these univels, geometric interrogations can be performed using a very efficient integer arithmetic algorithm and the resolution of the geometry is limited only by the imaging modality. **Sera3d** allows for the viewing of the univel geometry as well as particle tracks, beam line orientation, and dose contours in projected three-dimensional space. It is **seraCalc/seraMC** modules that are independently developed at INEEL and are used to calculate the complex radiation fields present in BNCT treatment. The **SeraDose** module can then be used to visualize image space simultaneously with overlapping dose contours in two-dimensional space.

Other modules in SERA include **seraPlan** and **seraPlot** which can be used to develop a treatment plan or to combine various beam fractionations and to visualize dose volume histograms and dose depth profile for a given treatment session, respectively. Discussed in this section are only a brief overview of these modules and the main technical points on how treatment planning is accomplished using SERA with some end objective in mind. Detailed information and a user's manual can be found on line [39] or in an INEEL in-house class document [38]. Before using SERA, one should have an endpoint in mind. SERA can be used

to accomplish a wide variety of tasks such as allowing for the display of dose volume histograms, dose depth profiles, two-dimensional and three dimensional visualization of the beam setup and various components of the total dose along with patient geometry. It can also be used to combine multiple fractionations from the same beam or from different beams, and for generation of various output files such as ***.contour*** or ***.cdf.sz*** which contain individual contour dose data overlaying particular image planes. For purposes of dose adjustment, the endpoint objectives were to demonstrate how current treatment planning is accomplished using a single beam setup in SERA, to display the various components of the dose, and to demonstrate how the homogeneous scaling of the dose is incorporated within treatment planning in SERA. In addition, another objective of using SERA was to obtain certain files that could later be utilized within MATLAB^{tm} subroutines to post-process the dose and to improve existing treatment planning by incorportating the PET derived boron bio-distribution. Thus, with these end objectives in mind, the author only discusses those parts of SERA in the next few sections that are important to reach these objectives. In a later chapter, there is also a description of how the MATLAB subroutines use the files generated by SERA and couple them with the registered PET and MRI images to better predict the average dose to the tumor and target volumes.

3.1 CONVERSION OF IMAGES INTO THE APPROPRIATE FORMAT

Before one can begin treatment planning in SERA, MRI or PET images must be medically acquired and reconstructed. Physicist's access to these images must be approved by mutual consent of the patient and the physician. The images must then be converted into an appropriate format suitable for SERA. MRI images are structurally and anatomically different

from PET and CT images. There are certain advantages and disadvantages to using MRI, CT, or PET images within treatment planning. MRI images are hydrogen density maps and provide for detailed anatomical description of patient geometry. Contrast enhanced T1-weighted MRI images can provide soft tissue contrast in which the edema region and the tumor region may be readily visible. CT images are tissue density map, and likewise they also provide detailed anatomical description of patient geometry. However, MRI and CT images lack the much needed physiological and biodistribution information that can be provided by PET.

Because MRI, CT, and PET images are generally retrieved and reconstructed using properitary software, image format conversion is always needed before one can directly use these images within treatment planning in SERA. Conversion of these images is no trivial matter given the proprietary nature of the images. SERA only accepts 256X256 8 bit or 16 bit row major data or QSH format images. The QSH format, developed by Maguire and Noz, provides for a common image format and incorporate data generated by image-processing into a picture archiving and communication system (PACS) [40-42]. This format is an adaptation of the American Association of Phycisists in Medicine (AAPM) standard image format and consists of an ASCII header and a separate image file. The header may contain any of the following information: date of image creation, imaging modality, field of view, bytes per pixel, and other parameters needed to convert the N-dimensional array of pixel values to the visual representation for interpretation. The QSH format will generally include 2 files. The .qhd is the header file and the .qsh file is a concatenated file containing the bitmap pixel intensities of all the image planes.

MRI and PET images obtained from the University of Tennessee Medical Center, Knoxville (UTMCK) generally exist in the General Electric (GE) SIGNA 5.0 format or the CTI ECAT (6.4 or 7.0) formats, respectively. MRI images from UTMCK are generally archived as several series under a main study. There may be 4-5 series per MRI study. Each series of the study will correspond with T1, T2, pulse gated, combined T1+T2, contrast enhanced, or a combination of these MRI techniques. Which technique is most appropriate depends on the radiologist's preference and the object of interest. Each series generally consists of about 20 image planes with 150 mm axial field of view ana an interspacing of 7.5 mm between each image plane with a resolution of about .8953 mm in the x and y directions. Most commonly, the MRI image planes tend to be 256X256 16-bit integer format with 7904 bits of header information. There is also a main header associated with a particular study, a subheader associated with each series, and an imbedded header of 7904 bits of information within each image plane. The header information can be used to understand the setup of the scanner used to retrieve the particular image for a particular patient.

The GE SIGNA 5.0 MRI data and the associated header file must be converted into raw data or the QSH format. There are at least three options to convert proprietary MRI data into SERA suitable raw format. One is to use an IDL (Interactive Data Language by CREASO) software, **GEVIEW**, which is licensed to run only at the medical center. I do not recommend this option since the license for **GEVIEW** is expensive and constantly in use by radiologists at the medical center. **GEVIEW** allows for visualizing of the data as well as provides the header information such as pixel size and interplane spacing needed for treatment planning.

The second option is to use the **MRI2RAW.m** program that the author developed in Matlab[tm] in tandem with the **MRICro** shareware developed by Chris Rorden. **MRI2RAW.m** is included in the appendix (Appendix G and H) and the **MRICro** software to visualize MRI data can be downloaded from the internet as shareware [43]. Both programs, **MRI2RAW.m** and **MRIcro** must be used in tandem to convert the MRI data into the format needed for SERA. **MRICro** software allows one to visualize MRI image data as obtained from UTMCK and to convert the MRI data into the **Analyze** format. Although, this allows one to look at the pixel sizes in the x, y, and z dimensions and to read the proprietary header information, the draw back of this program is that it does not allow one to convert the individual image planes into raw 8 bit 256X256 images suitable for SERA. Thus, one must resort to alternative means.

To resolve the problem of not having easy access to **GEVIEW,** and the fact that **MRICro** only allows one to convert the data into the **Analyze** format, I have developed a subroutine that guesses at the SIGNA image size and the image format and strips out all of the header information. Knowing that the image size is 138976 bits and with some trial and error, it was determined that the SIGNA 5.0 MRI images existed in 256X256 16 bit integer format with 3952 cryptic 16 bit header lines. The author used this information to remove all unnecessary header information while keeping only the pixel intensity data. The pixel intensity data is then rewritten into a raw file. Since the header file information is in a proprietary GE SIGNA 5.0 format, the header file could not be easily parsed. Thus, one can only get the pixel intensity information from **MRI2RAW.m**, but no header information

regarding the x,y, and z pixel size is available. Thus, to get the pixel size information, one should use **MRICro** in tandem with **MRI2RAW.m**

To use **MRI2RAW.m** to convert UTMCK MRI images into raw 8 bit 256X256 format, one must supply an input file (**MRI2RAW.in**) which lists the location of the directory on the seventh line of the input. Directly below this line, the **MRI2RAW.m** expects a listing of individual image planes that one would like to convert. The original **MRI2RAW.m** Matlab[tm] and the **MRI2RAW.in** input file is included in appendices G and H. The program takes a SIGNA 5.0 MRI image, strips out all the header information and takes the 256X256 16 bit pixel intensities and converts them into raw 8 bit 256X256 row major image, which is suitable for **SeraImage**. All of the header information is discarded. Thus, one should note the x,y, and z pixel sizes as shown in **MRIcro** before using **MRI2RAW.m** to convert the individual image planes into raw 8 bit 256X256 row major image data. An overview of **MRI2RAW.m** is shown in Figure 27. The x, y, and z pixel sizes are later used in SERA to relate the image objects with the patient geometry.

In addition to **MRI2RAW.m** which converts individual MRI image planes into 256X256 raw 8 bit image format without providing the x,y, and z, pixel sizes, the author has also developed another Matlab[tm] subroutine, **Read_registration.m**, which reads the MRICro converted **Analyze** MRI images and converts them into raw images suitable for SERA. **Read_registration.m** also displays registered PET images in the **Analyze** format as contours

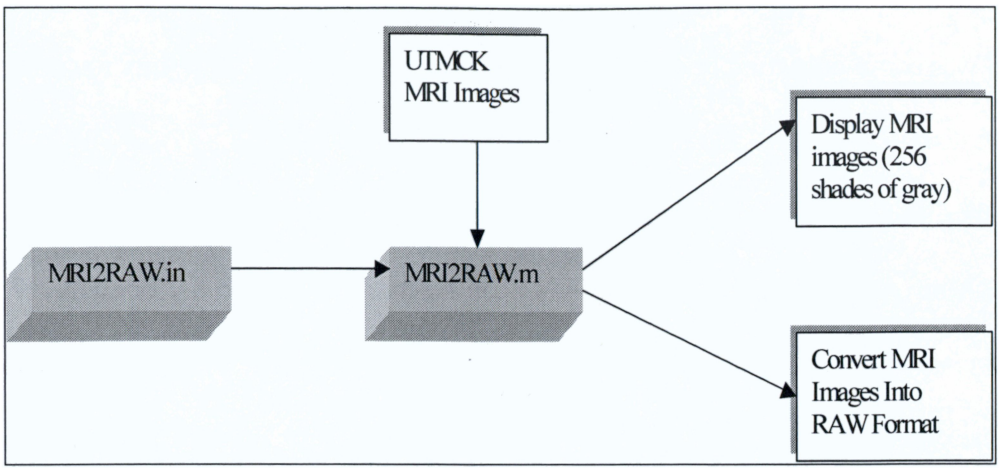


Figure 27: Overview of MRI2RAW.m Subroutine to Convert MR Images Into SERA Compatible Format

lying on top of the original MRI image. The program is attached in Appendix I. Of the three approaches, the author prefers the later approach of using the **Analyze** format images from **MRICro** and using **Read_registration.m** algorithm to convert the images. Read_registration.m uses the spm_hread.m function of the Statistical Parametric Program (discussed in a later section). The only drawback to **Read_registration.m** is that it must also be supplied registered PET images in the **Analyze** format for this to convert MRI images into SERA compatible format.

PET scans at UTMCK prior to June 1998 were reconstructed using the CTI ECAT 6.4 software. PET scans after June 1998 were reconstructed using the CTI ECAT 7.0 software. As such, separate programs have been developed to convert these images into appropriate format since some patients were imaged prior to June 1998 while others were imaged after June 1998. Different methods for converting PET images have been explored by the author over the course of this work. One method to convert PET images involves the use of **zoom_img** software developed by Hedley Bond to strip out one of the frames of a PET study and then

split the frame out into individual planes [44]. This software only works with ECAT 6.4 data. This method was commonly used in earlier work with PET data. Use of **zoom_img** to convert PET images into raw 8 bit 256X256 format is described elsewhere [45].

Another method is to use the Matlab[tm] subroutines developed by the author, **ECAT2SERA.m** (Appendix J). The image data can be in either ECAT 6.4 or ECAT 7.0 format. There are only two inputs needed to convert the images. One must specify the name of the file that one wants to convert and if the file is in ECAT6.4 or ECAT7.0 data format. The program uses **ecatfile.m** function (Appendix F) to read the header information and the data from the file and converts all the planes from the last frame of the PET. **Ecatfile.m** is a program developed by Flemming Hermansen for the purpose of allowing users to interface with ECAT6.4 or ECAT7.0 PET image data. Both of the programs, **ECAT2SERA.m** and **ecatfile.m** are attached in the appendix section (Appendix J and F, respectively). For the purpose of converting PET images for SERA, I recommend **ECAT2SERA.m** option over the **zoom_img** option.

3.1 SERAIMAGE

Basic concepts used by SERA include slices, bodies, and body slices. Shown in Figure 28 are slices that correspond to medical imaging data (i.e. reconstructed CT/MRI/or PET images). These images contain several anatomical regions of interest over various planes. These anatomical regions of interest are sometimes referred to as bodies (like skull, brain, scalp, etc.). The body slices are intersections of these bodies with medical slices. These

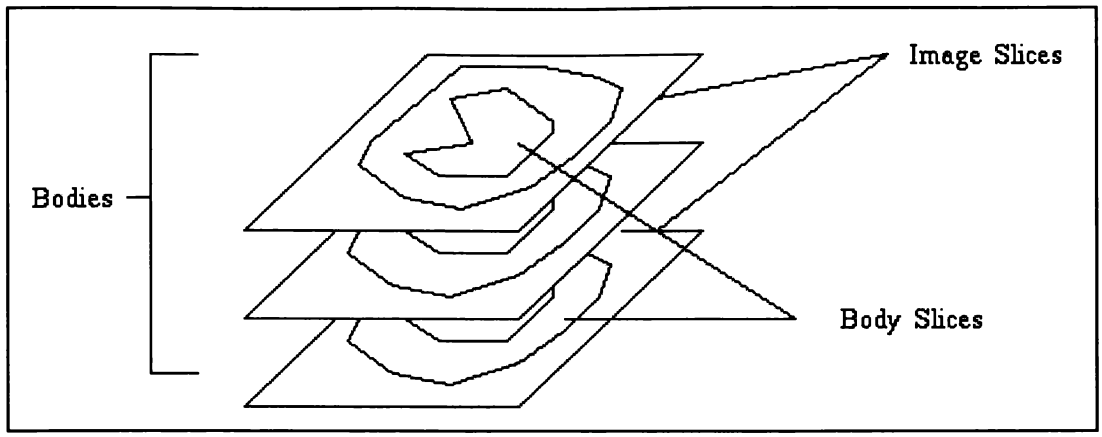


Figure 28: Bodies, Image Slices, and Body Slices

bodies must be defined by picking points, drawing, filling, or painting a body on the slice. These body slices can then be combined to form a three-dimensional volume.

The loading of image slices is accomplished by the use of **seraImage.** Most treatment plans in SERA begin with this module. **SeraImage** is essentially used to convert the oiginal image format into the QSH format, which is the standard image format used by SERA. Currently, **seraImage** only accepts individual image planes with or without headers and takes a guess at what the image structure may be, depending on the size of the image. It tries to guess at the image size, bytes per pixel, and header size. It also allows a user to change these values. This section of **seraImage** is not perfect and in many cases, separate programs must be written so that images can be converted into a suiTable format for **seraImage**. Some of these programs have been described in an earlier section. The image format must be known to SERA. It is recommended that SERA accept 8 bit or 16 bit, 256X256 row-major image data, or QSH formatted data. One can expect to get strange results in sera3d (and other parts of SERA) if the images are not converted into a 256X256 8 or 16 bit format. One strange result

that I have seen included a misalignment of the image space with that of patient geometry within sera3d. The author has developed independent Matlab[tm] programs to convert MRI and PET data into a 256X256 RAW 8 or 16 bit format needed for **seraImage**, some of which have been described earlier.

Invoking **seraImage** launches a GUI in which one can open a QSH image set or load raw images. For raw images, one must also edit the header file to include x,y, and z pixel sizes and define the locations of each image plane along the z-axis. This is critical, or else volume rendering will be erroneous and the resulting doses will be inaccurate. Images can also be reversed, rotated, resliced, or (in limited cases) registered, using the advanced options of **seraImage**. After raw images are loaded and the appropriate header information supplied, the images can then be converted into the .qsh format. This will result in two files, an IMAGE.qsh and a IMAGE.qhd file, where IMAGE refers to a file name chosen by the user. The .qsh file contains the actual pixel intensity medical data, whereas the .qhd contains all of the relevant header information supplied by the user. Some of this header information may include patient name, image data type information (8 bit/16 bit), imaging modality, number of pixels, the physical sizes of the pixel, automatic z separation, z values corresponding to each image plane, and other information used by other modules of the program to relate image coordinate system with the patient geometry coordinate system.

3.2 SERAMODEL

One of the most time consuming parts of BNCT treatment planning is the patient modeling (or image editing). After the images are generated using seraImage, the next step in treatment planning is to define various anatomical bodies to create the model geometry.

SeraModel allows for many useful image operations, including manual and semi-automated (threshold) definition and generation of univel-based regions of interest, and various image processing tools to aid in the manual definition of these regions. Some of these tools include region copying, scaling, overwriting, painting by fill or borders, threshold segmenting, 3D region growing, and automatic target definition. Use of SERA model results in the generation of .uv and .uvh files that describe the geometry of the model as well as assigning the physical properties to the univels.

Invoking the **seraModel** module via the command prompt or the SERA GUI, yields the popup menu. To begin defining bodies, one should first load the QSH files (.qhd/.qim) files by selecting file, and open QSH image in the drop down menu. This will allow the user to load the QSH formatted medical images created using **seraImage**. The images will be displayed in columns and rows, as is shown in Figure 29. One can then use the EDIT regions selection of **seraModel** to create univel geometry by thresholding, picking points, painting, filling a region, copying a body, or by growing a region beyond another defined region. Using PET or MRI images, one should define the buffer region (the space beyond the scalp) in which neutrons will not be tracked. Other regions that can be defined using the various edit options of SERA include scalp, skull, brain, tumor, target, and others. Within the edit region, one can select the Edit Body/Materials options to relate material properties with those of geometric objects drawn on the images.

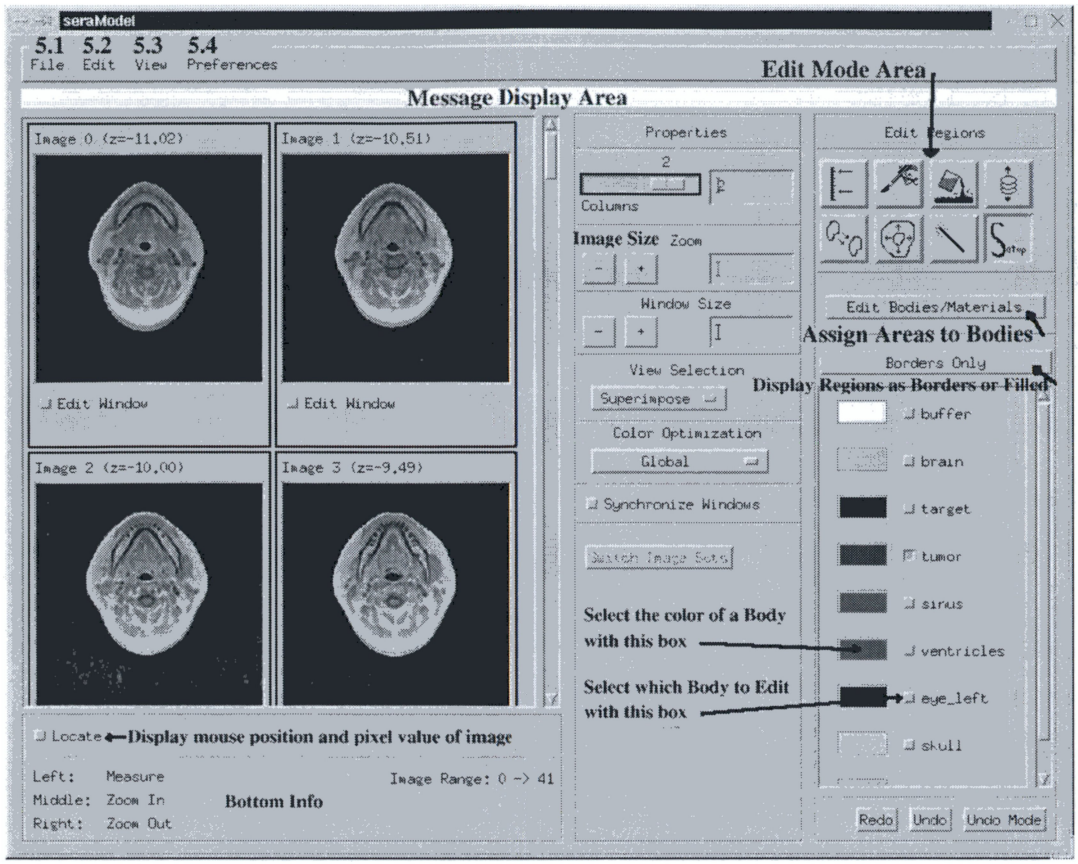


Figure 29: SeraModel GUI Used to Define Various Volumes of Interest

One can specify the material properties using the Edit Bodies/Materials selection in **seraModel**. For example, one can define the brain region to be based on the International Council on Radiation Units (ICRU) publication number 46 data on adult brain tissue. Other materials that one can define within SERA include void, buffer, Lucite, polyethylene with Lithium Hydroxide, water, ICRU skin, ICRU cranium, lung, concrete, MIT_brain, Fairchild tissue, the ICRU eye lens, and many others. For the brain, target, and tumor regions, the author recommends choosing the ICRU 46 brain with 10 ppm of boron placed through out the brain. For the skull region, the author recommends choosing the ICRU 46 skeleton with no boron, and ICRU 46 skin for the scalp region. The default material properties such the boron

concentration factor for various materials, and the relative biological effectiveness factor (RBE) associated with gammas, nitrogen-14 reaction, recoil protons, and ultrafast neutrons can be changed using the view/edit sub option within body materials option in **seraModel**. For the born concentration factor (i.e tissue-to-blood), the author chose 3.5, 3.5, and 1.0 for the tumor, target, and scalp regions, respectively. For the gamma RBE, recoil RBE, nitrogen RBE, hydrogen RBE, and the ultrafast RBE, the author chose 1.0, 1.0, 3.2, 3.2, and 1.0, respectively. The maximum constraint dose for the normal brain was set at 12.5 Gy-equivalent, and the maximum dose for the scalp and skull regions was set at 20 Gy-equivalent.

After completely defining all of the regions and defining the materials representing the regions, the save model file results in the generation of two files, a .uv and a .uvh file. The .uvh is a header file and the .uv file are the region definition file. Within the .uv file, one finds individual region definitions which overlay corresponding image planes. These regions are generally given a number, between 52-60, where each number represents a particular region. The value corresponding to a particular region on the image is included in the .uvh file.

Shown in Figure 30 is a 3-D mesh plot of the uniform volume elements (univel) region definitions generated by **seraModel**. These univel regions correspond to the non-uniform rational b-spline interpolation of the points chosen by the user. These univels provide the link between the medical image space, the geometry space of the calculation, the various bodies defined by the user, and the material properties associated with the particular univel volume. These region definitions can then be used within **PET2MRI** to relate a particular

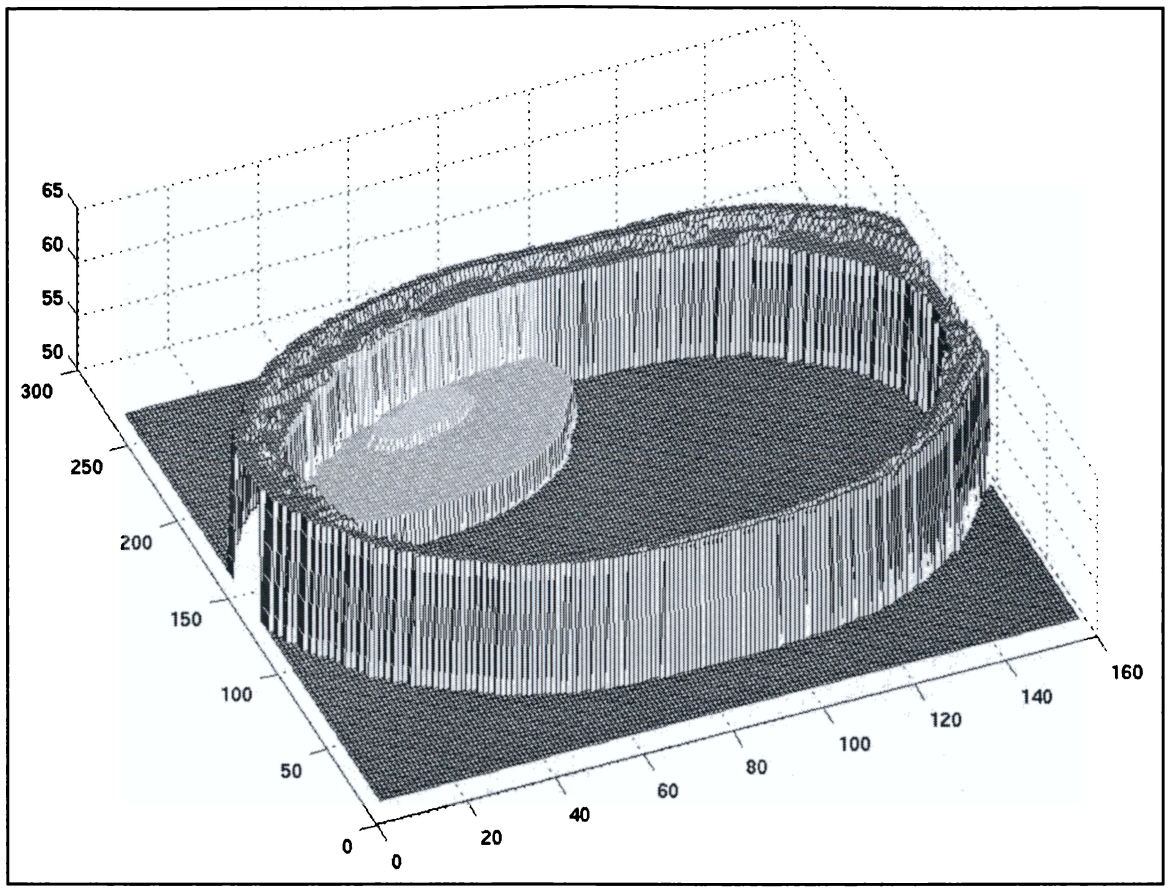


Figure 30: Region Definition Using Univels

regions with the material properties within that region. The **PET2MRI** algorithm is described in a later section.

3.3 SERA3D

Sera3d is used to provide a three-dimensional representation of the geometry and the dose data generated during treatment planning. Sera3d also allows for the QSH images generated by seraImage and the univel data generated by seraModel to be displayed simultaneously. In addition, it allows for display of particle tracks and beam alignment to be

displayed superimposed on the univel geometry and a particular image plane. This information is critical in visualizing the entire treatment process and helps to facilitate beam alignment. The beam alignment and the generation of the volume can be coupled with the dose data for better visualization of the dose in geometry space. An example of having the univel data and a particular image plane superimposed with particle tracks and beam alignment is shown in Figure 31. The three-dimensional geometry can be visualized in various ways. The geometry can be viewed as points, lines, solid volume elements, hollow volume elements, and image planes. A detail discussion of the various options within SERA3d is described elsewhere [38, 39].

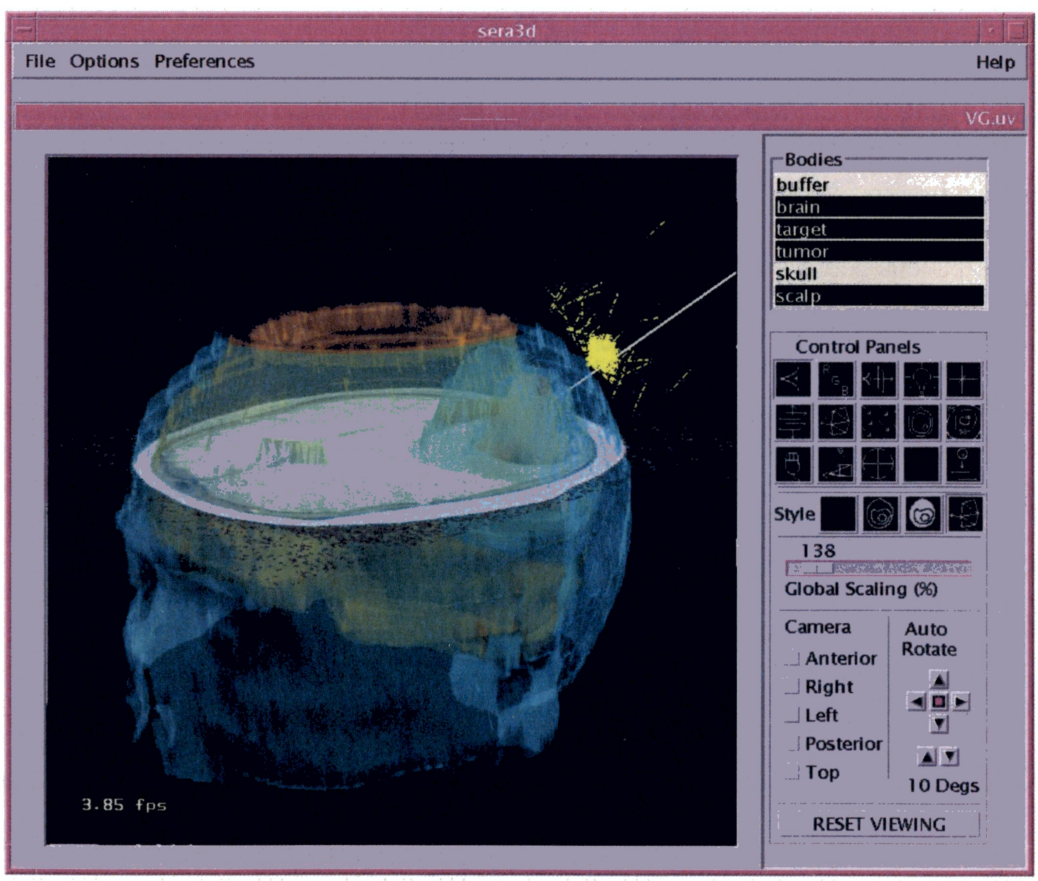


Figure 31: Patient Geometry Setup Within Sera3d

3.4 SERACALC

Once the QSH files (the .qim and .qsh) and the univel files (.uvh and .uv) files have been successfully created and the materials specified, one can launch the **seraCalc** module. This module couples the patient geometry with Monte Carlo simulation of particle interactions within the geometry. The user interface to **seraCalc** is shown in Figure 32. One can use the user interface, shown in Figure 32, to setup a treatment plan, or supply an input file, as shown in the Appendix section (Appendix K). Using the GUI, one must define an input file name, a combinatorial (**CG**) geometry file, a patient geometry file, and a new radiation shielding transport (RST) file. The input file name is the name of the file to which all of the user supplied information will be written out to. It is this file, which serves as the Monte Carlo input file. The combinatorial geometry (or CG) file is a file that describes in the analytic geometry and is associated with the patient geometry files.

The patient geometry file is the univel (.uv) file that contains the model and body information. Within this file are also RBE values and material properties which are associated with a particular body. The new RST file (.rst) is a file to which all of the detailed flux and dose information will be written to. In addition, other files must be specified. Some of these files include the source file, the material file, the cross section file, and the **seraMC** run script (**run_seraMC**). The BNL 12 cm aperture source file (BS.source) is placed in the **/ne1/mkhan/BNCT/Sera/ST/Results/** directory, the material file (SeraMC.m) is placed in the **$SERA_HOME/Resources/SeraMC/** directory, the cross section library (SeraMC.sigma) containing the neutron and gamma cross sections is in the **$SERA_HOME/Resources/ SeraMC/** directory. The run script used to execute the input file on the unix workstation is run_seraMC.

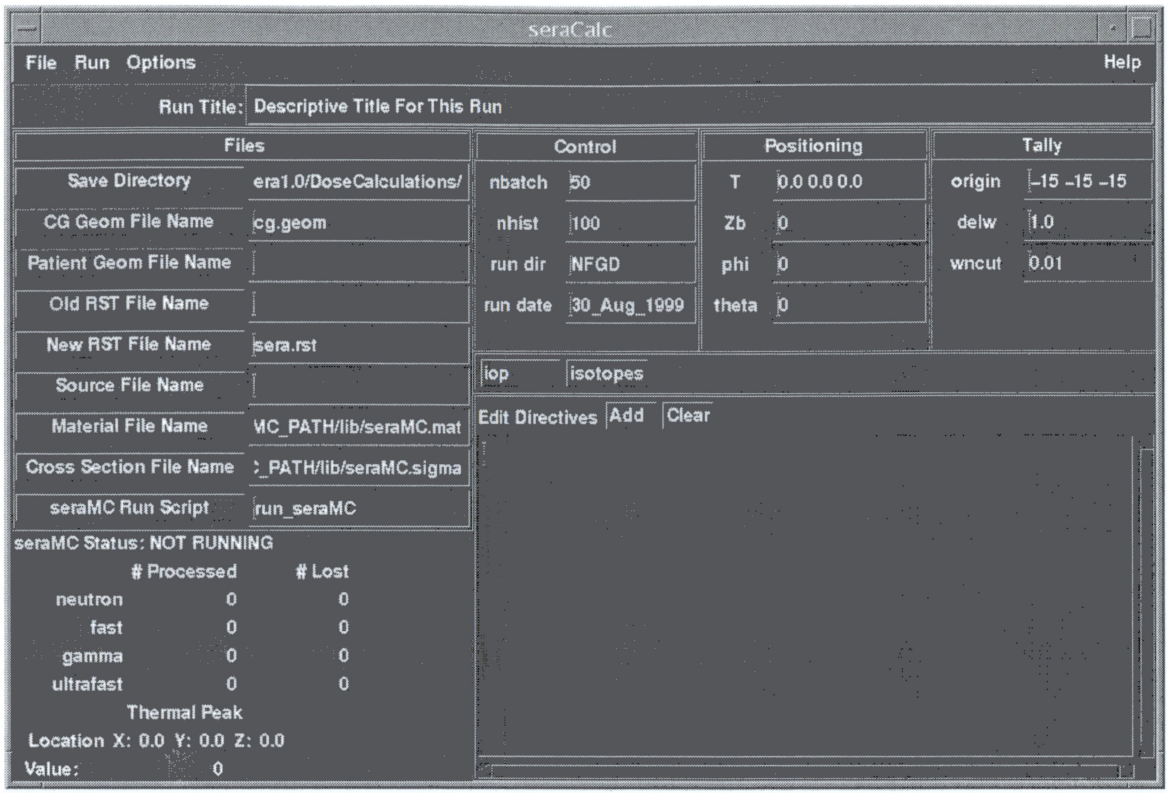


Figure 32: Sera Calc Graphical User Interface

After supplying the appropriate file information, **seraCalc** must be supplied information regarding the number of histories to simulate, the number of batches to run, and various run directives which specify particular particles to be tallied. The nbatch controls the number of batches, the nhist controls the number of histories to process, where as **NFGUPD** or some combination of this represents the run directives. **N** specifies neutrons with energy 0 to 16.9 MeV, **F** specifies source-biased neutrons with energy as specified for the first edit bin which are processed to determine the calculated value for the proton recoil dose, **G** specifies that beam and induced gammas, **U** specifies that neutrons with energies 0 to 100 MeV are processed (that is if you are doing fast Neutron Capture Therapy), **P** specifies that recoil

protons generated from neutron scatter with incident neutron energies > 16.9 MeV, and **D** specifies that an edit is performed. I recommend the use of **NFGD** run directives.

The position panel of the seraCalc GUI is used to align the beam. T represents the target point about which the source is rotated, Z_b represents the distance from source to target, ϕ represents the polar angle, and θ represents the azimuthal angle. The target point, the source to target separation, ϕ, and θ are used collectively to align the beam above the tumor. The target point is generally chosen to be the center of the tumor point. The target point can be changed slightly to allow for the dose contours to align up as such that maximum dose is delivered to the tumor region while sparing the normal surrounding tissue. Alignment of the beam can take some trial and error to align the beam optimally. Shown in Figure 33 is the target location (Xp, Yp, Zp), the source center, X_p, Y_p, Z_p+Z_b, and the beam/target point separation (Z_b). Shown in Figure 34 is a 90 polar (ϕ) rotation of the beam that moves the beam from the top of the head to the right side of the patient about the target point.

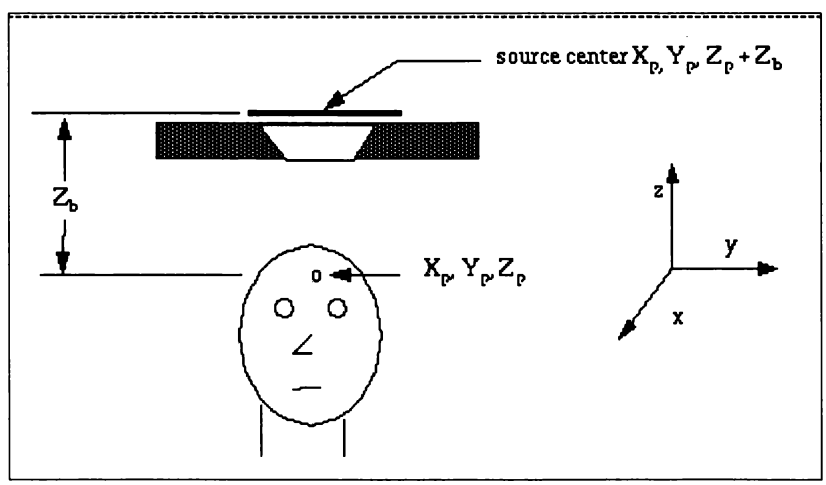


Figure 33: Target Location

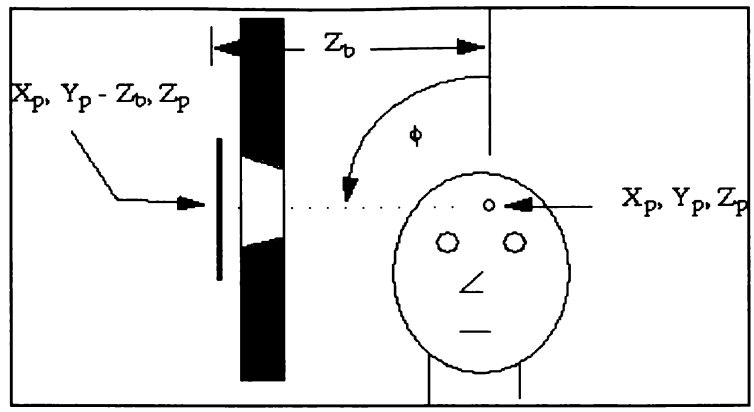


Figure 34: Polar Angle Beam Rotation From Top of the Head to the Right Side of Patient

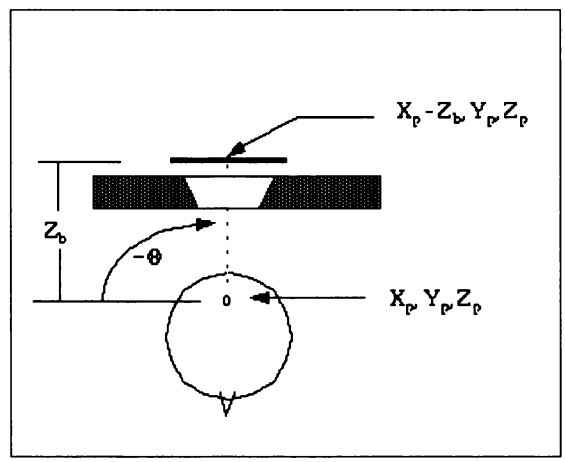


Figure 35: Azimuthal Rotation From Back of the Head to the Right Side of Patient

Shown in Figure 35 is the rotation from the right side of the head to the back of the head. The azimuthal angle, the polar angle, the target point, and the beam separation (Z_b) can be used to align the beam manually.

The tally panel includes the origin, delw, and the wncut options. The origin option sets the lower origin of the edit mesh. The wncut option sets the particle weight below which a binary decision is made to either throw the particle away or to double its weight. The delw

option controls the width of the edit mesh in centimeters. For example, if one defines the origin to be –15, -15, -15, then the mesh size of the geometry will be 30cmX30cmX30 cm. This is much larger than the geometry of the patient, thus mesh tallying will include all of patient geometry. A delw of 1 would mean that the mesh tallies should be conducted on 1cmX1cmX1cm voxel size and a wncut of .01 is the Russian roullete weight cut off in which if a particle weight falls below this value, then a random number is used to decided weather or not to follow this particle or to discard it. A random number below .5 will discard the particle and a random number equal to or greater than .5 will cause the weight of the particle to double and the particle will continue to be tracked. This simulation is continued for all particles.

In addition to the panels described earlier, the **IOP** panel option numbers 2, 3, and 4 are also useful. They allow for the beam to be automatically placed. **IOP** of 3 allows ϕ and θ to be determined automatically as such that the Z separation, Z_b, between the scalp and the source is defined by the user. The author used a z separation of .5 cm with the automatic beam alignment option number 3 in most of the treatment plans setup within this work. This was used because beam orientation was for a single-field irradiation, and the author wanted to ensure a consistent beam scalp separation between all patients. This was chosen so that difference in the beam placement above the scalp would not play a significant role in the results from patient to patient.

In addition to the inputs used by **seraCalc**, one can also specify edit directives that control the output of the calculation after all of the transport calculations are completed. The edit directives are critical if one is to post process the dose and to generate dose volume histograms. There are four edit directive types. There are passive edit directives that do not

influence the **seraMC** edits, inactive directives that must be placed before active directives, active directives that must appear before a particular edit directive, and active edit directives. The passive edit directive may include patient name (#patient), date (#date), and beam (#beam). These edit directives are used in **seraPlot** to label the patient, date, and beam information when generating dose volume histograms or dose depth profiles. Although there are many edit directives, only a few are discussed within this report. A user's manual on SERA should be consulted for further detail pertaining to other edit directives [38, 39].

The minimum set of edit directives to allow for the generation of contours needed for the post processing of the dose and for generation of dose volume histograms and dose depth profiles are shown in Figure 36.

```
edit_dir
#patient VG UTMCK T1 MRI
#date May_9_1975
#beam BMRR_12_CM
 mw_min 1.0
 b10_blood 1.0
 refvol 1.0 0
 ref_reg brain tumor target
 ottocon 40 40 scalp tumor target brain
 delta 0.25
 eps 0.05
#Dose-Volume Histogram for Brain
 in_reg brain target
 b10_blood 13
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
#Dose-Volume Histogram for tumor
 in_reg tumor target
 b10_blood 45.5
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
#Dose-Volume Histogram for Scalp
 in_reg scalp
 b10_blood 19
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
```

Figure 36: Use of Edit Directives To Generate Dose Volume Histogram and a Concatenated Dose Contour (.cdf.sz) File

The mw_min edit directive specifies the Power level of the reactor. The b10_blood edit directive specifies the boron blood concentration (in parts per million), the ref vol and the ref region make the edit directives active within the specified regions. The ottocon edit directives in the most important directive of all since it allows for the generation of a concatenated dose contour file (.cdf.sz) which is a zipped file that contains all of the dose information that overlays particular image planes containing those regions specified in the ottocon directive. For example the "ottocon 40 40 scalp" edit directive results in 40X40 dose contour grid which overlays any image plane containing the scalp region. In addition to the ottocon edit directive, the DVbs edit directive allows for dose volume histograms to be generated for those volumes specified by the in_reg edit directive. One may also want to specify the b_10 blood value and the RBE values for those regions. The RBE's shown in Figure 36 as 1.3, 1.0, 3.2, 3.2, 1.0, 1.0 are for the boron, gamma, nitrogen, hydrogen, other, and gamma production. In addition, one can also provide RBEs for proton recoil (E>16.9 MeV) and ultrafast dose components using the rbe edit directive. And example of the dose volume histogram is shown earlier in Figure 7.

3.5 SERADOSE

Once dose files have been generated using seraCalc, the dose contours can be viewed on the images using seraDose. The primary objective of seraDose is to allow for visualization of the dose contours from the dose contour (.cdf.sz) files generated by seraCalc. This task can also be implemented in sera3d. Much of the dose contour displaying presented within this work has used either **sera3d** or within Matlab^{tm} using softwares developed by the author.

3.6 SERAPLOT

Sera plot allows for a different type of visualization of the dose in that it allows for displaying of dose-depth plots and for dose volume histograms. Within the dose depth profile option, one can visualize the total dose, b_10 dose, gamma dose, N-14 dose, hydrogen dose, G1-fluence, Gp-2 fluence, thermal fluence, gamma production, ultrafast gamma production, , and other components of the total dose along the beam centerline. The **seraMC** output file (.rst) is used to visualize these values along the beam centerline. Once can also include the stride distance along the beam center line, a normalization factor, and a boron concentration factor to scale the dose depth profile. For example a stride distance of 1 will pick every point along the beam centerline, a normalization factor of 3.5 is a multiplicative factor which is applied uniformly to every data point in the dose, and the boron concentration factor of 13 ppm is a multiplicative factor applied only to the boron dose, and indirectly to the total dose. **SeraPlot** is useful in visualizing the dose and the dose rate as a function of depth along the beam centerline. An example of the dose rate depth profile generated using **seraPlot** is shown earlier in Figure 8. **SeraPlot** can also be used to generate the dose volume histograms for the various regions (as is shown in Figure 7).

3.7 SERAPLAN

SeraPlan allows for the dose (.rst) file from **seraCalc** to be combined into a treatment plan that couples different beam fractions into one single treatment plan. Generally, this is useful if the physicist wants to treat the tumor from various angles or at various times. **SeraPlan** allows for incorporation of the boron concentration and for exposure data from

multiple fractions to be incorporated into one treatment plan and for generation of one set of contour data that represents the collective dose contours that would be administered if several fractionations were given to the patient with different beam alignments for each fractionation. Currently, under the Phase I/II FDA approved guidelines, patients were administered a single fraction, thus the treatment plans within this work only focused on a single beam, single fraction setup. Thus, this part of SERA is under utilized within this work.

IV. Image Registration

Generally, image registration can involve resampling of images, smoothing (low pass filtering), affine transformations (rigid body), some sort of optimization (to minimize the objective function to align the two images), and in some cases intensity transformations (for fine tuning of the registration). The images can range from belonging to the same individual from the same modality to one that is from different patients for different modalities (like PET/MRI). For inter modality registration such as PET/PET or MRI/MRI, the problem is not quite as difficult as that for intra-modality registration for a given patient. Intra-modality registration like PET/MRI is complicated by the fact that the intensities of the two images do not correlate with one another. Likewise, objects in one image may or may not be readily apparent in the other. The primary objective of this work is to focus on intra-modality registration for a given subject. Intra-modality registration for a given subject involves rigid body transformations (constrained), image classification into grey matter, white matter, and cerebro spinal fluid (CSF), optimization, and resampling of images to get new images that have been transformed onto a "registered" coordinate system.

4.1 RESAMPLING OF IMAGES

Resampling of images includes determining voxel intensities for the new image once a mapping of one image into the next is found. The simplest approach is to take nearest neighbor (or zeroth order) value for the new intensities. The advantage of this approach is that it preserves the original intensities but can result in image that is quite degraded. Another

approach would be to take tri-linear interpolation (first-order) to resample the data. An example of bilinear interpolation (which can be extended to tri-linear) would be as follows:

Let points a through p represent the original regular grids of pixels (x and y represent the location values for these points, a through p). Point u can be a point whose value is to be determined using bilinear interpolation. Points q to t are used as intermediates in the computation. From this, the following intermediates can be computed before computing the value for u,

Let v_r, v_s, represent intermediate pixel intensities for (first interpolation)

$$V_r = ((x_g - x_r)v_f + (x_r - x_f)v_g) / (x_g - x_f)$$

$$V_s = ((x_k - x_s)v_j + (x_s - x_j)v_k) / (x_k - x_j)$$

From this, the pixel value for u can be determined as follows (i.e. second interpolation):

$$V_u = ((y_u - y_s)v_r + (y_r - y_u)v_s) / (y_r - y_s).$$

Shown in Figure 37 are the points.

Bilinear interpolation can be extended to three dimensions using 8 nearest points for interpolation. Extension to the third dimension is trivial. Many of these interpolation subroutines are already built in to Matlab™ as functional routines and can be readily implemented. Other more common interpolation technique may involve polynomial

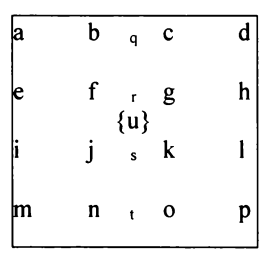


Figure 37: Example of Bilinear Interpolation

interpolations (like Lagrange polynomials) or sinc interpolation (as is currently used in Statistical Parametric Mapping software). The sinc interpolation (in time domain) has the added advantage in that it avoids certain artificats in images (since ideal interpolation for images should be done in the Fourier frequency domain).

4.2 SMOOTHING

Generally, smoothing is important for registering images from different patients but is of not much practical use within this work. Smoothing is used to "spread" out the different areas of the brain since homologous brains of subjects of different subjects can never be precisely registered. This is similar to blurring two images from various patients to try and reduce the discrepancy between the two images. A discrete Gaussian convolution kernel can be used if needed. Smoothing of images followed by registration can provide initial guesses for parameters that would register the two images. This initial guess can then be applied to more rigorous registration using non-smooth images.

4.3 AFFINE TRANSFORMATIONS

Transformation is the mapping of points in one coordinate system (or in one frame of reference in an image) into another. For each point, (x, y, z) in an image, a mapping can be defined into the coordinates of another space (x', y', z') as follows:

$$x' = m_{11}x + m_{12}y + m_{13}z + m_{14}$$
$$y' = m_{21}x + m_{22}y + m_{23}z + m_{24}$$
$$z' = m_{31}x + m_{32}y + m_{33}z + m_{34}$$

This can also be re-written in matrix notation as follows:

y = Mx, where

$$y = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}, \quad M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \text{ and } x = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$m_{n,m}$, where n and m range from 1 to 4 are the 9 cosine angles of rotation and 3 translations about the given axis. When n = 1, it represents the rotation and translation of x,y,and z about the x' axis. When n=2, it represents the rotation and translation about the y' axis, and when n=3, it represents the rotation and translation about the z' axis. When n=4, it represents translation only without any rotation. The cosine angles of rotation (n=1 to 3 and m = 1:3), are also more commonly known as Euler angles of rotation, and can be expressed in spherical coordinates (with three parameters of rotation and three parameters of translation) for a rigid body transformation for the same patient is as follows:

Rotation about the X axis:

X axis : (i.e. Θ or Roll)

$$M_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & \sin(\Theta) & 0 \\ 0 & -\sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Y Axis: (i.e. Φ or Pitch)

$$M_y = \begin{pmatrix} \cos(\Phi) & 0 & \sin(\Phi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\Phi) & 0 & \cos(\Phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Z Axis: (i.e. Ω or Yaw)

$$M_z = \begin{pmatrix} \cos(\Omega) & \sin(\Omega) & 0 & 0 \\ -\sin(\Omega) & \cos(\Omega) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The translation matrix can be expressed as follows:

$$T = \begin{pmatrix} 1 & 0 & 0 & x_{trans} \\ 0 & 1 & 0 & y_{trans} \\ 0 & 0 & 1 & z_{trans} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where x_{trans}, y_{trans}, and z_{trans} represent the translations in x, y, and z of the original image coordinate to system to get alignment in the new coordinate system (x', y', z').

Collectively, $M_xM_yM_zT$, represents the transformation and translation that is needed in the spherical system to register the two images (that is if the two images have the same pixel sizes). The order in which the rotations are performed is important. A rotation about X axis of $\pi/2$ followed by an equivalent rotation about Y axis would produce different results if the order of operations was reversed. An algorithm developed by the author to accomplished optimization and to solve the simple rigid body transformation and translation is included in the appendix (Appendix L). The Euler_transform8.m program attached in Appendix L generates 10 random data points, picks some pitch, yaw, and roll, and adds the known

coordinate transformations and translation to this synthetic data. It then adds 10% noise to the synthetic data. Then starting with an initial parameter guess of [0 0 0 0 0 0] for the pitch, yaw, roll, and the translation parameters, respectively, the code optimizes the solution to determine the known transformations/translations. The original points, the known transformation/translation vector, and constrained optimization results are shown in Appendix L. The error in each of the points was within half of a percent. This code only implements rigid body transformation and translation and can be used to align images if one specifies corresponding points in the two images. However, this code is subject to non-reproducibility and user dependency. A preferred method (described in a later section) would allow for automatic image registration.

Because MRI and PET images as retrieved at the UT Medical Center have different pixel sizes, another transformation matrix must be included to account for the two different pixel sizes. The pixels for the two images (f and g) will often have anisotropic pixel sizes. Thus, for an MRI image (f) retrieved at the UT medical center, the pixel size are generally .8975 mm by .8975 mm (in x,y) and 7.5 mm in depth (z axis). In addition, the MRI image is generally a 256 X 256 image for a total of about 19-20 image planes. Thus the affine transformation matrix which maps these pixels into the normal Euclidean space would be as follows:

$$M_f = \begin{pmatrix} .8975 & 0 & 0 & -256*.8975/2 \\ 0 & .8975 & 0 & -256*.8975/2 \\ 0 & 0 & 7.5 & -19*7.5/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The M_f transformation will map the original MRI image into the Euclidian space whose axes are parallel to those of the image and distances will be measure in mm, with the origin at the center of image. A similar matrix can be defined for the PET image (g) as acquired at the UT Medical Center. Generally the PET image as acquired at the UT Medical Center consists of 128X128X47 images with a sampling resolution of 2.11X2.11X3.75 mm in the x,y, and z dimensions, respectively. Thus, the corresponding matrix that maps this into the Euclidean space is:

$$M_g = \begin{pmatrix} 2.11 & 0 & 0 & -128*2.11/2 \\ 0 & 2.11 & 0 & -128*2.11/2 \\ 0 & 0 & 3.75 & -47*3.75/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The objective function of any co-registration is to determine the rigid body transformations that will map an image (such as f for the MRI) to that of another (such as g for the PET). For this to be possible, a rigid body transformation matrix, M_r, must be determined such that $Mg^{-1}M_rM_f$ will register images. Various authors have developed different algorithms to solve such image registration problems [46]. These algorithms range from being semi-automated to fully automated algorithms. Regardless of the approach used, most of these algorithms involve some sort of least squares optimization.

4.4 OPTIMIZATION

The objective of optimization in image registration is to determine a set of parameters for which some function of the parameters is minimized. The simplest method of determining optimum parameters for a model would be to minimize the sum of the squared differences

between the model and a set of real world data (X^2). In an image registration sense, this is the equivalent of minimizing the squared difference between the template image and the object image. Due to the nature of the image registration problem, several parameters need to be determined. In a Cartesian coordinate system, this can involve 12 unknowns (9 Euler cosine angles, and 3 translation parameters) or 6 parameters in the spherical coordinate system (3 rotations and 3 translations). The 12 unknowns in the Cartesian coordinate system can be used to constrain the problem in the spherical coordinate system.

Because the search space is large due to the fact that there are many parameters involved in optimizing the problem, it is generally not possible to exhaustively iterate through the entire parameter search space. Thus, the usual approach is to make an initial guess for the parameter, and begin iteratively searching from there. A judgement is then made about how the parameter estimate should be modified, before continuing on to the next iteration. Once some convergence criteria is reached (like when X^2 or the sum of the squared difference stops decreasing), optimization is terminated.

Image registration is essentially an optimization problem in that one image (the object image) is spatially transformed so that it matches the other (the template image), by minimizing X^2. The parameters that are optimized are those described in the previous section. Various algorithms are available for optimization. But the algorithm of choice, and that which is implemented in Statistical Parametric Mapping Software developed in Matlab™ is similar to that of Gauss-Newton and is illustrated as follows:

Let $d_i(p)$ be a function that describes the difference between the object and template images at voxel, i, where the vector of the model parameters has value p. Thus, for each

voxel, i, a first order approximation of Taylor's Theorem can be used to estimate the value that this difference will take if the parameters p are increased by some step size, t, where t is also a vector representing the step size for individual parameters, p. Using Taylor Series approximation, the next value for $d_i(p+t)$ would be as follows:

$$d_i(p+t) = d_i(p) + t_1 \frac{\partial d_i(p)}{\partial p_1} + t_2 \frac{\partial d_i(p)}{\partial p_2} + t_3 \frac{\partial d_i(p)}{\partial p_3} + \cdots$$

where the series is truncated after the first derivative terms.

From this, one can get simultaneous equations (of the form Ax = b) to estimate the values that t should have to minimize $\sum_i d_i(p+t)^2$:

$$\begin{pmatrix} -\frac{\partial d_1(p)}{\partial p_1} & -\frac{\partial d_1(p)}{\partial p_2} & -\frac{\partial d_1(p)}{\partial p_3} & -\frac{\partial d_1(p)}{\partial p_k} \\ -\frac{\partial d_2(p)}{\partial p_1} & -\frac{\partial d_2(p)}{\partial p_2} & -\frac{\partial d_2(p)}{\partial p_3} & -\frac{\partial d_2(p)}{\partial p_k} \\ -\frac{\partial d_3(p)}{\partial p_1} & -\frac{\partial d_3(p)}{\partial p_2} & -\frac{\partial d_3(p)}{\partial p_3} & -\frac{\partial d_3(p)}{\partial p_k} \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{\partial d_j(p)}{\partial p_1} & -\frac{\partial d_j(p)}{\partial p_2} & -\frac{\partial d_j(p)}{\partial p_3} & -\frac{\partial d_j(p)}{\partial p_k} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_k \end{pmatrix} \cong \begin{pmatrix} d_1(p) \\ d_2(p) \\ d_3(p) \\ \vdots \\ d_k(p) \end{pmatrix}$$

where, k represents the total number of parameters and j represents the total number of voxels, i.

From this, one can derive an iteration scheme for improving the parameter estimates. For iteration n the parameters p are updated as follows:

$$p^{(n+1)} = p^n + (A'A)^{-1} A'b$$

where,

$$A = \begin{pmatrix} -\frac{\partial d_1(p)}{\partial p_1} & -\frac{\partial d_1(p)}{\partial p_2} & -\frac{\partial d_1(p)}{\partial p_3} & & -\frac{\partial d_1(p)}{\partial p_k} \\ -\frac{\partial d_2(p)}{\partial p_1} & -\frac{\partial d_2(p)}{\partial p_2} & -\frac{\partial d_2(p)}{\partial p_3} & & -\frac{\partial d_2(p)}{\partial p_k} \\ -\frac{\partial d_3(p)}{\partial p_1} & -\frac{\partial d_3(p)}{\partial p_2} & -\frac{\partial d_3(p)}{\partial p_3} & & -\frac{\partial d_3(p)}{\partial p_k} \\ \vdots & \vdots & \vdots & & \vdots \\ -\frac{\partial d_j(p)}{\partial p_1} & -\frac{\partial d_j(p)}{\partial p_2} & -\frac{\partial d_j(p)}{\partial p_3} & \cdots & -\frac{\partial d_j(p)}{\partial p_k} \end{pmatrix}, \quad b = \begin{pmatrix} d_1(p) \\ d_2(p) \\ d_3(p) \\ \vdots \\ d_k(p) \end{pmatrix}$$

This process is repeated until X^2 no longer decreases or for a fixed number of iterations. One of the problems with the previous approach is that there is no guarantee that a global solution will be reached since the algorithm can get trapped in a local minimum. To reduce this problem, the starting point, p should be as close as possible to the optimum solution. The number of potential local minima can be decreased by working with smooth images (smoothing of images has the effect of making the first order Taylor approximation more accurate for larger displacements). Once registration is close to the true solution, registration can continue with less smooth images.

In addition to spatial transformations, (p_s), one can also incorporate intensity information into the context of the optimization problem. The optimization problem can be assumed to minimize two sets of parameters, those that describe spatial transformation (p_s) (as described previously and implemented in the algorithm that I developed) and those that describe intensity transformations (p_t). Thus, one can modify the objective function as follows:

$$d_i(p) = f(s(x_i, p_s)) - t(x_i, p_t),$$

where f represent the intesity transformation function and t represents the spatial transformation function as described previously. The intensity transformation can be linear or non-linear. However, the objective function ($d_i(p)$) is more appropriate for inter-modality registration in which the intensities in the two modalities are directly related to the objects that they represent and have similar representations across images (for example MRI/MRI intensities or PET/PET intensities). However, this is not appropriate for PET/MRI registration unless the intensities in one are scaled accordingly to the other. Thus, an appropriate intensity function must be sought before using an objective function that accounts for intensity differences along with spatial transformation differences in the optimization process. The intensity transformation function must also account for various differences in the modality (PET, SPECT, ^{18}F-DOPA, ^{18}F-BPA, ^{18}F-FDG, T1-MRI, T2 MRI, etc..) or different anatomical regions (cortical gray matter, sub-cortical gray matter, cerebellum, etc.), or some combination of the above.

4.5 INTRA-MODALITY REGISTRATION THEORY (SPM APPROACH)

Automatic co-registration of brain images of the same subject from different modalities has proven itself to be challenging since there is nothing obvious about what to minimize in the two modalities. Intensity values that represent a particular object in one modality may represent a completely different object in the other. Thus, intra-modality registration is not a trivial task. At least two algorithms have been developed to carry out intra-modality registration. Woods has developed the Automated Image Registration (AIR) algorithm to co-register MRI and PET images [47]. However it is complicated by the fact that laborious manual scalp editing must be carried out to remove any tissue that is not part of the

brain. AIR 3.08 has been setup on the engineering workstation (felix.engr.utk.edu) under /ne1/mkhan/registration/AIR/AIR3.08 directory.

A more recent algorithm (SPM99) involves the matching of images by maximizing mutual information developed by Ashburner and Friston [48-52, 54]. An alternative example would involve the manual identification of homologous objects in both images (for example matching Gray Matter, White Matter, and Cerebrospinal Fluid regions). These landmarks could then be used to align the images together. The two approaches, one using mutual coregistration, and the other using certain landmarks, are currently implemented in the Statistical Parameteric Mapping algorithm (SPM99) developed by researchers and collaborators of the Wellcome Department of Cognitive Neurology in United Kingdom and is the choice of software for the registration work which implements fully automated image registration. The SPM algorithm has the added advantage in that it is reproducible. Earlier algorithms which involved visual registration (2D and 3D visual registration) developed by the author and the semi-automated algorithms (which involved picking corresponding points in two images and implementing rigid body transformation and translation) failed in reproducibility and required manual interaction to register the two images.

The SPM algorithm is described as follows:

1. Determine the affine transformation that maps between images (f, g) and the template (t_f, g_t) by minimizing the sum of squares difference between f and t_f, and g and t_g (where t_f and t_g are template images). These transformations are constrained such that only the parameters that describe rigid body transformation are allowed to differ.

2. Segment or partition the images using the probability images and the modified mixture model algorithm. Then, map between the probability images to images f and g having been determined in step 1.

3. Co-register the partitions using the rigid body transformations computed from step 1 as starting estimates.

Step 1 is described above in an earlier section regarding transformation of images. Steps 2 and 3 are further discussed in SPM related papers [48-52].

Let M_f, M_g, and M_t be matrices which map from the voxel coordinates of images f, g, and the templates into their Euclidian space. For the UTMCK MRI images, M_f is as follows:

$$M_f = \begin{pmatrix} .8975 & 0 & 0 & -256*.8975/2 \\ 0 & .8975 & 0 & -256*.8975/2 \\ 0 & 0 & 7.5 & -19*7.5/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

For the UTMCK PET images, M_g is as follows:

$$M_g = \begin{pmatrix} 2.11 & 0 & 0 & -128*2.11/2 \\ 0 & 2.11 & 0 & -128*2.11/2 \\ 0 & 0 & 3.75 & -47*3.75/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and M_t depends on the template image that is used to align the PET or MRI image to. Generally it is possible (and easier) to obtain matching of images of most normal brains to a template image of the same modality using a twelve (or even nine) parameter transformation,

as described earlier. On can easily register image g to a template t_g from the same modality, and similarly register f to a template image, t_f, from the same modality. Let the transformation matrices which carry out these mappings of image f to the template t_f and g to t_g be M_{ft} and M_{gt}, respectively. Thus, a mapping from f to g, now becomes $M_g^{-1}M_{gt}M_{ft}^{-1}M_f$,

where $M_{gt}M_{ft}$ is the trasformation matrix (M_r) which maps f to g. However, it is important to note that this affine transformation has not been constrained to be rigid. For this to be a rigid body transformation, M_{gt} must be decomposed into a matrix that performs a rigid body transformation (M_{gr}), and one that performs scaling (M_{gr}) and shearing (M_{ta}). In other words, $M_{gt} = M_{gr}M_{ta}$. Similarly, M_{ft} can be decomposed into a scaling (M_{fr}) and shearing (M_{ta}) components. In other words, $M_{ft} = M_{fr}M_{ta}$. Also, note that M_{ta} is same for both f and g. The earlier mapping from f to g, now becomes $M_g^{-1}M_{gr}(M_{ta}M_{ta}^{-1})M_{fr}^{-1}M_f$. This represents the rigid body transformation that describes the mapping of f to g via the use of templates. This rigid body transformation and translation can be expressed as follows:

$$M_{gr} = TM_xM_yM_z = \begin{pmatrix} 1 & 0 & 0 & q1 \\ 0 & 1 & 0 & q2 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q_4) & \sin(q_4) & 0 \\ 0 & -\sin(q_4) & \cos(q_4) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(q_5) & 0 & \sin(q_5) & 0 \\ 0 & 0 & 0 & 0 \\ -\sin(q_5) & 0 & \cos(q_5) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(q_6) & \sin(q_6) & 0 & 0 \\ -\sin(q_6) & \sin(q_6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{fr} = T'M'_xM'_yM'_z = \begin{pmatrix} 1 & 0 & 0 & q7 \\ 0 & 1 & 0 & q8 \\ 0 & 0 & 1 & q9 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q_{10}) & \sin(q_{10}) & 0 \\ 0 & -\sin(q_{10}) & \cos(q_{10}) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(q_{11}) & 0 & \sin(q_{11}) & 0 \\ 0 & 0 & 0 & 0 \\ -\sin(q_{11}) & 0 & \cos(q_{11}) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(q_{12}) & \sin(q_{12}) & 0 & 0 \\ -\sin(q_{12}) & \sin(q_{12}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{ta} = Scale * Shear = \begin{pmatrix} q_{13} & 0 & 0 & 0 \\ 0 & q_{14} & 0 & 0 \\ 0 & 0 & q_{15} & 0 \\ 0 & 0 & 0 & q_{16} \end{pmatrix} \begin{pmatrix} 1 & q_{17} & q_{18} & 0 \\ 0 & 1 & q_{19} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Where, parameters, q_1, q_2, q_3, represent translation of g which will align with the template using rigid body translation, q_4, q_5, and q_6, represent the Roll, Pitch, and Yaw, respectively, which will transform g into the template coordinate system using rigid body transformations. Similarly, q_7, q_8, q_9, represent the rigid body translation of f which will align with the template, and q_{10}, q_{11}, and q_{12}, represent Roll, Pitch, and Yaw, respectively, which transform g into the template coordinate system using rigid body transformations.

When aligning the images to templates, one results in a set of parameters, $\mathbf{q} = (q_1, q_2, \ldots)$, which must be optimized in order to determine the transformations (and translations) that will minimize the sum of squares difference between the images and the templates. The Gauss-Newton method of iteratively optimizing the objective function was described in a previous section (Section 4.4). It essentially involves the use of generating a matrix A^tA and a vector A^tb for each iteration, solving the equations, and incrementing the parameter estimates:

$$p^{(n+1)} = p^n + (A'A)^{-1}A'b$$

For the purpose of this optimisation of matching two images to templates, one can define two matrices, $M_1 = (M_t^{-1}M_{ft}M_f)^{-1}$ and $M_2=(M_t^{-1}M_{gt}M_g^{-1})$. Thus, A can be described as follows:

$$A=\begin{pmatrix}
-\frac{\partial f(M_1x_1)}{dq_1} & \cdots & -\frac{\partial f(M_1x_1)}{dq_6} & 0 & & 0 & -\frac{\partial f(M_1x_1)}{dq_{13}} & \cdots & -\frac{\partial f(M_1x_1)}{dq_{18}} & t_f(x1) & 0 \\
-\frac{\partial f(M_1x_2)}{dq_1} & \cdots & -\frac{\partial f(M_1x_2)}{dq_6} & 0 & & 0 & -\frac{\partial f(M_1x_2)}{dq_{13}} & \cdots & -\frac{\partial f(M_1x_2)}{dq_{18}} & t_f(x2) & 0 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & & 0 & -\frac{\partial g(M_2x_1)}{dq_7} & \cdots & -\frac{\partial g(M_2x_1)}{dq_{12}} & -\frac{\partial g(M_2x_1)}{dq_{13}} & \cdots & -\frac{\partial g(M_2x_1)}{dq_{18}} & 0 & t_g(x_1) \\
0 & & 0 & -\frac{\partial g(M_2x_2)}{dq_7} & \cdots & -\frac{\partial g(M_2x_2)}{dq_{12}} & -\frac{\partial g(M_2x_2)}{dq_{13}} & \cdots & -\frac{\partial g(M_2x_2)}{dq_{18}} & 0 & t_g(x_2) \\
 & & 0 & \vdots & & \vdots & \vdots & & \vdots & & \vdots
\end{pmatrix}$$

Where, f(x) represent the intensity of the image f at the position x, g(x) is the intensity at position x, $t_f(x)$ is the intensity of the template image at position x, and $t_g(x)$ is the intensity of the template image at position x. Mx represent the mapping from one coordinate system to another (i.e., $f(M_1x_1)$ represents the mapping of pixel, x_1 at this position, into the new position which corresponds with the template coordinate system. The vector b, is computed as follows:

$$b = \begin{pmatrix} f(M_1x_1) - q_{19}t_f(x_1) \\ f(M_2x_2) - q_{19}t_f(x_2) \\ \vdots \\ g(M_1x_1) - q_{20}t_f(x_1) \\ f(M_2x_2) - q_{20}t_f(x_2) \\ \vdots \end{pmatrix}$$

The parameters, q_{13} to q_{18} describe the non-rigid transformations (shearing of images) could also be derived from either f or g, but instead these parameters are estimated from using both images. The results in SPM are biased so that the image that fits the template better has a greater influence over parameter estimates. This is achieved by weighting rows A and b that corresponds to the different images. The weights are derived from the sum of squares difference between the template and object images, obtained from the previous solution of q. The weights are for f and g, respectively:

$$\frac{I}{\sum_{i=1}^{I}(f(M_1x_i) - q_{19}t_f(x_i))^2} \qquad \textit{and} \qquad \frac{I}{\sum_{i=1}^{I}(f(M_2x_i) - q_{20}t_g(x_i))^2}$$

Once optimisation converges to a final solution, one can obtain the rigid body trasformation that approximately maps between f and g. This also gives the affine transformation matrices that map between the images and the templates, which can be used in

the partitioning of images into various segments (i.e, Gray Matter, White Matter, and Cerebrospinal Fluid, etc.).

Healthy brain can generally be classified into three broad tissue categories (Gray Matter (GM), White Matter (WM), and Cerebro-Spinal Fluid (CSF), on the basis of an MRI image. This classification can be performed on manually on a good quality T1 weighted MRI image by simply selecting suitable image intensity ranges that encompass most of the voxel intensities of a particular tissue type. (This is possible because of the varying tissue T1 relaxation properties). However, this manual selection can be highly subjective. Several groups have used clustering algorithms to partition MR images into different tissue types by either using images acquired from a single MRI image or by combining information from two or more registered images acquired using different scanning sequences (proton-density and T2-weighted). SPM uses the modified version of one of the clustering algorithms. The clustering algorithm used within SPM is the maximum likelihood "mixture-model" algorithm [53]. Essentially, SPM assumes that the MR image consists of a number of distinct tissue types (clusters) from which every voxel has been drawn. The intensities of the voxel belonging to each of these clusters conform to a multivariate normal approximation, which can be described by a "mean" vector, a covariance vector, and the number of voxels belonging to the distribution. In addition, SPM uses approximate knowledge of the spatial distribution of these clusters, in the form of probability images provided by Montreal Neurological Institute that were derived from MR images of a large number of subjects. The original images were segmented into binary images of GM, WM, and CSF, and all normalized to the same space using a 9 parameter (3 translations, 3 rotations, and 3 orthogonal zooms) affine transformation.

The probability images are the means of these binary images, so that they contain values in the range of 0 to 1. These images represent the prior probability of a voxel being either GM, WM, or CSF after an image has been normalized to the same space using a 9 parameter affine transformation. The difference between the approach used in SPM and the Mixture Model described elsewhere [53] is in the update of the belonging probabilities. The assignment of a probability for a voxel is based upon determining the probability of it belonging to cluster, ci, given that it has the intensity, v. This is based upon simple Bayesian statistics:

$$p(c_i \mid v) = \frac{p(v \mid c_i)p(c_i)}{\sum_{k=1}^{K} p(v \mid c_k)p(c_k)}$$

$p(v / c_i)$ represents the probability density for the cluster c_i at a value v, and $p(c_i)$ is the prior probability of the voxel belonging to cluster c_i (where $p(c_i)$ is based on apriori probability images) where p(ci) can be calculated as follows

$$p(c_i) = n_i b_i / \sum b_i$$

where b_i represent the value at the corresponding position of the *i*th probability image, n_i represents the number of voxels known to belong to a cluster, and $\sum b_i$ is the integral over the image. Step by step implementation of using probability images to map MRI or PET images in SPM onto templates are described in SPM related papers.

V. Statistical Uncertainty with SPM

The SPM algorithm [48-52, 54] developed by the Wellcome Department of Cognitive Neurology, University College, London, is written in Matlab[tm] and provides for a method of

detecting regional, task-related changes in brain activity from neuroimaging data along with image registration, smoothing of data, and resampling of data.

Keibel has cross-validated MRI and PET coregistration using Friston's SPM99 algorithm with another more commonly used algorithm developed by Woods from the University of California, Los Angeles (UCLA) using 32 simulated PET images with known rigid body transformations [54]. It was shown that using SPM without scalp editing and applying random transformation and translations, the mean error in the x, y, and z translation was .670 ± .236 mm, -1.150 ± .172 mm, and 2.071 ± .295 mm, respectively. In the transformation matrices, it was shown that pitch, roll, and yaw had errors of -1.131 ± .082 degree, -.145 ± .151 degree, and -.260 ± .078 degree, respectively [54]. Thus, the translation uncertainty was at most about 2 mm while angular uncertainty was at most about one degree [54]. Scalp editing PET images prior to registering can further reduce the uncertainty in PET/MRI image registration.

VI. Dose Adjustment

Once registered PET/MRI images are obtained then one can adjust the boron component of the total dose. The PET adjusted dose can than be compared with standard methodology. The boron dose contours shown in Figure 9 are for a standard 1 ppm boron distribution within the entire brain, scalp, target, and tumor regions. The unit of dose shown in Figure 9 is cGy/MW-min/ppm. The contours are homogeneous and well centered due to the fact that the scaling of the boron component of the dose has not been taken into account. The total dose is usually scaled by three factors under the standard protocols. One factor accounts for the treatment time (40-50 minutes) as well as the power of the reactor (2 MW). Another factor accounts for relative biological effectiveness (RBE) of killing cells with neutrons, photons, as well as the chemical nature of the process within tumor and brain cells. A third and final factor accounts for the boron distribution differences between the normal cells within the brain volume and tumor cells in the tumor and target volumes. The dose which takes into account for all three of these factors is generally referred to Gray-Equivalent because it represents the "boron equivalent" dose that would be delivered if X-rays or photons were being used to deliver the dose as opposed to using BNCT.

Although ^{10}B has the highest neutron capture cross section in a BNCT therapy situation, the concentration of nitrogen and hydrogen in the body is high and can also contribute significantly to the total dose delivered to the tumor. The total dose is the sum of the individual components contributing to the dose adjusted by a weighting factor, as shown in the following equation:

$D_i = A_i * D_{B\text{-}10} + B_i * {}^{14}N + C_i * \gamma + D_i * {}^{1}n$

where, A, B, C, and D are the respective weighting factors, i is at a given location, and ^{10}B, ^{14}N, γ, and fast neutron, ^{1}n, are the principal components of the dose that contribute to the total dose during therapy.

The total dose that is scaled appropriately for various factors are shown in Figures 7 and 8. Figure 7 accounts for RBE differences, treatment time using the BNL 2MW reactor and 50 minute exposure time, and the homogeneous scaling of the boron dose with 15 ppm of boron within the scalp and normal brain regions and with 52.5 (or 3.5 times as much) in the target and tumor volumes. Thus, the total BNCT "equivalent" dose in Gray-Equivalents (Gy-Eq) administered during a treatment protocol is shown in Figure 7 as a dose depth profile. Shown in Figure 8 is a dose rate depth profile along the beam centerline and is presented as a BNCT "dose equivalent" rate in (cGy/MW-min). This also takes into account the various scaling factors (RBE/power/treatment time) along with homogeneous scaling of the boron dose within the tumor and target volumes. The tumor and target volumes lay approximately 1 cm – 9 cm along the beam centerline (for this patient). Thus, there is a clear discontinuity in the predicted dose to the tumor and target cells at the target/normal brain border points. This is the case with assuming an equal boron concentration throughout all tumor cells and scaling the dose homogeneously with the 3.5 scaling factor.

However, this scaling of the dose can be better approached with using registered PET pixel intensity information as shown earlier in Figure 22. The heterogeneous distribution of the boron with the tumor and surrounding volumes is clearly evident within Figure 22. Thus, an

appropriate linear peak to average scaling of the dose could be accomplished as follows with registered ^{18}F-BPA PET images:

$$D_{adjusted}(x,y,z) = \frac{(D_{B-10}(x,y,z)) * (I_{PET}(x,y,z))}{< (I_{ave}(z)) >}$$

where, $(D_{adjusted})_{x,y,z}$ is the "corrected" boron dose, $(D_{b-10})_{x,y,z}$ is the calculated boron dose based on homogeneous B-10 distribution, $(I_{pet})_{x,y,z}$ is the PET 2D pixel intensity for a given image plane at an axial location, and z $<I_{ave}(z)>$ is the average pixel intensity for the brain and tumor regions for a given image plane at axial location, z and can be computed as follows:

$$I_{ave}(x,y,z_i) = \frac{\sum_{i=1}^{n} I_{brain}(x,y,z_i)}{N}$$

The uncertainty with the adjusted dose would be a function of the uncertainty with SPM image registration, SERA dose results, and with the reconstructed PET pixel intensities. The uncertainty with SPM registration is described in an earlier section. The next two sections describe the uncertainty with SERA dose results and with the reconstructed PET pixel intensities.

VII. Statistical Uncertainty with SERA

Executing **SeraCalc** results in the generation of an output file with an .**rst** extension. This output file contains the volume-integrated region edits at the body level defined. Each body level consists of many uniform volume elements. Some of the output contained within this file includes total current in, total current out, total absorption, gamma production, energy deposition, and the first three groups flux information for the various bodies (or regions). The standard deviations associated with these values at the body level for the various patients examined within this work all have ranges much less than 5 % when the Monte Carlo simulations are executed with the 2 million maximum allowable particles.

Although SERA does not explicitly compute statistics associated within each univel in a particular volume, the statistics at the univel level are adequate so long as the body level statistics are sufficient [55]. This is especially true since the statistics at the univel level and the doses at the univel level are interpolated from the tally grid that defines the particular volumes and computes the dose within those volumes [55].

VIII. Statistical Uncertainty with PET Pixel Intensity

To obtain statistical uncertainty with the reconstructed PET image data, one must first determined the number of counts that corresponds with a particular pixel intensity. Based on discussions with personnel at CTI, there are at least two methods available [56]. The first is the Heusmann method, and the other is the first order approximation to the number of counts. The Huesmann method is described elsewhere [57]. The second method, which is a reasonable first order approach according to Casey [56], involves the use of a scaling factor which scales the data in the image space such that the fractional contribution of the pixel intensity, $P_{i,j}$, to the total sum of the pixel intensity is the same in the unreconstructed domain and the reconstructed domain. In other words, the ratio of $P_{i,j}$ to the total sum of the pixel intensities in the reconstructed image space can be directly related to total number of counts measured in the sinogram space. The use of a scaling ratio in the two different image domains is also confirmed by Gregor in the Computer Science department at the University of Tennessee, Knoxville [58]. Thus, with this methodology, one can determine the number of counts corresponding to a given pixel in the image as follows:

$$C_{i,j} = \frac{P_{i,j}}{\sum_{i=1}^{128}\sum_{j=1}^{128} P_{i,j}} * \sum_{l=1}^{160}\sum_{m=1}^{192} S_{l,m}$$

where $C_{i,j}$ represents the number of counts associated with a pixel, $P_{i,j}$, in the 128X128 pet image, and $S_{l,m}$ represent the number of counts measured for a particular ray integral, l, at a particular angle, m.

Determining counts for each pixel in an image plane involves the use of the sinogram data associated with that particular plane. However, it would be much more useful to be able to arrive at a universal constant which scales the ECAT count data (i.e. the pixel intensity) in an image directly to the number of counts without the need for sinogram data. In other words, it is desired to arrive at a ratio of ECAT counts per second in a pixel as output in the reconstructed image data to the number of counts measured at that particular pixel. The pixel intensity, $P_{i,j}$, can be used to determine the ECAT counts by multiplying the pixel intensity by the Quantity Scale (QS) factor as follows:

$$ECAT\ cps = QS * P_{i,j}$$

The quantity scale factor is a factor contained in the image header files and allows one to relate the pixel intensity into ECAT counts per second. The ECAT counts per second can later be converted into a known unit of μci/ml.

If one takes the ratio of ECAT cps to the number of counts, one will note that the individual pixel intensity, $P_{i,j}$, drops out. This leads to what is referred to as a universal scaling factor using the first order approximation to the number of counts as follows:

$$\frac{ECAT\ cps}{Counts} = \frac{QS}{\dfrac{\sum_{l=1}^{160}\sum_{m=1}^{192} S_{l,m}}{\sum_{1}^{128}\sum_{1}^{128} P_{i,j}}}$$

This universal value using the first order approximation was determined for several studies using different planes and different time sequences for a given patient whose sinogram data was available. This value was determined to be 2.06E-5. Little variation in this value was noted if one picked an earlier PET study or a later PET study for the same patient. This value was also

found to be constant as one picked different planes in the same study or if one used different frame times in the dynamic PET data. Thus, this allows one to take the PET ECAT cps and convert it directly to the number of counts by multiplying the PET pixel intensity by the quantity scale factor and scaling it with this value as follows:

$$C_{i,j} = P_{i,j} * QS * (2.06E-5)^{-1}$$

This has been implemented on several patients to determine the number of counts in a given pixel. For most of the studies, it was determined that the number of counts was on the order of about 1000 counts for the data analyzed within this work. This corresponds to about 300-400 counts within the normal brain data, in the worst case. Thus, the uncertainty with 300 counts is about 5.7%. This represents the worst-case scenario for the PET data analyzed within this work since much of the data analyzed here involves the use of later frames to adjust the dose. Later time frames allowed for larger collection time and more concentration of the pharmaceutical in the tumor and target regions. Thus, the higher counts, on the order of a 1000 or so, corresponds to the tumor and target regions whereas the counts on the order of 300-400 represent the number of counts in the normal brain. Shown in Figure 38 is a plot of the statistical uncertainty with a single frame image data plane taken along at the center of the tumor. Also shown in Figure 38 is a plot of the statistical uncertainty with time integrated PET data. It can be noted that the statistical uncertainty with time integrated PET data is significantly reduced when compared with a single frame image.

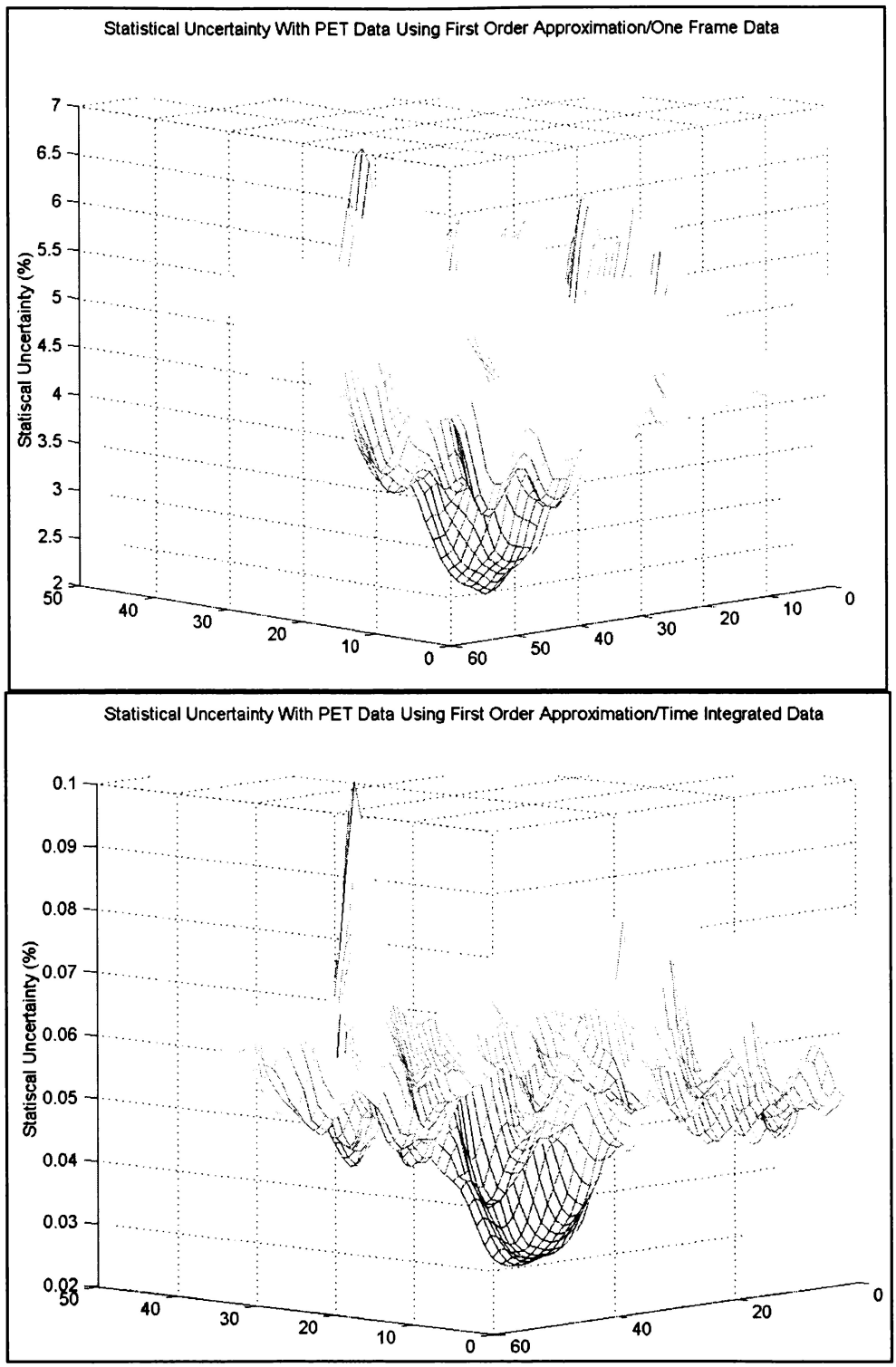


Figure 38: Statistical Uncertainty With PET Pixel Intensities Using First Order Approximation for Individual Frame Data and Time Integrated Data

IX. PET2MRI Algorithm

The **PET2MRI.m** algorithm is the main software that couples the results of the Statistical Parametric Mapping along with the SERA treatment planning program to generate new isodose contours based on ^{18}F-BPA determined boron biodistribution. Shown in Figure 39 is a schematic overview of the main program, its functions, the input/output files used by the program and the program functions, and the type of graphical output of results by the program. The program and the associated functions are included in the appendix (Appendices M, N, O, P, Q, R, and S). **PET2MRI.m** uses several files that are generated by the statistical parametric mapping algorithm and SERA. These files include the .qhd/.qsh MRI image files, the .uvh/.uv files containing volumetric definition of the regions of interest, the .cdf/.chd dose contour file generated by the Monte Carlo portion of SERA, and the registered PET .hdr/.img files which are generated by SPM. These files are combined within the main program and its functions to write out the new dose contours. The output files created by the PET2MRI.m program include an adjusted.cdf file, a BNL.cdf file, Table1.mat, Table2.mat, and several graphical figures. The adjusted.cdf is a PET adjusted dose file written in the same format as the original .cdf file created by SERA. The BNL.cdf file provides dose contours created by scaling the boron component of the dose within the tumor and target volumes by a factor of 3.5. Table 1.mat and Table 2.mat contain several columns of output data that repeats the total area (in mm^2) for a particular dose contour line. For example for a particular image plane, the total area falling within the 90% contour line, the 70% contour line, 50% contour line, and the 10% contour line

could be found within these files. Table 2.mat lists this data for the homogeneously adjusted dose. Table1.mat lists this data for the PET adjusted dose.

Part of the graphical display output on the screen includes a plane by plane view of the original boron dose contours superimposed on top of the image, the homogeneously scaled boron component of the dose superimposed on top of the image, and the PET adjusted boron component of the dose superimposed on top of the image. These are shown for each of the image planes for which SERA outputs the dose contour data. In addition, another figure (Figure 100) is also generated by the program. Within this figure, the output includes the original MRI images with superimposed registered PET image contours, and the tumor volume defined by using an MRI image superimposed over the PET tumor. This output allows for a quick check of the volume definition of the tumor regions if one were to use PET or MRI to define the tumor volume.

X. Results and Discussion

Shown in Table 3 are the results from the use of the SPM algorithm to register the ^{18}F-BPA PET images with T1-weighted MRI images. $M_{11} - M_{33}$ represent the cosines of the Euler angles of transformation, which align the PET images to the T1 MRI images. In most cases, pre-surgery PET images were selected for registration with presurgery T1 MRI images. No presurgery PET data was available for patient MD, thus post surgery MRI and post surgery PET images are registered for this patient. In addition, patient CR was a melanoma patient while all others were GBM patients. This did not affect the registration results in any manner since the tumor type was not a factor in the registration process.

Visual inspection of the registered PET/MRI images has been implemented for all the patients. Based on visual inspection, all patient data were successfully registered using brain segmentation and template matching within SPM. Shown in Figures 40 and 41 are the brain segmentations of T1-MRI and ^{18}F-BPA PET images, respectively, into gray and white matter. Shown in Figure 42 are MRI images with the contours of the MRI images plotted on top, in the left column. Shown in the right column of Figure 42 are the MRI contours plotted on top of ^{18}F-BPA PET images. Based on visual inspection of the MRI image contours plotted on top of the PET images, one is led to believe in the validity of the SPM registration algorithm. The findings in Figures 40-42 are shown only for VG patient. Similar findings are noted for other patients.

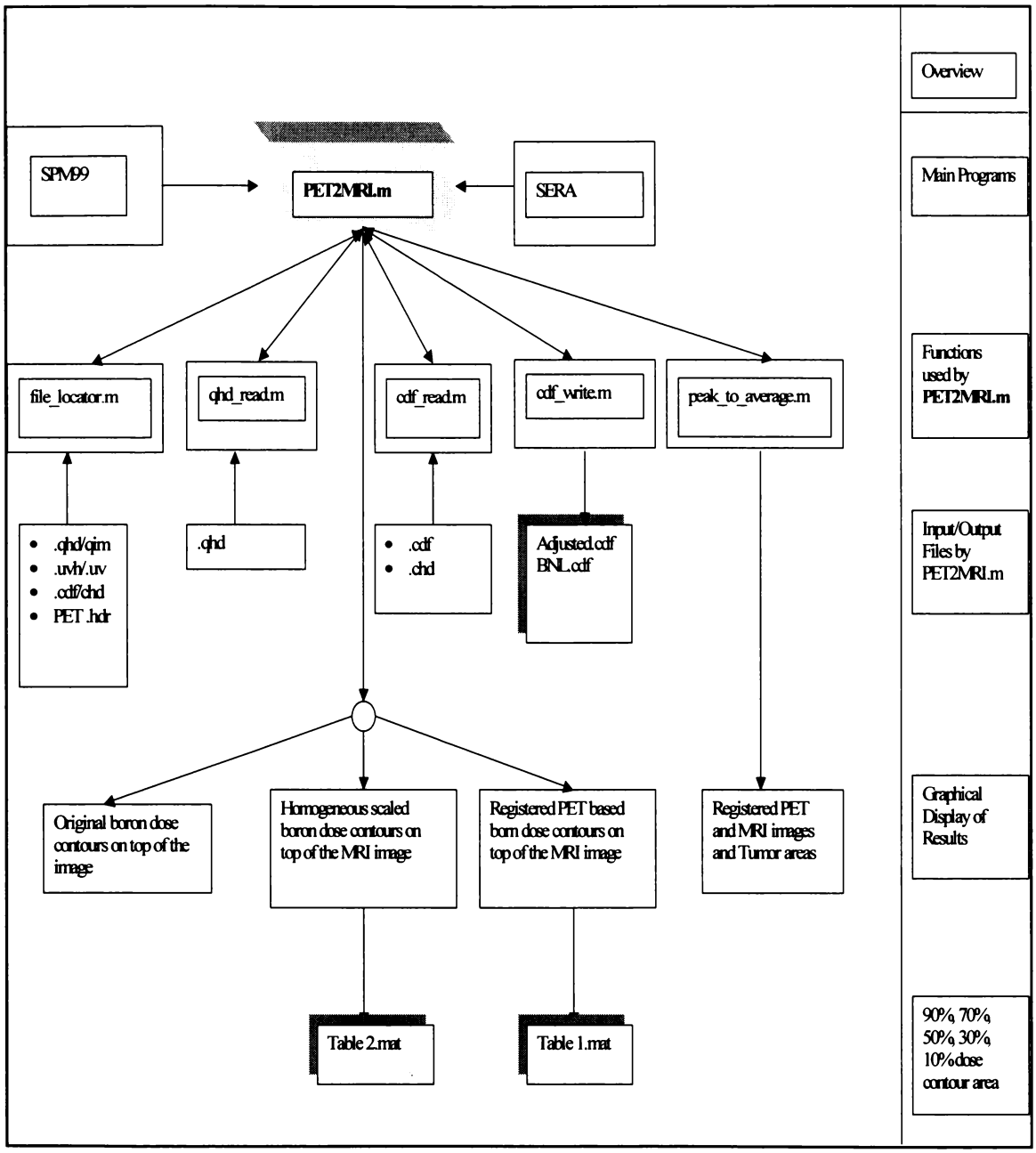


Figure 39: PET2MRI.m Program Overview

Table 3: Transformation/Translation Parameters Which Register PET Images with MRI Images

Patient	M_{11}	M_{12}	M_{13}	M_{21}	M_{22}	M_{23}	M_{31}	M_{32}	M_{33}	T_1	T_2	T_3
VG	.407	-.007	-.231	.011	.403	.554	.016	-.040	2.19	13.02	2.442	7.196
EG	2.337	-.221	.0282	.222	2.337	.091	.002	-.010	.675	-.326	-46.88	-8.07
CR	2.986	.087	.255	-.114	2.959	.569	-.021	-.051	.444	-58.27	-55.67	4.789
LB	2.940	-.179	.0672	.170	2.911	.551	.013	.070	-.6682	-47.19	-73.66	24.86
BP	-2.394	-.531	.193	-.543	2.370	-.547	.007	.062	.669	317.75	28.96	-12.16
MD	2.455	-.022	-.054	.022	-2.455	-.043	.006	.005	.676	-27.93	-69	-5.77
MB	4.737	.405	.978	-.213	4.842	-.622	-.218	.120	.645	-230.8	-199.9	5.207

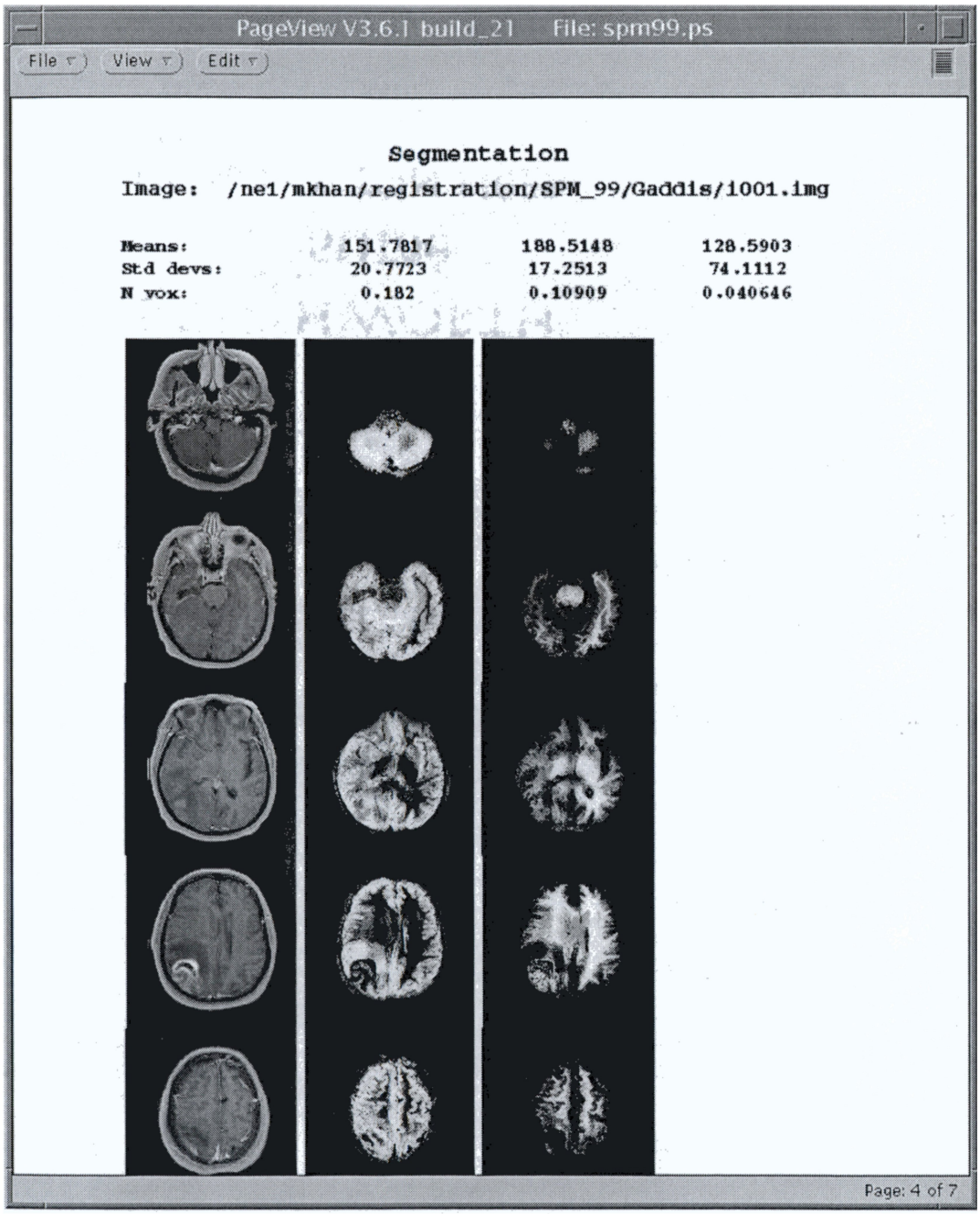


Figure 41: SPM Brain Segmentation of T1-Weighted MRI Images for VG

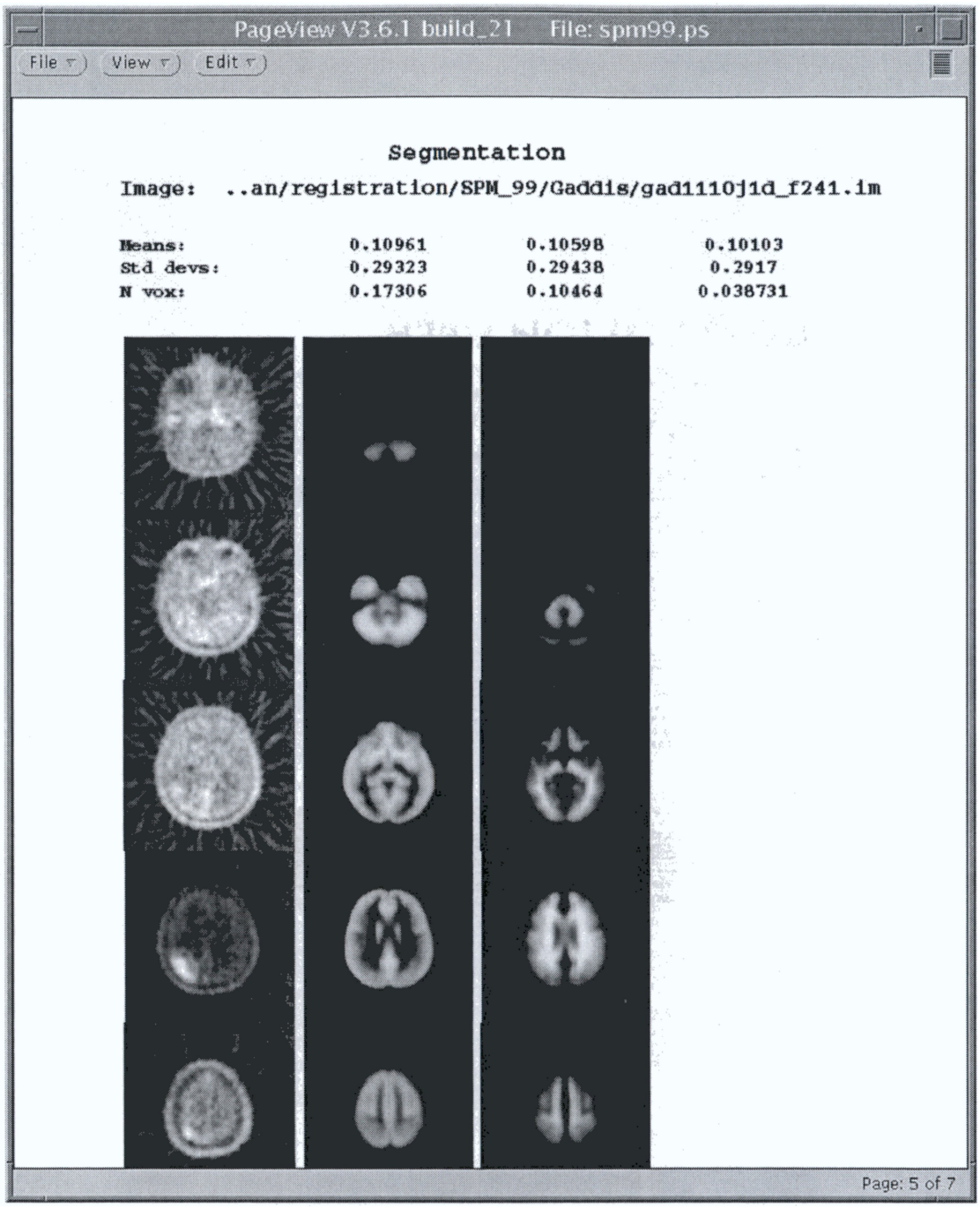


Figure 41: SPM Brain Segmentation for ^{18}F-BPA PET Images for Patient VG

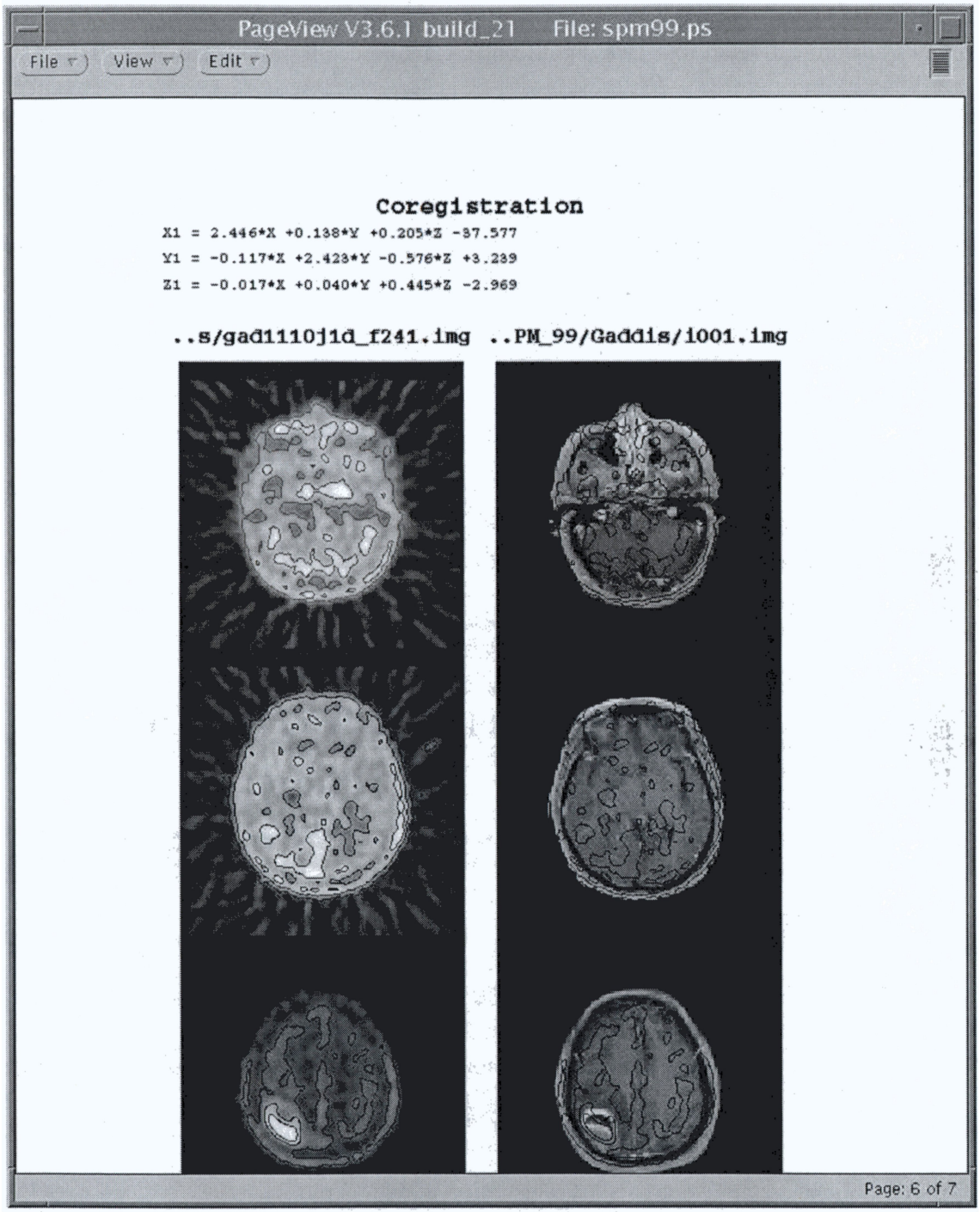


Figure 42: Verification of Image Registeration for VG: Registered PET Image Contours Plotted on Top of MRI Images

Shown in Figure 43 is a MRI image, a registered PET Image, and a superposition of the PET tumor volume on top of the MRI image for patient VG. As can be seen, the MRI provides anatomical detail, but lack significant metabolic detail provided by the PET. One can also note from Figure 43, that what is visualized in an MRI image may not be in the same location as that seen in the ^{18}F-BPA PET image. In addition, what may be seen as the more dense part of the tumor (the white contrasted portion of the tumor) may not correspond with the most metabolically active parts of the tumor.

Shown in Figure 44 are MRI and registered PET image planes taken along the boundary of the tumor along the z axis (i.e. the deeper portion of the tumor along the axial axis). Note that the tumor region in the MRI image is relatively small compared with registered PET image for patient VG. The tumor volume is axially located deeper in a registered PET image than that visualized in an MRI image. In addition, the size of the tumor area is larger and shifted in the registered PET image when compared to the MRI image.

The T1-MRI images are incorporated into **seraModel**. Tumor axial length is measured by visually looking for contrast enhanced regions of the MRI image along the z-axis. The first and the last planes along the z-axis are noted. Using the inter-plane separation of 7.5mm for the MRI images, one is able to determine the axial length of the tumor. A similar approach is applied with the registered ^{18}F-BPA PET images. The results are shown in Table 4. The differences range from 20-150% for the GBM patients, and no difference is seen for the Multiple Melanoma (MM) patient (i.e. CR). The resolution of the PET in the Z-axis is 7.5 mm and the uncertainty due to SPM registration is about 2 mm.

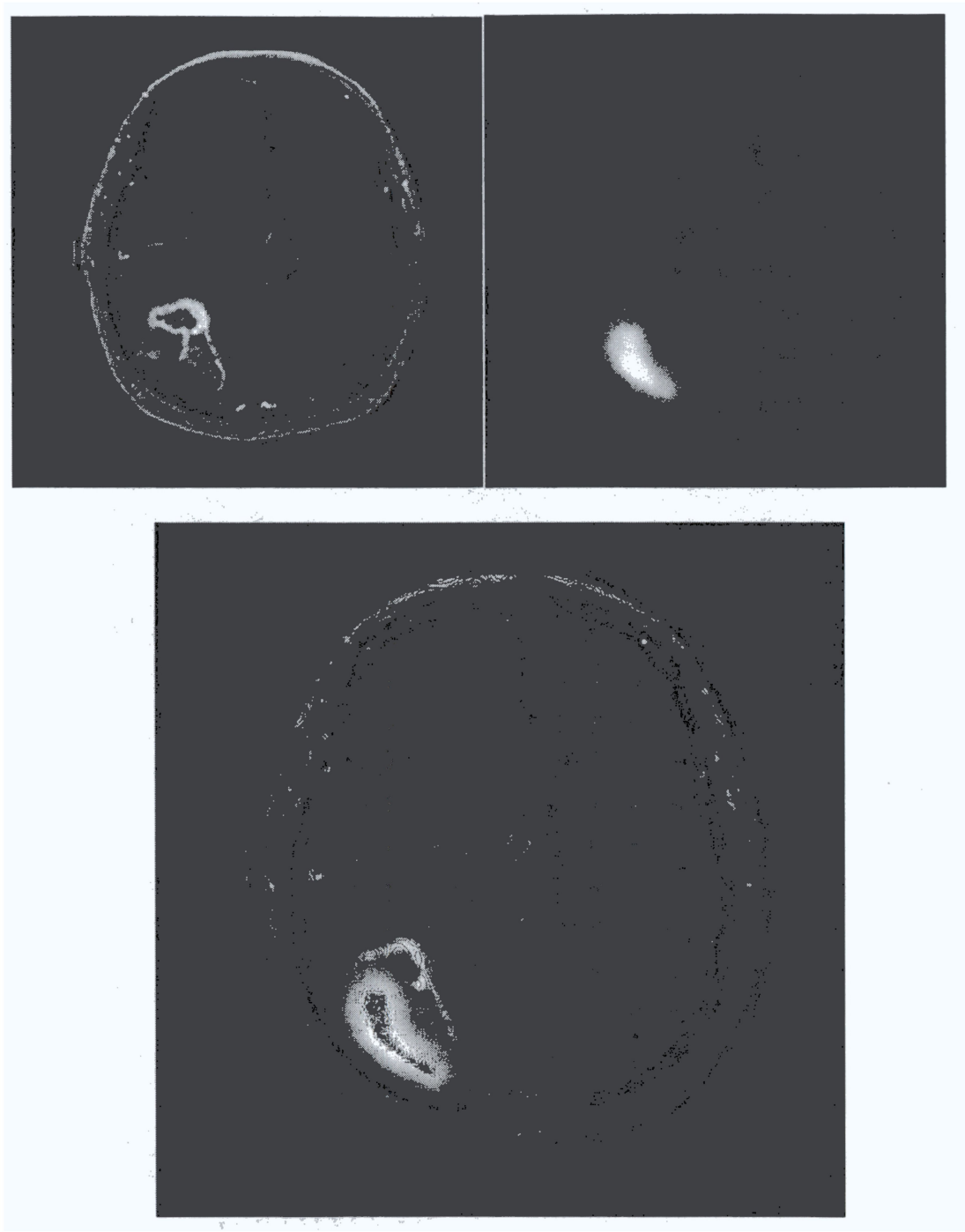


Figure 43: Side by Side Comparison of Registered PET/MRI Tumor Volumes (top images) and a Superimposing of the PET Tumor Volume On Top of MRI Image (bottom image)

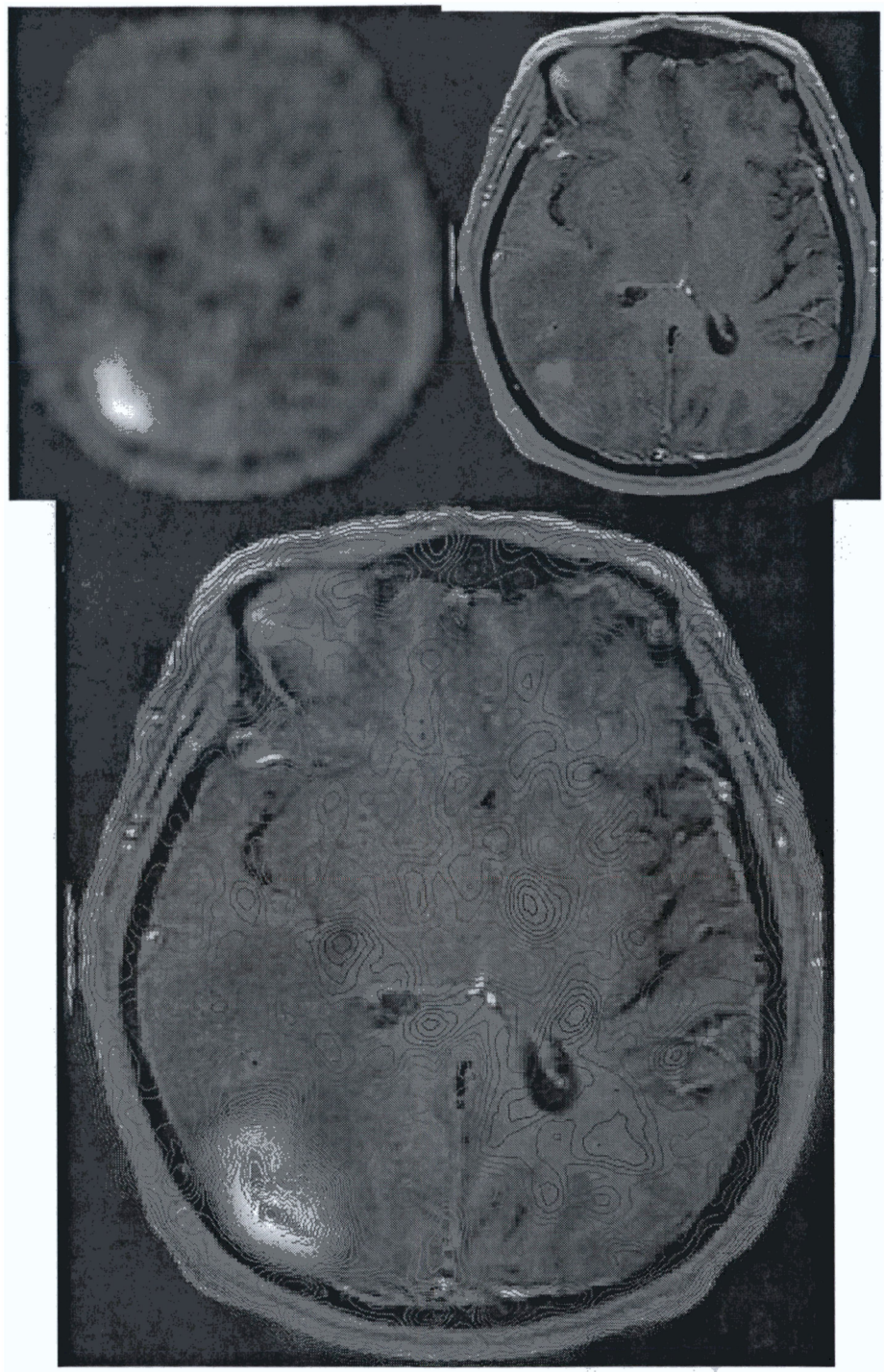


Figure 44: Registered Tumor Volume Differences Along the Z-Axis. A Side by Side Comparison of PET/MRI (top images) and a Superimposing of The PET Tumor Volume on Top of MRI (bottom image)

The differences in the tumor axial extent range from 0.75 cm to 4.5 cm for the GBM patients analyzed in this work. Therefore, tumor axial extent differences larger than a centimeter are greater than the resolution of the PET and the uncertainty due to registration. In all GBM cases, the tumor axial extent is larger in the registered PET images when compared with MRI images. These differences can perhaps be due to the fact that GBM tumors are known to have tumor extensions which are not visible on an MRI image. Generally the tumor volume and the extensions of the tumor fall about 1-3 cm outside the main MRI tumor volume [20]. Some of these tumor extensions may be picked up by the PET imaging resolution if the tumor cells readily uptake ^{18}F-BPA and the density of the tumor cells are such that the PET pixel intensity exceeds the background noise level.

The MRI images are loaded into **seraModel** as described earlier. Within **seraModel**, one can define the tumor volume by painting or picking points around the contrast enhanced regions of the tumor on each plane. These tumor bodies defined on individual planes can then be volume rendered and the total tumor volume can be determined within **seraModel.** A similar approach can be applied with registered ^{18}F-BPA PET images.

Table 4: Comparison of Tumor Axial Extent (in cm) Using Registered ^{18}F-BPA PET and T1-Weigted MRI images

Patient	MRI	PET	% Difference
VG	3.75	4.5	20.00
EG	2.5	3.75	50.00
LB	1.5	3	100.00
MB	3	7.5	150.00
BP	3.75	5.25	40.00
Mean	2.9	4.8	65.52
Standadrd Dev.	0.95	1.73	
CR	2	2	0.00

The results of determining the tumor volume based on MRI images and registered ^{18}F-BPA PET images for the same patient is shown in Table 5. The differences range from 14 cm^3- 93 cm^3 for the GBM patients. The relative percent differences range from 34.04 to 136.36%. These differences are greater than the .31 cm^3 (6.5mmX6.5mmX7.5mm) resolution of the PET image and the .08 cm^3 (2mmX2mmX2mm) resolution of the SPM registration algorithm. In all GBM cases, the tumor volume as defined with **seraModel** using registered PET images is larger than using the T1-MRI images to define the tumor volume. This correlates with the increased tumor axial extent visualized in Figures 44 and 45 when using PET images instead of using T1 MRI images. This finding also correlates with tumor axial extent differences noted in Table 4.

Generally, the target location is defined as the tumor center to ensure that dose contours is centered about the tumor. This centering of the tumor allows for beam placement to ensure adequate coverage of the tumor region while sparing the normal surrounding regions. Shown in Table 6 are the target regions defined using MRI images and registered PET images. The absolute three-dimensional differences in beam placement range from 1.33 cm to 4.29 cm for the GBM patients. Little difference is noted for the Melanoma patient (CR).

Table 5: Comparison of Tumor Volumes (in cm^3) Using Registered ^{18}F-BPA PET and T1-Weigted MRI images

Patient	MRI	PET	% Differenc
VG	47	63	34.04
EG	11	26	136.36
LB	20	34	70.00
MB	70	163	132.86
BP	64	127	98.44
Mean	42.4	82.6	94.81
Standadrd Dev.	26.16	59.97	
CR	9	8	-11.11

Table 6: Comparison of (x,y,z)Target Location (in cm) Using Registered ^{18}F-BPA PET and T1-Weigted MRI images

Patient	MRI	PET	Difference (cm)
VG	(3.72, -2.56, 8.25)	(4.98, -3.94, 9.00)	2.01
EG	(2.75, 2.31, 12)	(1.76, 2.78, 11.25)	1.33
LB	(-1.40, -4.88, 6)	(-1.95, -6.02, 6)	1.27
MB	(2.57, 2.22, 9.75)	(3.38, 1.23, 9.0)	1.48
BP	(3.07, 4.68, 7.5)	(4.81, 3.51, 5.25)	4.29
Mean (Std)			2.08 +/- 1.27
CR	(-5.98, 3.29, 9.75)	(-5.84, 3.65, 9.75)	0.39

The differences greater than approximately one centimeter are larger than the resolution of the PET image and the uncertainties associated with the SPM registration algorithm.

Besides using ^{18}F-BPA PET images to define the tumor volumes, one can also use the boron biodistribution contained in the PET images to determine the peak-to-normal brain values. The PET pixel intensity along the tumor axial center was shown earlier in Figures 22 and 26. From such plots, one can determine the relative uptake in the tumor volume compared with the normal brain. The results of such an analysis are shown in Table 7. The peak-to-normal brain values range from 2.05 to 3.4 for the last frame data for these patients. The peak-to-normal brain values using the time integrated data ranges from 1.71 to 4 for these patients. These are significantly different than the 3.5 homogeneous scaling factor used in current treatment plans. The post-surgery peak-to-normal brain values for the residual parts of the tumor are 1.5 to 1.7.

Table 7: Tumor Peak-To-Normal Brain (PNB) Value (@ Tumor Axial Center) Using ^{18}F-BPA PET

Patient	PNB	Time Integrated PNB	Post-Surgery Time Integrated PNB
VG	3.4	4	1.70
EG	2.6	2.5	1.50
LB	2.05	2.2	
MB	2.43	2.66	
BP	2.2	1.71	
Mean (Std)	2.53 +/- .83	2.61+/- .86	1.6 +/- .14
CR	3.33	4	

Shown in Table 8 are the doses that would be predicted for the tumor, target, scalp, and normal brain volumes using current treatment planning with T1-MRI images and using a homogeneous scaling factor of 3.5 within the tumor and target volumes. Table 8 presents a summary of the dose volume histograms generated using **seraPlot**. An example of the dose volume histogram was presented earlier in Figure 7. The treatment plans assumed a 50 min irradiation with 2 MW reactor power and a boron concentration of 13 ppm in the blood. As such, the scaling factor of 3.5 is applied within the tumor and target volumes to determine the dose.

Table 8: Gray-Equivalent Mean Dose Administered to Patients with 50 min Irradiation at @ 2MW, assuming 13 ppm blood boron concentration, a Tumor-to-Blood uptake of 3.5, Tumor-to-Scalp uptake of 1.5 using MRI defined Volumes

Patient	Scalp (Range)	Normal Brain (Range)	Tumor (Range)	Target (Range)
VG	0.9 (0.1, 8.9)	1.9 (0.2, 8.2)	38.8 (24.0, 48.4)	28.6 (12.5, 48.0)
EG	1.2 (0, 10.3)	2.4 (0, 9.7)	42.4 (29.6, 51.4)	36.2 (15.8, 52.7)
LB	1.9 (0.2, 10.9)	2.0 (0.3, 10.4)	44.5 (29.1, 54.0)	34.0 (14.0, 53.9)
MD	1.1 (0, 10.17)	1.7 (0, 8.6)	42.8 (24.2, 56.0)	31.2 (11.5, 53.5)
CR	1.2 (0.1, 10.1)	2.0 (0.2, 9.4)	46.6 (40.1, 53.0)	39.0 (25.3, 53.0)
MB	1.8 (0.3, 8.9)	1.2 (0.1, 10.37)	34.4 (16.1, 52.5)	26.9 (8.6, 53.3)
BP	1.4 (0.1, 10.4)	1.5 (0.2, 8.8)	33.5 (14.9, 52.7)	23.5 (7.0, 53.1)
Mean	1.36	2.12	40.57	31.44

One can take the doses presented in Table 8 and linearly scale them based on the PET determined peak-to-normal brain value instead of the 3.5 scaling factor. The resulting doses are shown in Table 9. The doses to the tumor and target volumes are significantly different in the two approaches. Table 8 and Table 9 both use the scaling factor and the peak-to-normal brain factor, respectively, in a homogeneous manner to scale the dose. No consideration is given to the shape of the boron biodistribution as visualized in a PET image, as shown in Figures 22 and 26.

Shown in Figure 45 are the boron dose rate contours as determined from normal treatment planning using SERA in which the tumor volumes are defined using MRI images. No scaling of the boron dose within the tumor and target regions is implemented in the dose rate contours shown in Figure 45. A three-dimensional view of the boron dose rate contours shown in Figure 45 is shown in Figure 46. The boron contours in Figures 45 and 46 are uniform in shape and correspond to the assumption of a uniform boron biodistribution throughout all the volumes. The units of the dose rate in Figure 46 are cGy/Mw-min/ppm since no considerations regarding the total irradiation time and the boron scaling factor have been made. Also, note that the maximum beam flux is expected to be about 2.5 cm into the brain. Thus the 90 % contours are generally located closer to the surface of the brain as opposed to being further in the brain.

Table 9: Gray-Equivalent Mean Dose Administered to Patients with 50 min Irradiation at @ 2MW, assuming 13 ppm blood boron concentration, a PET based Tumor-to-Normal Brain, Tumor-to-Scalp uptake of 1.5 using MRI defined Volumes

Patient	Scalp (Range)	Normal Brain (Range)	Tumor (Range)	Target (Range)
VG	0.87 (0.097, 8.74)	1.84 (0.19, 8.1)	38.66 (23.89, 48.28)	28.46 (12.43, 47.79)
EG	.89 (0, 7.65)	1.78 (0, 7.2)	31.49 (21.99, 38.18)	26.89 (11.73, 39.14)
LB	1.11 (0.11, 6.38)	1.17 (0.17, 6.09)	26.5 (17.04, 31.62)	19.9 (8.0, 31.57)
CR	1.14 (0.95, 9.6)	1.9 (0.19, 8.94)	44.3 (38.15, 50.42)	37.1 (24.1, 50.43)
MB	1.25 (0.21, 6.18)	.83 (0.07, 7.20)	23.88 (11.18, 36.45)	18.68 (5.97, 37)
BP	.88 (0.06, 6.53)	.94 (0.13, 5.53)	21.05 (9.36, 33.13)	14.77 (4.4, 33.38)

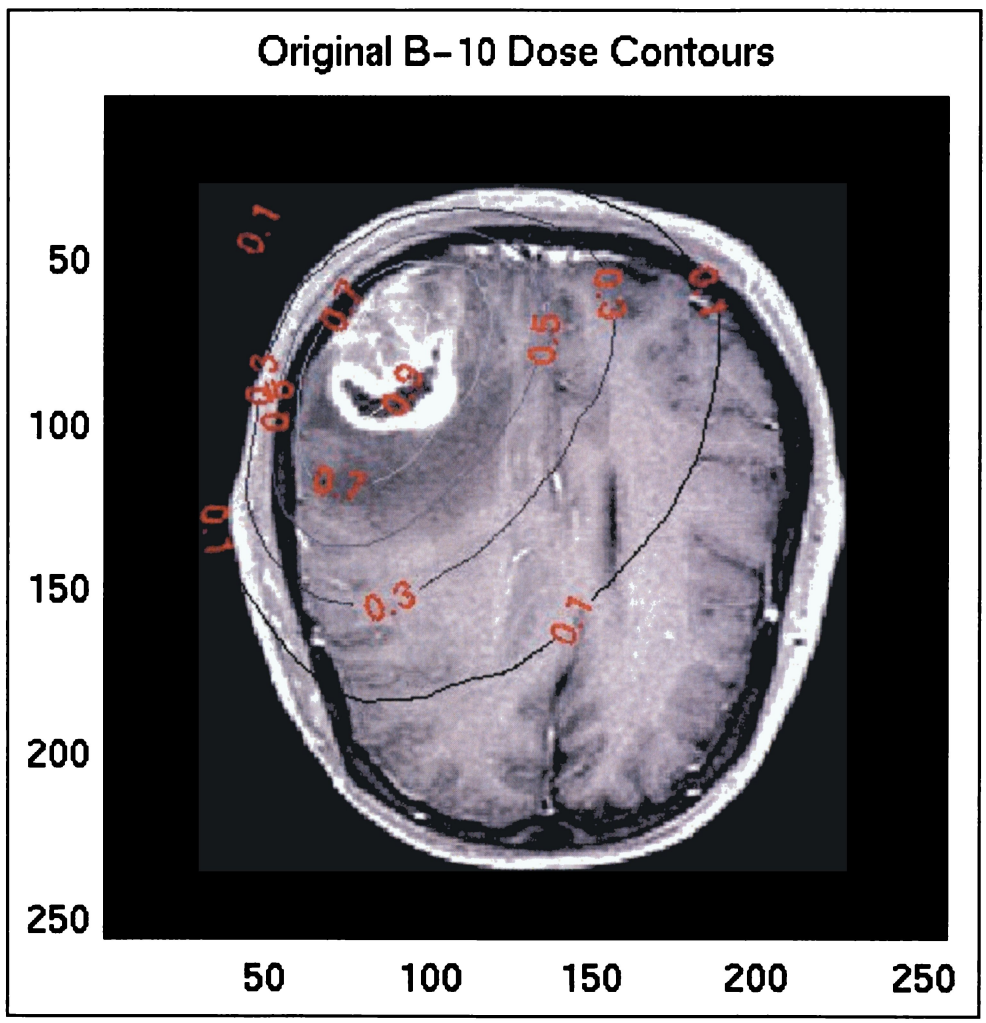


Figure 45: Original Boron Dose Contours

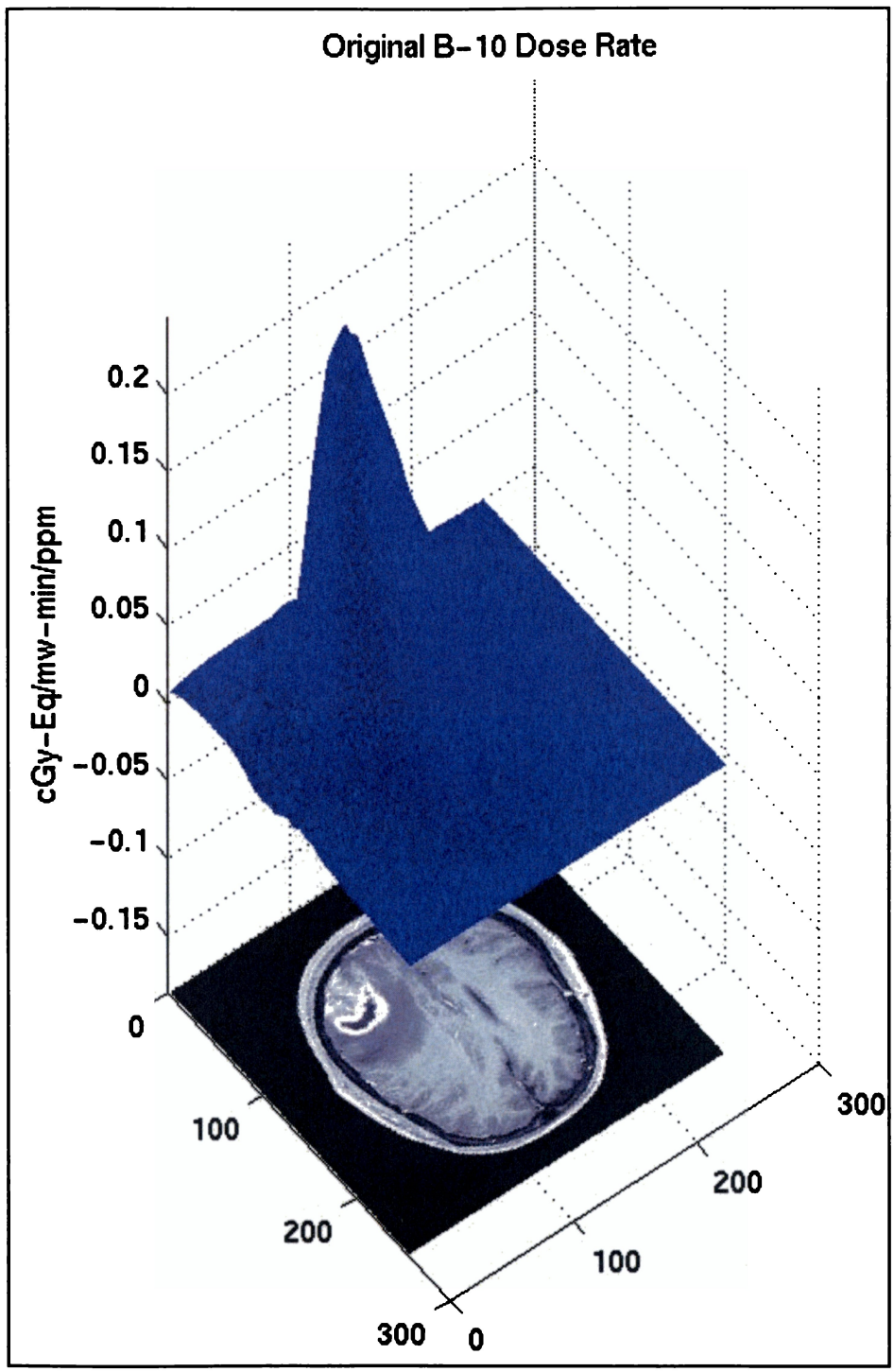


Figure 46: A Three Dimensional View of The Original Boron Dose Rate As Calculated by SERA

If a linear homogeneous scaling factor of 3.5 is applied within the tumor and target volumes to scale the boron component of the dose, then the resulting dose contours are shown in Figure 47. Figure 47 assumes that the boron concentration within the tumor and target volumes is 45.5 ppm and is 13 ppm in the normal brain regions, as defined within SERA. As such the dose rate in Figure 46 is scaled homogeneously using a step input scaling of 3.5 within the tumor and target volumes to account for the boron concentration scaling factor. This results in the 90% and 70% contours staying in essentially the same location. However, the 50% and 30% contours tend to overlap one another along the tumor/normal brain boundaries as a result of applying a constant input function of 3.5 to scale the dose. A three dimensional view of this type of homogeneous scaling is shown in Figure 48.

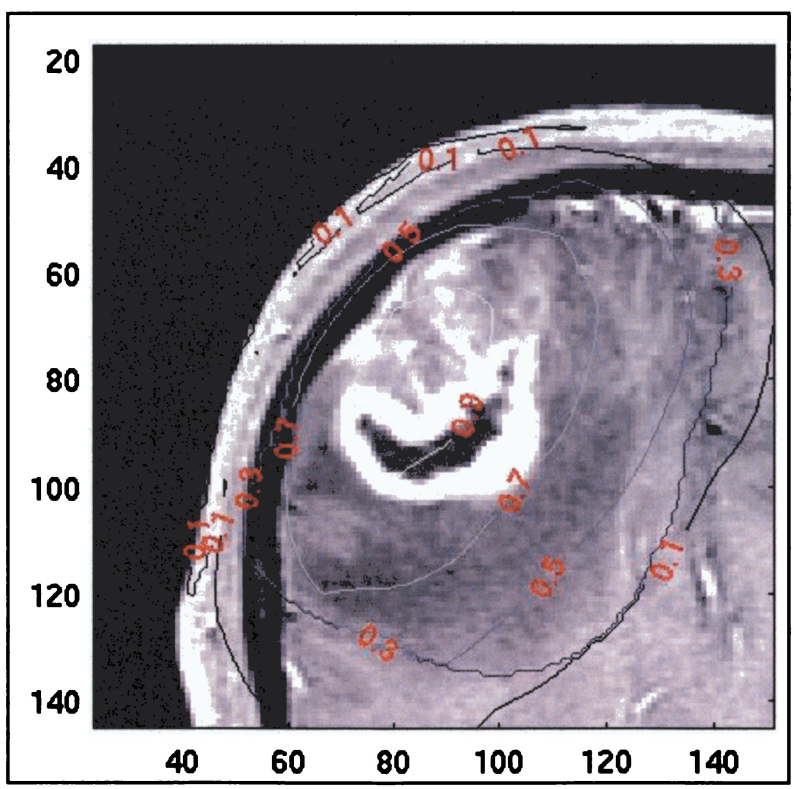


Figure 47: Boron Dose (or Dose Rate) Contours After Applying the Homogeneous Scaling Assumption

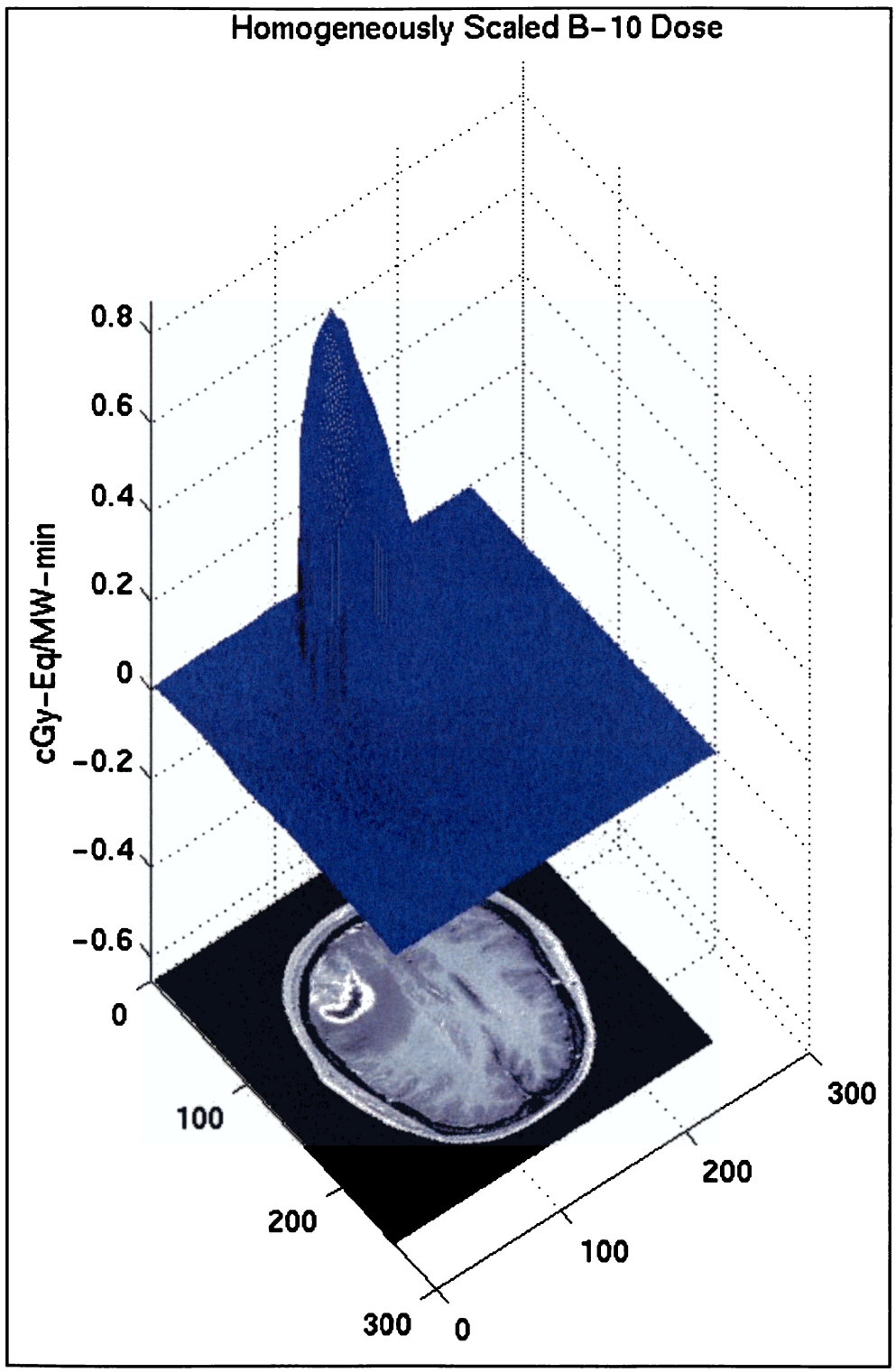


Figure 48: Three Dimensional View of Homogeneously Scaled Boron Dose Rate

However if one uses the registered ^{18}F-BPA PET images within treatment planning the boron dose rate contours would be as those shown in Figure 49. Note that the 90%, 70%, 50%, 30%, and the 10% dose contours have all shifted towards the metabolically active parts of the tumor, as is visualized in a registered ^{18}F-BPA PET image. The contours are narrower and more focused on the metabolically active parts of the tumor, unlike the homogeneously scaled contours shown in Figure 47. The differences in the two contours is attributed to the *in-vivo* boron biodistribution as determined from a registered ^{18}F-BPA PET image. A three dimensional view of the boron dose rate which is PET adjusted is shown in Figure 50.

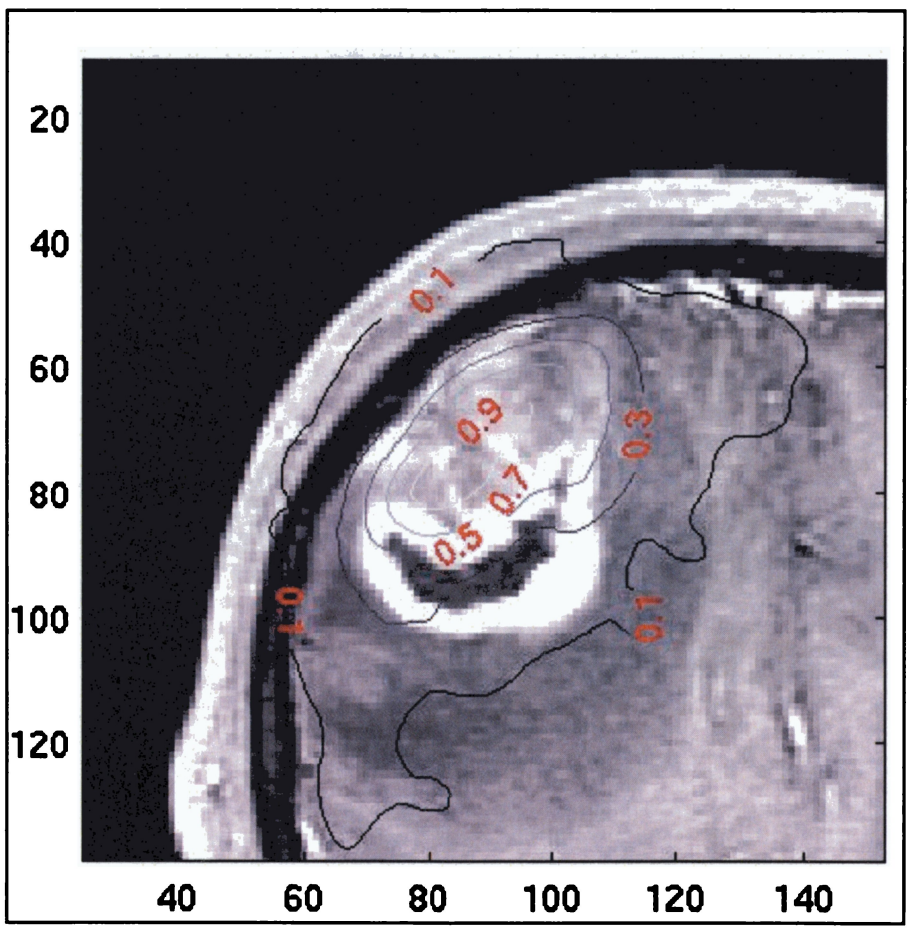


Figure 49: Registered ^{18}F-BPA PET Adjusted Boron Dose (or Dose Rate) Contours

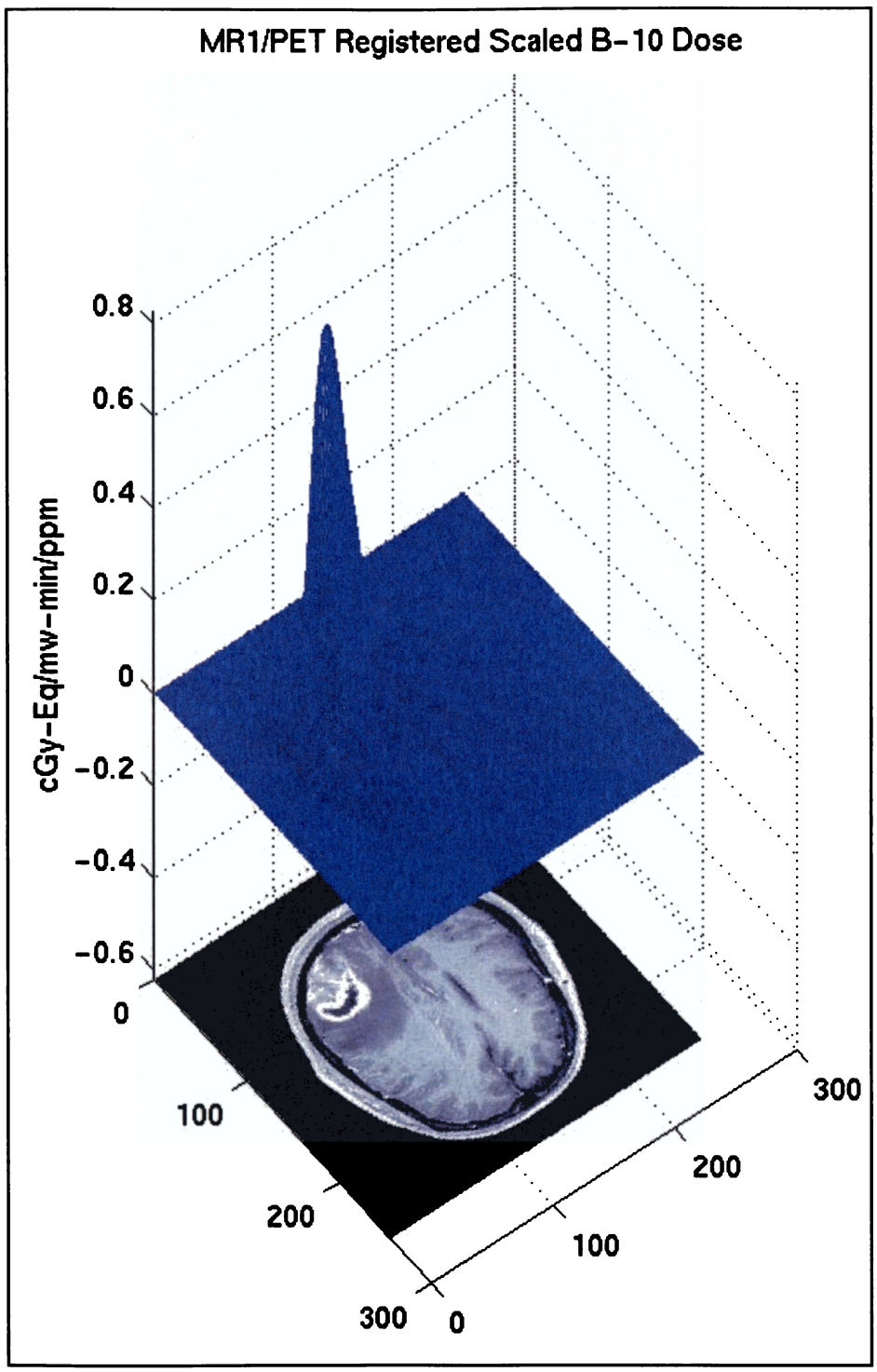


Figure 50: A Three Dimensional View of the Adjusted Boron Dose Rate (or Dose) Based on Registered ^{18}F-BPA PET Images

Note from Figure 48 that the scaling of the boron dose at the boundaries of the tumor and the target volumes leads to a sharp cutoff of the contours along the boundaries. This is evident in Figure 47 and 48 for the 30% and the 50% dose contour lines. This line corresponds with the 2 cm target/normal brain boundary that surrounds the tumor volume. Within this boundary, the dose is homogeneously scaled by a factor of 3.5. This sharp discontinuity along the target/normal brain boundary was also demonstrated earlier in Figure 8 with the beam centerline dose depth profile. The results presented in Figures 46 and 47 are for patient VG whose peak-to-normal brain value was 3.4 as shown in Table 7. The 3.4 peak-to-normal brain value is close to the 3.5 scaling value. However, note that the PET pixel intensity for this patient, as shown earlier in Figures 22 and 26 is not homogeneous in the tumor and target regions. For this patient data, the PET biodistribution demonstrated an increased uptake in the center/metabolic regions of the tumor and a tapering off of the boron biodistribution in nearby regions. Thus, if one takes the registered PET biodistribution data for this patient and integrates this biodistribution information together with the original boron dose contours as shown in Figure 46, contours shapes similar to those presented in Figures 49 and 50 result. The boron contours in Figure 49 are representative of the metabolic activity of the tumor and the biodistribution of boron within the tumor regions, unlike the contours shown in Figure 46.

The significant changing of the dose contours when using the PET boron biodistribution within treatment planning are also noted for other patients. The 3D dose images and the dose contours for all patients are included in the Appendix (Appendix T).

XI. Conclusions

Use of registered ^{18}F-BPA PET images can significantly improve treatment planning for BNCT. These images can provide much needed boron biodistribution information along with better visualization of the metabolically active regions of the tumor. This information provided by the PET image is not used in current treatment planning for BNCT, which uses MRI images to reconstruct the tumor volume and to predict the dose using the homogeneous assumption. The heterogeneous boron biodistribution information obtained from PET images better predicts the average boron component of the dose to the tumor volume than the value obtained based on the homogeneous assumption. Computational software and the methodology to implement such a task have been discussed in this work. Results show remarkable difference between current treatment planning and improved treatment planning using the registered ^{18}F-BPA PET images.

Better visualization of the metabolically active regions of the tumor, notable differences in the tumor axial extent, and differences in tumor volume definitions between the T1-MRI images and the registered PET images can lead to reorientation of the beam. In addition to reorienting of the beam based on anatomical and metabolic differences in the two imaging modalities, the ^{18}F-BPA PET boron biodistribution can also lead to significant changes in the dose contours relative to those obtained based on the homogeneous assumption. Better beam alignment and better dose contours, which result from using the PET boron biodistribution data, can significantly improve future BNCT treatment planning.

XII. Future Work

Future work should address several aspects of using ^{18}F-BPA PET images within treatment planning: (1) additional patients and PET studies should be utilized in treatment planning and a comparison of the tumor volume differences and the peak-to-normal brain differences should be further assessed across many patients; (2) the peak-to-normal brain value and the tapering of the boron concentration as visualized in PET should be correlated with histopathological sampling of those regions. The uncertainties in the histopathological sampling used to derive the 3.5 scaling ratio should be correlated with the PET determined boron biodistribution of those regions; (3) compartment modeling should be coupled with with treatment planning using ^{18}F-BPA PET images. The comparatmental information provided by PET can perhaps lead to optimization of the boron concentration within the tumor cells and this can be coupled with treatment planning to better predict the actual dose delievered to the tumor cells post-surgically.

Refernces

References

1. R. F. Barth, A. H. Soloway, R. G., Fairchild, "Boron Neutron Capture Therapy for Cancer," *Scientif. Am.*, 100 (1990).

2. M. Frederick Hawthorne, "New Horizons for Therapy Based on the Boron Neutron Capture Reaction," Molecular Medicine Today, 1998, 4:4:174-181

3. Sandberg-Wollheim M, Malmstrom P, Stromblad LG, et al. A randomized study of chemotherapy with procarbazin, vincristine, and lomustine with and without radiation therapy for astrocytoma grades 3 and/or 4. Cancer 1991; 68:22-29.

4. Jansen, E.P.M, Dewit, L.G.H, Herk, M.V, Bartelink H, Target volumes in radiotherapy for high-grade malignant glioma of the brain. Radiotherapy and Oncology. 2000. Vol 56. 151-156.

5. Locher, G.L. Biological effects and therapeutic possibilities of neutrons. Am. J. Roent. 36:1-13; 1936.

6. Slatkin, D.N. A history of boron neutron capture therapy of brain tumors – Postulation of a brain radiation dose tolerance limit. Brain. 114: 1609-1629, 1991.

7. Barth R.F. Soloway, A.H, Fairchild RG, Brugger RM: Boron neutron capture therapy for cancer. Realities and prospects. Cancer 70:2995-3007, 1992.

8. Coderre, Jeffrey A, Elowitz E.H., Chadha, M., Bergland, R., Capala, J., Joel, D. Liu, H., Slatkin, D., Chanana, A. Boron Neutron Capture Therapy for Glioblastoma Multiforme Using p-bronophenylalanine and Epithermal Nuetrons: Trial Design and Early Clinical Results. Journal of Neuro-Oncology 33: 141-152. 1997.

9. Coderre JA, Chanana AD, Joel DD. Elowitz EH, Micca PL, Nawrocky MM, Chadha M, Gebbers JO, Shady M, Peress NS, Slatkin DN. "Biodistribution of boronophenylalanine in patients with gliogblastoma multiforme: boron concentration correlates with tumor cellularity, " Radiation Research. 1998. Feb 149:2 163-170.

10. Elowitz, EH; Bergland RM; Coderre MA; Joel DD; Chadha M; Chanana AD; Biodistribution of p-boronophenylalanine in patients with GBM for use in boron neutron capture therapy. Neurosurgery. 1998. Mar. v42 (3). P 463-69.

11. Sweet, W.H., Javid, M., "The possible use of neutron-capturing isotopes such as boron-10 in the treatment of neoplasms," I. Intracaranial Tumors, J. Neurosurg., 9:200-209, (1952).

12. Barth R.F., Soloway A.H. and Brugger R. M. (1996). Boron Neutron Capture Therapy of brain tumors: past history, current status, and future potential. Cancer Invest., 14:534-550.

13. Chadha, Manjeet. Jacek Capala. Jeffrey A. Coderre, Eric H. Elowitz, JunIchi Iwali, Darrel D. Joel, HB Liu, L Wielopolski, A. D. Chanana. Boron Neutron Capture Therapy (BNCT) for

Glioblastoma Multiforme (GBM) using the Epithermal Beam at the Brookhaven National Lab. Int. J. Radiation Oncology. Vol 40. No. 4. pp 829-834. 1998.

14. Lewis, E.E., W.F. Miller, Jr. Computational Methods of Neutron Transport. American Nuclear Society. 1993..

15. Chilton, Arthur B., J. Kenneth Shultis Richard E. Faw, Principles of Radiation Shielding. Prentice Hall. NJ. 1984.

16. Nigg, David. W. Floyd J. Wheeler, Daniel E. Wessol. Jasek Capala, and Manjeet Chadha. Journal of Neuro-Oncology 33:93-104, 1997.

17. Goorley, T., F. Wheeler, J. Capala, W.S. Kiger III, M. Palmer, and R. Zamenhof. "A Comparison of Two BNCT Treatment Planning Programs: MacNCTPlan and BNCT_RTPE," in Frontiers in Neutron Capture Therapy, Hawthorne, et.al., editors, Kluwer Academic/Plenum Publishers, New York, 2001, Nov 1. Pp. 207, 217.

18. Stead Kiger. Personal Communications. Massachusetts Institute of Technology. Dec. 1998.

19. Coderre, J.A., Makar, M.S., Micca, P.L., et al. Derivations of Relative Biological Effectiveness for the high LET radiations produced during boron neutron capture irradiations of the 9L rat gliosarcoma *in-vitro* and *in-vivo*. Int. J. Radiation , Biology, Oncology and Physics. 27: 1121-1129; 1993.

20. Jansen, Edwin P.M., Luc G. W. Dewit, Marcel Van Herk, Harry Bartelink. Target Volumes in Radiotherapy for High-Grade Malignant Glioma of the Brain. Radiotherapy and Oncology. Vol56. 2000. P151-156.

21. Day, Charles. Alpha radiation can damage DNA even when it misses the cell nucleus. Physics Today. September 1999. page 19-20.

22. Kabalka, George W., Smith, G.T., Dyke, J.P, Reid, W.S., Longford, D, Roberts, T.G., Reddy, N.K, Bunocore, E., Hubner, K., Evaluation of Flourine F-18-BPA Fructose for Boron Neutron Capture Therapy Treatment Planning. The Journal of Nuclear Medicine. Vol. 38. No 11. Nov 1997.

23. Gross, Markus W. Wolfgang A. Weber, Horst Jurgen Feldmann, Peter Bartenstein, Markus Schwaiger, Micahel Molls. The Value of F-18-Flurodeoxyglucose PET for the 3-D radiation Treatment Planning of Malignant Gliomas. Int. J. Radiation Oncology Biol. Phys. Vol. 41. No. 5. Pp 989-995, 1998.

24. Bennet, B.D., Mumford-Zisk, J., Coderre, J.A., Morrison, George H. Subcellular Localization of *p*-Boronophenylalanine-Deliverd Boron-10 in the Rat 9L Gliosarcoma: Cryogenic Preparation *In Vitro* and *In-Vivo*. Radiation Research. 1994. Vol 140. P 72-78.

25. Ye Sung-Joon. Boron self-shielding effects on dose delivery of neutron capture therapy using epithermal beam and boronophenylalanine. Medical Physics. 26. (11). Nov. 1999.

26. Wright, G.A. Magnetic Resonance Imaging. IEEE Signal Processing Magazine. Jan. 1997. p56-66.

27. "PET proves best for revealing cancer spread" Nuclear News. October 2001. p75.

28. "PET alters treatment in breast cancer patients" Nuclear News. October 2001. p75.

29. "PET changes lung cancer staging, treatment" Nuclear News. December 2001. p56.

30. Jacek Kapala-Personal communication/discussion in 2000 during his visit to the University of Tennessee, Knoxville.

31. A.C. Kak, M. Slaney, Principles of Computerized Tomographic Imaging. New York: IEEE Press, 1988.

32. Gregor, Jens, Dean A. Huff, A computational study of the focus-of-attention EM-ML algorithm for PET reconstruction. Parallel Computing vol 24. 1998. 1481-1497.

33. Imahori, Y., *et al*, Journal of Nuclear Medicine, 1998, p325.e, Vol. 38, No. 11, (November 1997)

34. Jain, A. K. "Fundamentals of Digital Image Processing", p. 439.

35. P.A. Toft, "The Radon Transform, Theory and Implementation" (unpublished dissertation) p. 199.

36. T. Goorley, F. Wheeler, J. Capala, W.S. Kiger III, M. Palerm, and R. Zamenhof. "A Comparison of Two BNCT Treatment Planning Programs: MacNCTPlan and BNCT_RTPE," Frontiers in Neutron Capture Therapy, Hawthorne, et. al, editors, Kluwer Academic/Plenum Publishers, New York, 2001, Vol 1. pp. 207-217.

37. Chuck Wimperle, Personal Communications and Email. Jan. 2nd 2002.

38. Wessol, Dan. Cohen, Matt, Harkin, G., Rossmeir, M., Wemple, C. Wheeler, F. INEEL/EXT-99-00766: SERA Workshop Lab Manual. Aug 4-6, 1999.

39. **http://www.cs.montana.edu/~bnct/**

40. G. Q. Maguire, Jr. and M. E. Noz, QSH Reference Manual, Columbia University, Department of Computer Science; New York University, Department of Radiology, 1990.

41. B. S. Baxter, L. Hitchner, and G. Q. Maguire, Jr., A Standard Format for Digital Image Exchange, AAPM Report No. 10, American Institute for Physics, 1982.

42. M. E. Noz and G. Q. Maguire, Jr., QSH: A Minimal but Highly PorTable Image Display and Handling Toolkit, Computer Methods and Programs in Biomedicine, Vol. 27, 1988, pp. 229-240.

43. http://www.psyc.nott.ac.uk/staff/cr1/mricro.html

44. Bond, Hedley. Image Processing Laboratory at University of Tennessee Medical Center. Personal Communications. 1999.

45. Mohammad K. Khan, M.S. "Use of Positron Emission Tomography to Improve Radiation Dosimety for Boron Neutron Capture Therapy" A Thesis Presented for the Master of Science Degree. University of Tennessee, Knoxville. May, 1999.

46. Brown, Lisa G. Image Registration Techniques. ACM Computing Surveys. Vol. 24. No. 4. Dec. 1992.

47. Woods, R.P., Mazziotta, J.C, and Cherry, S. R. (1993). MRI-PET registration with automated algorithm. Journal of Computer Assisted Tomography, 17(4):536-546.

48. Friston KJ, Ashburner J, Poline JB, Frith CD, Heather JD, Frackowiak RSJ (1995). Spatial Registration and Normalization of Images *Human Brain Mapping* **2**:165-189

49. Ashburner J, Friston KJ (1997) Multimodal Image Coregistration and Partitioning - a Unified Framework. *NeuroImage* 6(3):209-217

50. Kiebel SJ, Ashburner J, Poline J-B, Friston KJ (1997). MRI and PET Coregistration - A Cross Validation of Statistical Parametric Mapping and Automated Image Regression. Neuroimage 5:271-279

51. K.J. Friston. Statistical parametric mapping and other analysis of functional imaging data. In Brain Mapping: The Methods, pages 363-385. Academic Press, 1996.

52. Ashburner, John, Karl J. Friston, Spatial Transformation of Images. The Wellcome Department of Cognitive Neurology, Institute of Neurology, Queen Square, London. WCtN 3BG, UK.

53. Hartigan, J.A. 1975. Clustering Algorithms. New York: John Wiley & Sons, Inc. Pages 113-129.

54. Kiebel SJ, Ashburner J, Poline J-B, Friston KJ (1997) *Neuroimage* **5**:271-279

55. Charles A. Wemple at INEEL. Email Discussion January 2, 2001.

56. Casey, Mike. Personal Communications with Mike Casey at CTI. Jan. 2002.

57. Heusmann, Ron. Physics, Medicine, and Biology. Vol. 29. no. 5. 1984. p 543-552.

58. Gregor, Jens. Personal Communications with Proj. Jens Gregor in Computer Science Dept. at UTK, Jan. 2002.

59. McKeever, P.E., Hood, T.W., Varani, J. et.al, Products of Cells Cultured from Gliomas. Cytology and Morphometry of Two Cell Types cultured from Glioma, *JNCI* 1987: 78, 1:75-84

Appendixes

A: EXAMPLE OF SERA MONTE CARLO INPUT

```
G1_PET_Presurgery
    1000     2000       1.000000E-02
   -4.1700000E+00     -3.8500000E+00      5.0600000E+00
    1.8500000E+01     -1.1000000E+02      1.4500000E+02
/home/mokhan/ST/Results/cg.geom
/home/mokhan/VG/G1_PET_Presurgery.uv
     30       3
     32      72      94
   -1.50000000E+01    -1.50000000E+01    -1.50000000E+01
    1.00000000E+00       0
none
/home/mokhan/VG/G1_PET_sera.rst
/home/mokhan/ST/Results/BS.source
      1       0
NFGD            05_Apr_2001
$SERA_HOME/Resources/SeraMC/seraMC.mat
$SERA_HOME/Resources/SeraMC/seraMC.sigma
 521  508  513  512  505     0     0 6.0143E-08 6.3150E-02 7.9110E-04 7.2700E-03 2.6800E-02 0.0000E+00
0.0000E+00
run_seraMC

iop
 3 APERTURE BUFFER
 0.50
edit_dir
#patient John Doe
#date Jan_07_2001
#beam BMRR_12_CM
 mw_min 1.0
 b10_blood 1.0
 refvol 1.0 0
 ref_reg brain tumor target
 ottocon 40 40 scalp
 delta 0.25
 eps 0.05
#dose-volume histogram for brain
 in_reg brain target
 b10_blood 13
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
#dose-volume histogram for tumor
 in_reg tumor target
 b10_blood 13
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
#dose-volume histogram for scalp
 in_reg scalp
 b10_blood 19
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
```

B : MCNP 4B CODE TO VERIFY LINEAR ASSUMPTION

```
c  Cells
c  Cells 1 = water and Boron10 in the cube 20x20x20
c        2 = tumor cell(water &Boron 10) 5x5x5 cell located in the
middle of the big cube but 2cm from the edge
c        3 = rest of the world
c        4 = void cell inside the cylinder
c cell# - material# -density (if its 1 write-1) -position of the cell(+
above plane,-below plane)
1 1 -1 6 -9 1 -4 10 -13  (-7:8:-2:3:-11:12)
2 2 -1 7 -8 2 -3 11 -12
3 0  -6:9:-1:4:-10:13
4 0 -14

c Surfaces   1-5 = the y slices
c           6-9 = the x slices
c          10-13 = the vertical slices
c           14=cylinder
1 py -4.5
2 py -2.5
3 py 2.5
4 py 15.5
5 py -24.5
6 px -10
7 px -2.5
8 px 2.5
9 px 10
10 pz -10
11 pz -2.5
12 pz 2.5
13 pz 10
14 cy 4

mode n
m1 8016.50c -33.333333 &
   1001.50c -66.666666 &
   5010.50c -.000001
m2 8016.50c -33.33332667 &
   1001.50c -66.66665333 &
   5010.50c -.00002
m3 5010.50c -100.00
 c FMn c m R
c c=N is the number of densities,  for
water=(1g/cm3)*(.6023*E24/18g.mole)*(E-24 cm2/bn)=.0334 atoms/bn/cm
c c-B10=.0334*E-6
c FMn, FM4 FM calculate the reaction rate see mcnp manual 3-74 and
n=4(flux averaged over a cell)  c 2=material#, -2 absrption cross
section
F5:N 0 1 0 .9
F15:N 0 -1 0 .9
```

```
F25:N 0 -3.5 0 .9
F35:N 0 3.5 0 .9
SDEF SUR=5 ccc=4  DIR=1 ARA=50.2655 ERG=D9
SI9 H        1.0000E-15 1.0000E-07  4.1400E-07 8.7642E-07&
            1.8553E-06 5.0435E-06 1.0670E-05 3.7267E-05&
            1.0130E-04  2.1445E-04 4.5400E-04 1.5846E-03 &
            3.3456E-03 7.1017E-03 1.5034E-02 2.1800E-02 &
            2.4176E-02 2.6058E-02 3.1828E-02 4.0868E-02 &
            6.7379E-02 1.1109E-01 1.8316E-01  2.9720E-01&
            3.6880E-01 4.9787E-01 6.0810E-01 7.4274E-01&
            8.2085E-01 1.0026E00   1.3534E00  1.6531E00 &
            1.9205E00  2.2313E00  2.3457E00 2.3653E00 &
            2.4660E00   2.7250E00  3.0119E00  3.6788E00 &
            4.9659E00  6.0653E00 7.4682E00  8.6070E00 &
            1.0000E01  1.2220E01  1.4190E01  1.7330E01
SP9     0          0.0037824    0.0076709      0.031638 &
     0.079111     0.16621      0.13363        0.094952&
     0.038046     0.04379      0.036715       0.011644&
     0.002897     0.00082789   0.0035116      0.0034636&
     0.0011275    0.0016118    0.0015223      0.001346&
     0.0018339    0.0011668    0.00074724     0.00076219&
     0.0002533    0.00017918   0.00011774     0.000097977&
     0.000041827  0.0000064622 0.0000036135   0.0000010106&
     0.037475     0.074        0.086655       0.096206&
     0.0023802    0.0013448    0.0027085      0.0031173&
     0.00056937   0.00090662   0.00035775     0.000065409&
     0.00013423   0.000085045  0.000000156    0.000000022654
FM5  3.34E-08 3 -2
FM15 3.34E-08 3 -2
FM25 3.34E-08 3 -2
FM35 3.34E-08 3 -2
imp:n 1 1 0 0
nps 100000
```

C: MRI_IMAGERECONSTRUCTION.M

```
clear all; close all;
%program to reconstruct MRI images using 2D Fast Fourier Transform

% ****************************Part a ************************
path(path, 'E\Moe\codes\project1\Pro1c.m');
load fspec;

figure(1)
mesh(abs(M));
Ms=fftshift(M);
mesh(abs(Ms));


figure(2)
imagesc(log(1+abs(Ms)));   %shows the logarithmic weighted image


% ****************************Part b ****************************

Mc=conj(Ms);    %take complex conjugate of signal

r=size(Ms,1);
c=size(Ms,2);

for i = 1:c;     %Step through all of the columns and take FFT of the
complex signal
    X(:,i)=1/128*fft(Mc(:,i));
end

for j=1:r;      %Step through all of the rows and take FFT of the
intermediate
    X(j,:)=1/128*fft(X(j,:));
end

figure(3)
imshow(abs(X));


%********************Double check on work ****************************
%Double Check on the FFT using the fft2 function
Ms2 = ifft2(Ms);   %the 2d Inverse Fourier Transform built into matlab

figure(4);
imshow(abs(Ms2));
max(max( (abs(Ms2)-abs(X))./abs(Ms2) ) )   %Comparison of my results

with matlab
```

D: FILTER_BACKPROJECTION.M

```
% fbp.m  Code to reconstruct CT/PET images using parallel-beam filtered
% back projection algorithm by using means of convolution and Radon
Inverse;
% Developed by Mohammad K. Khan (C.S. 594 Class).
clear all; close all;

theta = [0:1:179];   %Sequence of angles
Ns = 100;          %Number of rayintegrals computed per view angle;
f = phantom(128);
p = radon(f,theta, Ns);
path(path, 'E:\Moe\codes');

figure(1);
subplot(2,2,1); imshow(f); title('Original Image');
subplot(2,2,2); imshow(p); title('Sinogram');
[f2, h] = fbp_recon(100, p', 'Shepp', 10);   %filtered backprojection
function using convolution method
subplot(2,2,3); plot(h); title('Filter');
subplot(2,2,4); imshow(f2); title('Reconstructed');

figure(2);

p2=iradon(p, theta, 'nearest', 'Shepp-Logan');
imshow(p2); title('Radon Inverse');
```

```
function [f, h] = fbp_recon(n, p, filter, nh);

%n=100; p=p'; filter='ramp'; nh=10;

[nt, ns]=size(p);
ds = ( 2*ceil(128/sqrt(2))+2)/(ns-1);   %distance between ray integrals
assuming original image size to be 128X128
if strcmp(filter, 'ramp');
    h = ramp_filter(nh, ds);
elseif strcmp(filter, 'Shepp')
    h= shepp_logan_filter(nh, ds);
else
    error('unkown filter.');
end


%hh = zeros(1,nt);
%hh(1, (nt/2-nh):(nt/2+nh) )=h;
g=zeros(nt, ns+2*nh);
for i = 1:nt,
    g(i, :)=conv(p(i,:),h);         %convolution of p and h
end
```

```
x=1:100;
y=1:100;

%x=repmat([1:n]-ceil(n/2), n,1); y = rot90(x);

%coordinate matrices
[X,Y]=meshgrid(x,y);
xpr = X-n/2;
ypr= flipud((Y-n/2));

%transformation coefficients
theta = [0:1:179]*(pi/180);
costheta = cos(theta); sintheta=sin(theta);

%projection zero padding
max_s = 2*ceil(n/(ds*sqrt(2)))+2;
lrz = []; rrz = [];
if ns < max_s
    rz = max_s-ns;
    lrz=zeros(1,ceil(rz/2));
    rrz=zeros(1,floor(rz/2));
end

%projection center index
sctr = ceil(ns/2)+length(lrz);

f=zeros(n);

for k = 1:nt,                    %loop over each angle, theta
       q=conv(p(k,:),h);
       q=[lrz q(nh+1:nh+ns) rrz];
       s = xpr*costheta(k)+ypr*sintheta(k);
       as=floor(s./ds);
       f=f+q(as+sctr)+((s-as*ds)./ds).*(q(as+1+sctr)-q(as+sctr));

end
f = f*pi/(nt-1);
return;
```

```
function [h] = ramp_filter(nh, ds);
n = 2*nh+1;
h=zeros(1,n);
k=0;
for i = -nh:nh;
    k=k+1;
    h(1,k) = 1/(4*(ds)^2)*(2*sinc(i)-sinc(i/2)^2);
end

return;
```

```
function [ h]=shepp_logan_filter(nh,ds)
```

```
n = 2*nh+1;
h=zeros(1,n);
k=0;
for i = -nh:nh;
    k=k+1;
    h(1,k) = 2/( pi^2*ds^2*(1-4*i^2) );
end
return;
```

E: TIME_INTEGRATED.M

```
%program to read ECAT 6.4 DATA and to see the header information (works with Gaddis data)
%by Mohammad Khurram Khan (c) copyright 2002
%this shows the PET images (statistically better visulatization of PET data.

clear all; close all; fclose('all'); clear; pack;fclose('all');
filename= 'E:/BNCT_RAW_DATA/Godwin/Pet/Nov97/god1124j1d.img';
%file_system= 'ecat7';
fpname = filename;
%[fid, message] = ecatfile('open', filename, '');


%[ fid, message ]      = ecatfile( 'open', filename, 'ecat7' );
 [ fid, message ]      = ecatfile( 'open', filename, 'ecat6.4' );
 [ mh, message ]       = ecatfile( 'mh', fid );
 selmatrix= [1 1 1 0 0];    %frame number num_planes
[ vol, hd, message ]   = ecatfile( 'read', fid, selmatrix );
% [ vol, hd, message ]   = ecatfile( 'read', fid, selmatrix, segment ); % 3D si$
% [ descr, hd, message ] = ecatfile( 'read_offset', fid, selmatrix );
% [ descr, hd, message ] = ecatfile( 'read_offset', fid, selmatrix, segment ); $
% [ hd, message ]       = ecatfile( 'hd', fid, selmatrix );
% message               = ecatfile( 'writemh', fid, mh );
% message               = ecatfile( 'write', fid, vol, hd, selmatrix );
% message               = ecatfile( 'write', fid, vol, hd, selmatrix, segment $
% message               = ecatfile( 'close', fid );
% message               = ecatfile( 'close', 'all' );
 [ matranges, message ] = ecatfile( 'matranges', fid );

% fclose('all');
hd.mh.num_frames=matranges(2,1);
I=zeros(hd.mh.num_frames, hd.mh.num_planes, 128, 128);
frametime=zeros(1,hd.mh.num_frames);
quantscale=zeros(1,hd.mh.num_frames);

for k=1:matranges(2,1),
  for i = 1:47,
    selmatrix= [k i 1 0 0]   %frame number & plane number
    [vol, hd, message]=ecatfile('read',fid,selmatrix);
    frametime(1,k) = hd.sh.frame_duration;
    quantscale(1,k)= hd.sh.quant_scale;
   I(k, i,1:128, 1:128) = vol(1:128, 1:128);
end

end

Iimage15 = squeeze(I(1:matranges(2,1), 15, :,:));
img15 = 0;
imgt15 = 0;


for i = 1:matranges(2,1),          %shows a particular image plane over all frames
```

```
   img15=squeeze(Iimage15(i, 1:128, 1:128));
   imgt15=squeeze(Iimage15(i, 1:128, 1:128))*quantscale(i)*frametime(i)/sum(frametime)+imgt15;
end


figure(1);
subplot(1,2,1);
imagesc(img15');title('Particular Frame Image');colormap('gray');
subplot(1,2,2);
imagesc(imgt15'); title('Time Integrated'); colormap('gray');
```

F: ECATFILE.M FUNCTION TO READ ECAT6.4/7.0 DATA

```
function [ res1, res2, res3 ] = ecatfile( action, p1, p2, p3, p4, p5 )
% USAGE:
% [ fid, message ]      = ecatfile( 'open', filename, file_system );
% [ fid, message ]      = ecatfile( 'open', filename, 'ecat7' );
% [ fid, message ]      = ecatfile( 'open', filename, 'ecat6.4' );
% [ fid, message ]      = ecatfile( 'open', filename, '' );
% [ fid, message ]      = ecatfile( 'create', filename, mh );
% [ mh, message ]        = ecatfile( 'mh', fid );
% [ vol, hd, message ]   = ecatfile( 'read', fid, selmatrix );
% [ vol, hd, message ]   = ecatfile( 'read', fid, selmatrix, segment ); % 3D sinogram
% [ descr, hd, message ] = ecatfile( 'read_offset', fid, selmatrix );
% [ descr, hd, message ] = ecatfile( 'read_offset', fid, selmatrix, segment ); % 3D sinogram
% [ hd, message ]        = ecatfile( 'hd', fid, selmatrix );
% message               = ecatfile( 'writemh', fid, mh );
% message               = ecatfile( 'write', fid, vol, hd, selmatrix );
% message               = ecatfile( 'write', fid, vol, hd, selmatrix, segment ); % 3D sinogram
% message               = ecatfile( 'close', fid );
% message               = ecatfile( 'close', 'all' );
% [ matranges, message ] = ecatfile( 'matranges', fid );
% [ matlist, matstatus, message ] = ecatfile( 'matlist', fid );
% [ hd, hds, message ]   = ecatfile( 'blankhd', file_system );
% [ mh, hds, message ]   = ecatfile( 'blankhd', 'ecat7' );
% [ mh, hds, message ]   = ecatfile( 'blankhd', 'ecat6.4' );
% [ sh, hds, message ]   = ecatfile( 'blankhd', 'ecat7', file_type );
% [ sh, hds, message ]   = ecatfile( 'blankhd', 'ecat6.4', file_type );
%
% The user has the responsibility that the written headers are compatible
%  with the file and data structure
% Revision: April 24, 2001.
% ecatfile has been tested under MATLAB5.3. It has not been tested thoroughly under MATLAB6.

% Copyright (C) 1999, 2000, 2001 by Flemming Hermansen.

% Disclaimer, copyright & licencing
% ecatfile is free but copyright software, distributed under the
% terms of the GNU General Public Licence as published by the Free
% Software Foundation (either version 2, or
% at your option, any later version). Further details on "copyleft" can
% be found at http://www.gnu.org/copyleft/. In particular, ecatfile is
% supplied as is. No formal support or maintenance is provided or
% implied. The author will have no responsibility of the use and possible errors.

% Known problem with opened files:
% The ecatfile.m reads all headers and all the index blocks when the ECAT file is opened.
% Further calls to open will not reread the headers and index blocks if the file is
% already open. Each call to open or create will generate different fid values for the
% same file. The headers and the index blocks will only be discarded, when all fid's have
% been closed (or when 'close all' is issued). This will give problems if the file is
% modified outside the Matlab process, because the headers and index blocks in the Maltab
```

```
% process then don't correspond to those in the physical file. This may also happen, if a
% file is renamed or copied. In this situation the user may call
%    message = ecatfile( 'close', 'all' )
% to make sure that ecatfile.m will reread the headers and index blocks. Usually, each
% program should close a file after using it. Still, files may be left opened, if a
% program crashes before closing the file. A subroutine that is used to read portions of a
% file will thus open and close the file many times. This will impose a high overhead
% because the headers and index blocks will be read repeatedly. This overhead may be
% prevented if the calling program, itself, opens the file before calling the
% subroutine, and closes the file when it has performed the last call to the subroutine.

% UNRESOLVED PROBLEM: I have not seen a definite report from CTI stating all possible
% formats. There may, therefore, be errors due to misunderstandings and ignorance of
% certain formats.
% Especially, I do not know the machine format of the file. I have assumed that 6.4
% files can be read by using machineformat 'vaxg' in the fopen routine. Likewise, I
% assume that ecat7 files can be opened using 'ieee-be'. I assume that the entire file
% be read by the same machineformat.

% The routines have been tested, and they can read and write the
% following ecat6.4 files types: 1:*.scn, 2:*.img, 3:*.atn, 4:*.nrm
% and the following ecat7 files types: 3:*.a, 7:*.v, 11:*.S, 14:*.S
% It can read the header from the following file type: 13:*.N
% It should also be able to read other headers (see the tables below), but they have not been tested.
%
% The written files have been tested binarily, and there are some differences:
% The CTI generated files do not contain a truly doubly linked list of index blocks, as index block 2
% points back to block 0.
% Floating point numbers are not reproduced binarily exact, but the read value is the same,
% except for values very close to zero. (assumedly a problem with the representation of
% underflow, or illegal values)
% The last block in CTI generated ecat7 3D sinograms are filled with non-zeros.
% The last block in CTI generated ecat7 3D attenuation is truncated.

%NOT IMPLEMENTED and known errors:
% Problem not yet solved (requires a call to the c-routine stat in stat.h):
% A file may be opened twice or more times at the same time. The filename must,
% however, be spelled exactly the same way. Otherwise, the ecatfile-system
% can not keep the headers and directoryblocks updated for all fids.
%
% check that seek succeeds.
% seek during write should write zeros in front of the matrix, if the file is too short.
%
% The system updates the index blocks after the data have been written, so that
% the file always will contain valid data. The only exception is when 3D files are written.
% Then the index block is updated after the first 3D-segment has been written. The file
% will thus be too short, until the last 3D-segment has been written.

% ------------------------------------------------------

% pers.handlelist
% pers.filelist
```

```
% pers.filelist{}.filename
% pers.filelist{}.mainheader
% pers.filelist{}.dirblocks.data( 1:n )
% pers.filelist{}.dirblocks.filepos( 1:n )
% pers.filelist{}.pos
% pers.filelist{}.pos.dirblock()
% pers.filelist{}.pos.fpgdb()
% pers.filelist{}.pos.blockstart()
% pers.filelist{}.pos.blockend()
% pers.filelist{}.pos.status()
% pers.filelist{}.pos.header{} % may be empty

  global pers
%  persistent pers

if isempty( pers )
  pers = [];
  pers.handlelist = [];
  pers.filelist = [];
end

switch action

case 'blankhd'
% [ hd, hds, message ]  = ecatfile( 'blankhd', file_system );
% [ mh, hds, message ]  = ecatfile( 'blankhd', 'ecat7' );
% [ mh, hds, message ]  = ecatfile( 'blankhd', 'ecat6.4' );
  file_system = p1;
  if nargin == 2
    [ hds , message ] = getmhs( file_system );
    if isempty( message )
      [ hd , message ] = blankhd( hds );
    end
    res1 = hd;
    res2 = hds;
    res3 = message;
    return
  end
  file_type = p2;
  [ hds , message ] = getshs( file_system, file_type );
  if isempty( message )
    [ hd , message ] = blankhd( hds );
  end
  res1 = hd;
  res2 = hds;
  res3 = message;

case 'matlist'
  % [ matlist, matstatus, message ] = ecatfile( 'matlist', fid );
  res1 = [];
  res2 = [];
  fid = p1;
```

```
  res3 = 'Illegal handle';
  if fid > length( pers.handlelist ) | fid < 1
    res3 = 'Illegal handle';
    return;
  elseif pers.handlelist( fid ) == 0
    res3 = 'Illegal handle';
    return;
  end
  file = pers.filelist{ pers.handlelist( fid ) };

  res1 = file.pos.fpgdb;
  res2 = file.pos.status;
  res3 = '';
  return

case 'matranges'
  % [ matranges, message ] = ecatfile( 'matranges', fid );
  res1 = [];
  res2 = '';
  fid = p1;

  if fid > length( pers.handlelist ) | fid < 1
    res2 = 'Illegal handle';
    return;
  elseif pers.handlelist( fid ) == 0
    res2 = 'Illegal handle';
    return;
  end
  file = pers.filelist{ pers.handlelist( fid ) };

  [ ranges, message ] = getranges( file );
  res1 = ranges;
  res2 = message;
  return

case 'close'
  % message             = ecatfile( 'close', fid );
  % message             = ecatfile( 'close', 'all' );

  res1 = '';
  fid = p1;
  if strcmp( fid,'all' )
    pers = [];
    pers.handlelist = [];
    pers.filelist = [];
    res1 = '';
    return
  end

  if fid > length( pers.handlelist ) | fid < 1
    res1 = 'Illegal handle';
    return;
  elseif pers.handlelist( fid ) == 0
```

```
    res1 = 'Illegal handle';
    return;
  end

  fileno = pers.handlelist( fid );
  pers.handlelist( fid ) = 0;

  if any( pers.handlelist == fileno ) | fid < 1
    return
  end

  pers.filelist{ fileno } = [];
  return

case 'write'
  % message = ecatfile( 'write', fid, vol, hd, selmatrix );
  % message = ecatfile( 'write', fid, vol, hd, selmatrix, segment ); % 3D sinogram
  fid = p1;
  vol = p2;
  hd = p3;
  selmatrix = p4;
  % 3D sinogram segment:
  if nargin < 6
    selsegment = 1;
  else
    selsegment = p5;
  end

  res1 = '';

  postponeindexupdate = 0;

  xyz_dimension = size( vol );
  while length( xyz_dimension ) < 3
    xyz_dimension = [ xyz_dimension, 1 ];
  end
  if prod( xyz_dimension ) ~= prod( xyz_dimension(1:3 ) )
    error( 'dimension of vol is higher than 3' )
    return
  end

  if fid > length( pers.handlelist )
    res1 = 'Illegal handle';
    return;
  elseif pers.handlelist( fid ) == 0
    res1 = 'Illegal handle';
    return;
  end

  file = pers.filelist{ pers.handlelist( fid ) };

  [ datadescr, message ] = getdatadescr( file.mainheader.file_system, ...
    file.mainheader.file_type, hd.sh.data_type );
```

```
if isempty( message )
  [ hd, offsetfromheader, writefill, numdatablocks, message ] = ...
    set_xyz_dimension( hd, datadescr, xyz_dimension, selsegment  );
end

if isempty( message )
  [ hds, message ] = getshs( file.mainheader.file_system, file.mainheader.file_type );
end

if ~isempty( message )
  return
end

% total number of blocks == numdatablocks + hds.header_size / 512

fidphys = fopen( file.filename, 'r+', file.mhmachineformat );

if isempty( file.pos.fpgdb )
  matpos = 0;
else
  matpos = firstno( ...
    file.pos.fpgdb( 1, : ) == selmatrix( 1 ) & ...
    file.pos.fpgdb( 2, : ) == selmatrix( 2 ) & ...
    file.pos.fpgdb( 3, : ) == selmatrix( 3 ) & ...
    file.pos.fpgdb( 4, : ) == selmatrix( 4 ) & ...
    file.pos.fpgdb( 5, : ) == selmatrix( 5 ) );
end

% calculate number of blocks
%  data_type.ftype = 'int16';
%  data_type.size = 2;
%  data_type.fopentype = 'ieee-be';

if matpos == 0
  % entry does not exist, create entry
  % allocate space in the index block,
  % do not write the updated index block yet
  % function matnum = fpgdb2matnum( fpgdb )

  % find an index block that is not full
  indexblockno = firstno( file.dirblocks.data( 1, : ) ~= 0 );

  if indexblockno == 0
    % all index blocks are full, create a new index block
    newindexblock = max( max( file.dirblocks.data( 1, : ) ), ...
      max( file.pos.blockend ) ) + 1;
    lastindexblock = file.dirblocks.filepos( end );
    file.dirblocks.data( 2, end ) = newindexblock;
    file.dirblocks.data( 3, 1 ) = newindexblock;
    file.dirblocks.filepos = [ file.dirblocks.filepos, newindexblock ];
    file.dirblocks.data( :, end + 1 ) = ...
      [ 31, 2, lastindexblock, 0, zeros( 1, 124 ) ];
```

```
  % the following writing sequence ensures, that next will be consistent in
  % the links, even in case of interruption during writing
  % write new index block;
  status = fseek( fidphys, ( newindexblock - 1 ) * 512, -1 );
  if status ~= 0
    fclose( fidphys );
    res1 = 'Error writing the index block';
    return
  end
  count = fwrite( fidphys, file.dirblocks.data( :, end ), 'uint32' );
  if count ~= 128
    fclose( fidphys );
    res1 = 'Error writing the index block';
    return
  end
  % write first index block
  status = fseek( fidphys, ( 2 - 1 ) * 512, -1 );
  if status ~= 0
    fclose( fidphys );
    res1 = 'Error writing the index block';
    return
  end
  count = fwrite( fidphys, file.dirblocks.data( :, 1 ), 'uint32' );
  if count ~= 128
    res1 = 'Error writing the index block';
    return
  end
  % write last index block
  if length( file.dirblocks.filepos ) > 2
    status = fseek( fidphys, ( file.dirblocks.filepos( end - 1 ) - 1 ) * 512, -1 );
    if status ~= 0
      fclose( fidphys );
      res1 = 'Error writing the index block';
      return
    end
    count = fwrite( fidphys, file.dirblocks.data( :, end - 1 ), ...
      'uint32' );
    if count ~= 128
      fclose( fidphys );
      res1 = 'Error writing the index block';
      return
    end
  end
  pers.filelist{ pers.handlelist( fid ) } = file;

  indexblockno = length( file.dirblocks.filepos );
end % create a new index block

% add item into indexblock
% 0: matnum
% 1: first data block (subheader)
% 2: last data block
```

```
  % 3: status: 1:exists; 0: allocated, no data yet written, -1: matrix deleted
  startblock = max( max( file.dirblocks.data( 2, : ) ), ...
    max( [ file.pos.blockend, 0 ] ) ) + 1;
  p = file.dirblocks.data( 4, indexblockno ) * 4 + 4;
  file.dirblocks.data( p+1:p+4, indexblockno ) = [ ...
      fpgdb2matnum( selmatrix ), ...
      startblock, ...
      startblock + numdatablocks + hds.header_size / 512 - 1, ...
      1 ];
  file.dirblocks.data( 1, indexblockno ) = file.dirblocks.data( 1, indexblockno ) - 1;
  file.dirblocks.data( 4, indexblockno ) = file.dirblocks.data( 4, indexblockno ) + 1;

  matpos = length( file.pos.dirblock ) + 1;
  file.pos.dirblock( matpos ) = indexblockno;
  file.pos.fpgdb( :, matpos ) = selmatrix( : );
  file.pos.blockstart( matpos ) = startblock;
  file.pos.blockend( matpos ) = ...
    startblock + numdatablocks + hds.header_size / 512 - 1;
  file.pos.status( matpos ) = 1;
  file.pos.header{ matpos } = hd.sh;
  file.pos.relpos( matpos ) = p / 4;

  postponeindexupdate = 1;
  % postpone the updating of file and writing the above changes to the hard disk
  % until the matrix has been succesfully written
else
  % entry exists, check size of existing entry
  if file.pos.blockend( matpos ) - file.pos.blockstart( matpos ) + 1 ~= ...
      numdatablocks + hds.header_size / 512
    error( 'Trying to change the size of a matrix' )
    return
  end
  file.pos.header{ matpos } = hd.sh;
  if file.pos.status ~= 1
    % change status to 1
    p = file.pos.relpos( matpos );
    file.pos.status( matpos ) = 1;
    file.dirblocks.data( p * 4 + 4, indexblockno ) = 1;
    postponeindexupdate = 1;
  end
end

% pers.filelist{}.pos.dirblock()
% pers.filelist{}.pos.fpgdb()
% pers.filelist{}.pos.blockstart()
% pers.filelist{}.pos.blockend()
% pers.filelist{}.pos.status()
% pers.filelist{}.pos.header{} % may be empty

status = fseek( fidphys, ( file.pos.blockstart( matpos ) - 1 ) * 512, -1 );
if status ~= 0
  fclose( fidphys );
  res1 = 'Error writing matrix';
```

```
  return
end

message = writehd( fidphys, hd.sh, hds );
if ~isempty( message )
  fclose( fidphys );
  res1 = message;
  return
end

% 'a', ftell( fidphys )

% skip to the right 3D segment
if offsetfromheader ~= 0
  fseek( fidphys, offsetfromheader, 0 );
end

for pl = 1:xyz_dimension( 3 )
  count = fwrite( fidphys, double( vol( :, :, pl ) ), datadescr.ftype );
  if count ~= prod( xyz_dimension( 1:2 ) )
    fclose( fidphys );
    res1 = 'Error writing matrix';
    return
  end
end

% fill last block with zeros
count = fwrite( fidphys, zeros( 1, writefill / datadescr.size ), datadescr.ftype );
if count ~= writefill / datadescr.size
  fclose( fidphys );
  res1 = 'Error writing matrix';
  return
end

if hds.header_size + ...
    ( file.pos.blockstart( matpos ) - 1 + numdatablocks ) * 512 ...
    ~= ftell( fidphys )
  % warning( 'The length of the written matrix is wrong.' );
end

% write the updated indexblock
if postponeindexupdate
  status = fseek( fidphys, ( file.dirblocks.filepos( indexblockno ) - 1 ) * 512, -1 );
  if status ~= 0
    fclose( fidphys );
    res1 = 'Error writing the index block';
    return
  end
  count = fwrite( fidphys, file.dirblocks.data( :, indexblockno ), 'uint32' );
  if count ~= 128
    fclose( fidphys );
    res1 = 'Error writing the index block';
    return
```

```
    end
  end

  pers.filelist{ pers.handlelist( fid ) } = file;

  fclose( fidphys );
  return
case { 'read', 'hd', 'read_offset' }
% [ vol, hd, message ]  = ecatfile( 'read', fid, selmatrix );
% [ vol, hd, message ]  = ecatfile( 'read', fid, selmatrix, segment ); % 3D sinogram
% [ descr, hd, message ] = ecatfile( 'read_offset', fid, selmatrix );
% [ descr, hd, message ] = ecatfile( 'read_offset', fid, selmatrix, segment ); % 3D sinogram
% [ hd, message ]       = ecatfile( 'hd', fid, selmatrix );
  fid = p1;
  selmatrix = p2;
  % 3D sinogram segment:
  if nargin < 4
    selsegment = 1;
  else
    selsegment = p3;
  end
  vol = [ ];
  hd = [ ];
  descr = [];
  message = '';
  if fid > length( pers.handlelist ) | fid <= 0
    message = 'Illegal handle';
  elseif pers.handlelist( fid ) == 0
    message = 'Illegal handle';
  else
    file = pers.filelist{ pers.handlelist( fid ) };
    if isempty( file.pos.fpgdb )
      message = 'Matrix does not exist';
    else
      matpos = firstno( ...
        file.pos.fpgdb( 1, : ) == selmatrix( 1 ) & ...
        file.pos.fpgdb( 2, : ) == selmatrix( 2 ) & ...
        file.pos.fpgdb( 3, : ) == selmatrix( 3 ) & ...
        file.pos.fpgdb( 4, : ) == selmatrix( 4 ) & ...
        file.pos.fpgdb( 5, : ) == selmatrix( 5 ) );
      if matpos == 0
        message = 'Matrix does not exist';
      end
    end
  end

  % pers.filelist{}.pos.dirblock()
  % pers.filelist{}.pos.fpgdb()
  % pers.filelist{}.pos.blockstart()
  % pers.filelist{}.pos.blockend()
  % pers.filelist{}.pos.status()
  % pers.filelist{}.pos.header{} % may be empty
```

```
  if isempty( message )
    fidphys = fopen( file.filename, 'r', file.mhmachineformat );
    fseek( fidphys, ( file.pos.blockstart( matpos ) - 1 ) * 512, -1 );
    [ hds, message ] = getshs( file.mainheader.file_system, file.mainheader.file_type );
  else
    fidphys = -1;
  end

  if isempty( message )
    [ sh, message ] = readhd( fidphys, hds );
    hd.sh =sh;
    hd.mh = file.mainheader;
  end

  if strcmp( action, 'hd' )
    res1 = hd;
    res2 = message;
    if fidphys > 0
      fclose( fidphys );
    end
    return
  end

  % 'a', ftell( fidphys )

  if isempty( message )
    [ descr, message ] = getdatadescr( file.mainheader.file_system, file.mainheader.file_type, sh.data_type
);
  end

  if isempty( message )
    [ xyz_dimension, offsetfromheader ] = get_xyz_dimension( hd, descr, selsegment );
    descr.xyz_dimension = xyz_dimension;
    descr.headeroffset = ( file.pos.blockstart( matpos ) - 1 ) * 512;
    descr.header_size = hds.header_size;
    descr.dataoffset = descr.headeroffset + hds.header_size + offsetfromheader;
  end

  if strcmp( action, 'read' ) & isempty( message )

    % skip to the right 3D segment
    if offsetfromheader ~= 0
      fseek( fidphys, offsetfromheader, 0 );
    end

    % preallocate vol:
    switch descr.ftype
    case 'float32'
      vol1 = zeros( xyz_dimension( 1 ), xyz_dimension( 2 ) );
    case { 'int8', 'int16' }
      vol1 = int16( zeros( xyz_dimension( 1 ), xyz_dimension( 2 ) ) );
    end
    vol = vol1( :, :, ones( xyz_dimension( 3 ), 1 ) );
```

```
    for pl = 1:xyz_dimension( 3 )
      [ vol1, count ] = fread( fidphys, prod( xyz_dimension( 1:2 ) ), ...
        descr.ftype );
      if count ~= prod( xyz_dimension( 1:2 ) )
        message = 'file too short';
        break
      end
      switch descr.ftype
      case 'float32'
        vol( :, :, pl ) = ...
          reshape( vol1, xyz_dimension( 1:2 ) );
      case { 'int8', 'int16' }
        vol( :, :, pl ) = ...
          reshape( int16( vol1 ), xyz_dimension( 1:2 ) );
      end
    end
  end

  if strcmp( action, 'read' )
    if ~isempty( message )
      vol = [];
    end
    res1 = vol;
  else
    res1 = descr;
  end

  res2 = hd;
  res3 = message;
  if fidphys > 0
    fclose( fidphys );
  end

  return

case 'writemh'
  % message = ecatfile( 'writemh', fidphys, mh );
  fid = p1;
  mh = p2;
  res1 = '';
  if fid > length( pers.handlelist ) | fid <= 0
    res1 = 'Illegal handle';
  elseif pers.handlelist( fid ) == 0
    res1 = 'Illegal handle';
  end
  file = pers.filelist{ pers.handlelist( fid ) };
  fidphys = fopen( file.filename, ...
    'r+', file.mhmachineformat );
  if fidphys <= 0
    res1 = [ 'Can''t open file: ', file.filename ];
    return
  end
```

```
  [ hds, message ] = getmhs( mh.file_system );
  if isempty( message )
    message = writehd( fidphys, mh, hds )
  end
  fclose( fidphys );
  if ~isempty( message )
    res1 = message;
    return
  end
  pers.filelist{ pers.handlelist( fid ) }.mainheader = mh;
  return

case 'mh'
  % [ mh, message ] = ecatfile( 'mh', fid );
  res1 = [];
  res2 = '';
  fid = p1;
  if fid <= 0 | fid > length( pers.handlelist ) | fid <= 0
    res2 = 'Illegal handle';
    return;
  elseif pers.handlelist( fid ) == 0
    res2 = 'Illegal handle';
    return;
  end
  res1 = pers.filelist{ pers.handlelist( fid ) }.mainheader;
  return

case { 'open', 'create' }
% [ fid, message ]      = ecatfile( 'open', filename, file_system );
% [ fid, message ]      = ecatfile( 'open', filename, 'ecat7' );
% [ fid, message ]      = ecatfile( 'open', filename, 'ecat6.4' );
% [ fid, message ]      = ecatfile( 'open', filename, '' );
% [ fid, message ]      = ecatfile( 'create', filename, mh );
  filename = p1;
  if strcmp( filename, '' )
    fid = -1;
    message = 'Illegal filename';
    res1 = fid;
    res2 = message;
    return
  end

  if nargin < 3
    p2 = '';
  end

  for i = 1:length( pers.filelist )
    if ~isempty( pers.filelist{ i } )
      if strcmp( pers.filelist{ i }.filename, filename )
        if strcmp( action, 'create' )
          % file already open. Truncate the file.
          [ file, message ] = openecat( filename, 'create', p2 );
          if ~isempty( message )
```

```
            fid = -1;
            res1 = fid;
            res2 = message;
            return
          end
          pers.filelist{ i } = file;
        end
        % file already open, return a duplicate fid
        fid = firstno( [ pers.handlelist, 0 ] == 0 );
        pers.handlelist( fid ) = i;
        message = '';
        res1 = fid;
        res2 = message;
        return
      end
    end
  end

  [ file, message ] = openecat( filename, action, p2 );
  if ~isempty( message )
    fid = -1;
    res1 = fid;
    res2 = message;
    return
  end

  pos = length( pers.filelist ) + 1;
  for i = 1:length( pers.filelist )
    if isempty( pers.filelist{ i } )
      pos = i;
      break
    end
  end

  pers.filelist{ pos } = file;
  fid = firstno( [ pers.handlelist, 0 ] == 0 );
  pers.handlelist( fid ) = pos;
  message = '';
  res1 = fid;
  res2 = message;
  return

otherwise
  error( 'Illegal action' )

end % switch action

% ------------------------------------------------------------
function matnum = fpgdb2matnum( fpgdb )
frame = fpgdb( 1 );
plane = fpgdb( 2 );
gate = fpgdb( 3 );
data = fpgdb( 4 );
```

```
bed = fpgdb( 5 );
matnum = bitand( frame, 511 ) + ...
  bitshift( bitand( plane, 3 * 256 ), 9 - 8 ) + ...
  bitshift( bitand( data, 4 ), 11 - 2 ) + ...
  bitshift( bitand( bed, 15 ), 12 ) + ...
  bitshift( bitand( plane, 255 ), 16 ) + ...
  bitshift( bitand( gate, 63 ), 24 ) + ...
  bitshift( bitand( data, 3 ), 30 );

% ------------------------------------------------------------

function [ file, message ] = openecat( filename, create, file_system );
% function [ file, message ] = openecat( filename, 'open', file_system );
% function [ file, message ] = openecat( filename, 'create', mh );
if nargin < 2
  create = 'open';
end

message = '';
file = [];

% read mh, matdir, fpgdb

if strcmp( create, 'create' )
  % truncate or create
  mh = file_system;
  file_system = mh.file_system;
  switch mh.file_system
  case 'ecat7'
    fidphys = fopen( filename, 'w', 'ieee-be'  );
  case 'ecat6.4'
    fidphys = fopen( filename, 'w', 'vaxg'  );
  otherwise
    error( 'Wrong header' )
  end
  if fidphys == -1
    message = [ 'Can''t open file: ', filename ];
    return
  end
  [ hds, message ] = getmhs( mh.file_system );
  if isempty( message )
    message = writehd( fidphys, mh, hds );
  end
  if ~isempty( message )
    fclose( fidphys );
    return
  end

  fwrite( fidphys, 31, 'uint32' ); % 31 unused entries
  fwrite( fidphys,  2, 'uint32' ); % 1: next index block
  fwrite( fidphys,  2, 'uint32' ); % 2: previous index block ? 0
  fwrite( fidphys,  0, 'uint32' ); % 3: number of used entries in the block?
  fwrite( fidphys, zeros( 4 * 31, 1 ), 'uint32' );
```

```
  if ftell( fidphys ) ~= 1024
    message = 'File write error';
  end
  fclose( fidphys );
end % create or truncate

[ file, message ] = readmh( filename, file_system );

if ~isempty( message )
  file = [];
  return
end

if strcmp( create, 'create' )
  fidphys = fopen( filename, 'r+', file.mhmachineformat ); % 'w+' ?
else
  fidphys = fopen( filename, 'r', file.mhmachineformat ); % 'w+' ?
end

if fidphys == -1
  file = [];
  message = [ 'Can"t open file: ', filename ];
  return
end

% read indexblocks

% first entry of an index block:
% 0: number of unused entries?
% 1: next index block
% 2: previous index block
% 3: number of used entries in the block?

% succeeding entries in an index block:
% 0: matnum
% 1: first data block (subheader)
% 2: last data block
% 3: status: 1:exists; 0: allocated, no data yet written, -1: matrix deleted

fseek( fidphys, 512, -1 );

blockno = 2;
matdir = [ ];
ok = 1;

file.dirblocks.data = [ ];
file.dirblocks.filepos = [ ];

while ok
  fseek( fidphys, 512 * ( blockno - 1), -1 );
  [ a1, count]= fread( fidphys, 128, 'uint32');
  if count < 128
    file = [];
```

```
    message = [ 'Can"t read index block: ', filename ];
    return
  end

  file.dirblocks.data = [ file.dirblocks.data, a1 ];
  file.dirblocks.filepos = [ file.dirblocks.filepos, blockno ];

  if a1(2) == 2
    ok = 0;
  elseif any( a1(2) == file.dirblocks.filepos( 2:end ) ) | a1( 2 ) <= 0
    % error if the directory block numbers are circular or not positive
    message = 'Illegal index block numbers';
    file = [];
    return;
  end

  blockno = a1(2);

end

% matdir(1:8, :)

%  fclose(fidphys);

sz = size( file.dirblocks.data );

direntries = [ ];
file.pos.dirblock = [ ];
file.pos.relpos = [ ];
for i = 1:sz( 2 )
  jj = file.dirblocks.data( 4, i );
  if jj < 0 | jj > 31
    message = 'Wrong index block';
    file = [];
    return
  end
  direntries = [ direntries, reshape( file.dirblocks.data( 5:4 + jj * 4, i ), 4, jj ) ];
  file.pos.dirblock = [ file.pos.dirblock, i + zeros( 1, jj ) ];
  file.pos.relpos = [ file.pos.relpos, 1:jj ];
end

fpgdb = file.dirblocks.data( 5:128, : );

fpgdb = [ zeros(7, sz(2)*31); reshape( fpgdb, 4, sz(2)*31 ) ];

m = direntries( 1, : );
frame   = rem( m, 512 ); m = floor( m / 512 );
hiplane = rem( m, 4 ); m = floor( m / 4 );
hidata  = rem( m, 2 ); m = floor( m / 2 );
bed     = rem( m, 16 ); m = floor( m / 16 );
plane   = rem( m, 256 ) + hiplane * 256; m = floor( m / 256 );
gate    = rem( m, 64 ); m = floor( m / 64 );
data    = rem( m, 4 ) + hidata * 4; m = floor( m / 4 );
```

```
file.pos.fpgdb = [ frame; plane; gate; data; bed ];
file.pos.blockstart = direntries( 2, : );
file.pos.blockend = direntries( 3, : );
file.pos.status = direntries( 4, : );

fclose( fidphys );

% ------------------------------------------------------
function [ ranges, message ] = getranges( file )

ranges = [ ];
message = '';

% check homogeneity
if length( file.pos.fpgdb( 1, : ) ) == 1
  m1 = file.pos.fpgdb;
  m2 = m1;
else
  m1 = min( file.pos.fpgdb' )';
  m2 = max( file.pos.fpgdb' )';
end
tot = m2 - m1 + 1;
f = 1;
for i = 1:5
  f( i, 1 ) = f( end ) * tot( i );
end

if f(5) ~= length( file.pos.status )
  message = 'Inhomogeneous fpgdb\n';
  return
end
testexists = zeros( f(5), 1 );
p = 1 + [ 1, f(1:4 )'] * ...
  ( file.pos.fpgdb - m1( :, ones( 1, size( file.pos.fpgdb, 2 ) ) ) );

testexists( p ) = 1;

if sum(testexists) ~= length( file.pos.status )
  message = 'Inhomogeneous fpgdb\n';
  return
end

ranges = [ m1, m2 ]';
if any( file.pos.status ~= 1 )
  message = 'Matrices with wrong status\n';
end

return

% ------------------------------------------------------

function [ hd, message ] = blankhd( hds )
```

```
message = '';
hd = [];
n = 0;
pos = [];

for i =1:length( hds.s )
  pos( end + 1 ) = n;
  switch hds.s(i).type
  case 'int16'
    n = n + 2 * hds.s(i).noelements;
    hd = setfield( hd, hds.s(i).fieldname, zeros( 1, hds.s(i).noelements ) );
  case { 'float32', 'mat1st', 'mat2nd', 'int32' }
    n = n + 4 * hds.s(i).noelements;
    hd = setfield( hd, hds.s(i).fieldname, zeros( 1, hds.s(i).noelements ) );
  case 'char'
    n = n + 1 * hds.s(i).noelements;
    hd = setfield( hd, hds.s(i).fieldname, '' );
  end % switch hds.s(i).type
end
if n ~= hds.header_size
  error( 'Blank header, wrong internal description' )
end

% make sure that ecat 7 mh contains a non-zero magic number
if strcmp( hds.file_system, 'ecat7' )
  hd.magic_number = 'MATRIX7';
end

% ------------------------------------------------

function message = writehd( fidphys, hd, hds )
message = '';
initialpos = ftell( fidphys );

% make sure that ecat 7 mh contains a non-zero magic number
if strcmp( hd.file_system, 'ecat7' )
  if isempty( hd.magic_number )
    hd.magic_number = 'MATRIX7';
  elseif all( hd.magic_number == 0 )
    hd.magic_number = 'MATRIX7';
  end
end

ok = 1;
for i =1:length( hds.s )
  val = getfield( hd, hds.s(i).fieldname );
  if strcmp( hds.s(i).type, 'char' )
    if length( val(:) ) < hds.s(i).noelements
      val = [ val(:); char( ...
          zeros( hds.s(i).noelements - length( val(:) ), 1 ) ) ];
    end
  end
  if strcmp( hds.s(i).type, 'mat1st' )
```

```
    if length( val(:) ) ~= 12
      error( 'illegal header length\n' )
    end
  elseif strcmp( hds.s(i).type, 'mat2nd' )
    % the size has already been tested
  elseif length( val(:) ) ~= hds.s(i).noelements
    error( 'illegal header length\n' )
  end

  switch hds.s(i).type
  case 'int16'
    ok = ok & length( val(:) ) == fwrite( fidphys, val, 'int16' );
  case 'int32'
    ok = ok & length( val(:) ) == fwrite( fidphys, val, 'int32' );
  case 'float32'
    ok = ok & length( val(:) ) == fwrite( fidphys, val, 'float32' );
  case 'mat1st'
    val = val( 1:3, 1:3 )';
    ok = ok & length( val(:) ) == fwrite( fidphys, val(:), 'float32' );
  case 'mat2nd'
    val = val(:,4);
    ok = ok & length( val(:) ) == fwrite( fidphys, val, 'float32' );
  case 'char'
    ok = ok & length( val(:) ) == fwrite( fidphys, val, 'uchar' );
  end % switch hds.s(i).type
end
if ~ok
  message = 'Can"t write header';
  return
end

if ftell( fidphys ) - initialpos ~= hds.header_size
  error( 'Writing header, wrong internal description' )
end

% ------------------------------------------------------
function [ file, message ] = readmh( filename, file_system )
% this function determines the header type and the machineformat
% that is used for reading the main header and the index blocks.
% if the file is too short, it returns file = [], otherwise, it returns
%   file.mainheader = mh;
%   file.mhmachineformat = mhmachineformat;
%   file.filename = filename;
% the routine uses ad hoc testing to determine the format. This testing may
% need to be changed if the routine can not determine the format correctly

message = '';
file = [];
mhmachineformat = 'ieee-be';
mh = [];

fidphys = fopen( filename, 'r', mhmachineformat );
if fidphys == -1
```

```
  message = [ 'Can"t open file: ', filename ];
  return
end

magic_number = fread( fidphys, 14, 'char' );
% check length of main header
[ mh, count ]= fread( fidphys, 256-7, 'uint16');
if count < 256-7
  message = 'Header read error';
  return
end

if ( all( magic_number == 0 ) & strcmp( file_system, '' ) ) | ...
    strcmp( file_system, 'ecat6.4' )
  fclose( fidphys );
  % ecat 6.4
  mhmachineformat = 'vaxg';
  fidphys = fopen( filename, 'r', mhmachineformat );
  if fidphys == -1
    return
  end
  hds = getmhs( 'ecat6.4' );
elseif strcmp( file_system, '' ) | strcmp( file_system, 'ecat7' )
  % ecat7
  fseek( fidphys, 0, -1 );
  hds = getmhs( 'ecat7' );
else
  message = 'Illegal file system';
  fclose( fidphys );
  return
end

[ mh, message ] = readhd( fidphys, hds );
if ~isempty( mh )
  file.mainheader = mh;
  file.mhmachineformat = mhmachineformat;
  file.filename = filename;
end

fclose( fidphys );

% ------------------------------------------------------

function [ hd, message ] = readhd( fidphys, hds )
message = '';
ok = 1;
hd = [];
initialpos = ftell( fidphys );

hd.file_system = hds.file_system;

for i =1:length( hds.s )
  switch hds.s(i).type
```

```
  case 'int16'
    [ val, count ] = fread( fidphys, hds.s(i).noelements, 'int16' );
    hd = setfield( hd, hds.s(i).fieldname, val' );
  case 'int32'
    [ val, count ] = fread( fidphys, hds.s(i).noelements, 'int32' );
    hd = setfield( hd, hds.s(i).fieldname, val' );
  case { 'float32', 'mat1st' }
    [ val, count ] = fread( fidphys, hds.s(i).noelements, 'float32' );
    hd = setfield( hd, hds.s(i).fieldname, val' );
  case 'mat2nd'
    [ val, count ] = fread( fidphys, hds.s(i).noelements, 'float32' );
    mt = getfield( hd, hds.s(i).fieldname );
    mt = [ reshape( mt, 3, 3 )', val ];
    hd = setfield( hd, hds.s(i).fieldname, mt );
  case 'char'
    [ val, count ] = fread( fidphys, hds.s(i).noelements, 'uchar' );
    val = char( val( val ~= 0 ) );
    hd = setfield( hd, hds.s(i).fieldname, val' );
  otherwise
    count = hds.s(i).noelements;
  end % switch hds.s(i).type
  ok = ok & count == hds.s(i).noelements;

end
if ~ok
  message = 'Header read error';
end
if ftell( fidphys ) - initialpos ~= hds.header_size
  error( 'Reading header, wrong internal description' )
end

function [ hds, message ] = getmhs( file_system )
message = '';

switch file_system

case 'ecat7'
  a = { 'magic_number',              14, 'char'
    'original_file_name',          32, 'char'
    'sw_version',                  1, 'int16'
    'system_type',                 1, 'int16'
    'file_type',                 1, 'int16'
    'serial_number',               10, 'char'
    'scan_start_time',             1, 'int32'
    'isotope_name',                8, 'char'
    'isotope_halflife',            1, 'float32'
    'radiopharmaceutical',           32, 'char'
    'gantry_tilt',               1, 'float32'
    'gantry_rotation',             1, 'float32'
    'bed_elevation',               1, 'float32'
    'intrinsic_tilt',            1, 'float32'
    'wobble_speed',                1, 'int16'
    'transm_source_type',            1, 'int16'
```

```
    'distance_scanned',              1, 'float32'
    'transaxial_fov',                1, 'float32'
    'angular_compression',           1, 'int16'
    'coin_samp_mode',                1, 'int16'
    'axial_samp_mode',               1, 'int16'
    'ecat_calibration_factor',       1, 'float32'
    'calibration_units',             1, 'int16'
    'calibration_units_label',       1, 'int16'
    'compression_code',              1, 'int16'
    'study_type',                   12, 'char'
    'patient_id',                   16, 'char'
    'patient_name',                 32, 'char'
    'patient_sex',                   1, 'char'
    'patient_dexterity',             1, 'char'
    'patient_age',                   1, 'float32'
    'patient_height',                1, 'float32'
    'patient_weight',                1, 'float32'
    'patient_birth_date',            1, 'int32'
    'physician_name',               32, 'char'
    'operator_name',                32, 'char'
    'study_description',            32, 'char'
    'acquisition_type',              1, 'int16'
    'patient_orientation',           1, 'int16'
    'facility_name',                20, 'char'
    'num_planes',                    1, 'int16'
    'num_frames',                    1, 'int16'
    'num_gates',                     1, 'int16'
    'num_bed_pos',                   1, 'int16'
    'bed_position',                 16, 'float32'
    'plane_separation',              1, 'float32'
    'lwr_sctr_thres',                1, 'int16'
    'lwr_true_thres',                1, 'int16'
    'upr_true_thres',                1, 'int16'
    'user_process_code',            10, 'char'
    'acquisition_mode',              1, 'int16'
    'bin_size',                      1, 'float32'
    'branching_fraction',            1, 'float32'
    'dose_start_time',               1, 'int32'
    'dosage',                        1, 'float32'
    'well_counter_corr_factor',      1, 'float32'
    'data_units',                   32, 'char'
    'septa_state',                   1, 'int16'
    'fill',                          6, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;
  hds.file_system = 'ecat7';

case 'ecat6.4'
  a = { 'fill1',                    14, 'int16' % changed type
    'original_file_name',           20, 'char' % changed length
    'sw_version',                    1, 'int16'
```

```
'data_type',                1, 'int16' % ecat6.4
'system_type',              1, 'int16'
'file_type',                1, 'int16'
'node_id',                 10, 'char' % serial_number
... %'scan_start_time', 1, 'int32'
'scan_start_day',           1, 'int16' % ecat6.4
'scan_start_month',         1, 'int16' % ecat6.4
'scan_start_year',          1, 'int16' % ecat6.4
'scan_start_hour',          1, 'int16' % ecat6.4
'scan_start_minute',        1, 'int16' % ecat6.4
'scan_start_second',        1, 'int16' % ecat6.4
'isotope_code',             8, 'char' % isotope_name
'isotope_halflife',         1, 'float32'
'radiopharmaceutical',     32, 'char'
'gantry_tilt',              1, 'float32'
'gantry_rotation',          1, 'float32'
'bed_elevation',            1, 'float32'
... %'intrinsic_tilt', 1, 'float32'
'rot_source_speed',         1, 'int16' % ecat6.4
'wobble_speed',             1, 'int16'
'transm_source_type',       1, 'int16'
... %'distance_scanned', 1, 'float32'
'axial_fov',                1, 'float32'
'transaxial_fov',           1, 'float32'
'transaxial_samp_mode',     1, 'int16' % ecat6.4
... %'angular_compression', 1, 'int16'
'coin_samp_mode',           1, 'int16'
'axial_samp_mode',          1, 'int16' % subtracted 1 in ecat7
'calibration_factor',       1, 'float32' % ecat_calibration_factor
'calibration_units',        1, 'int16'
... %'calibration_units_label', 1, 'int16'
'compression_code',         1, 'int16'
'study_name',              12, 'char' % study_type
'patient_id',              16, 'char'
'patient_name',            32, 'char' % changed length
'patient_sex',              1, 'char'
'patient_age',             10, 'char' % changed type
'patient_height',          10, 'char' % changed type
'patient_weight',          10, 'char' % changed type
'patient_dexterity',        1, 'char' % changed position
... %'patient_birth_date', 1, 'int32'
'physician_name',          32, 'char'
'operator_name',           32, 'char'
'study_description',       32, 'char'
'acquisition_type',         1, 'int16'
'bed_type',                 1, 'int16' % ecat6.4
'septa_type',               1, 'int16'
... %'patient_orientation', 1, 'int16'
'facility_name',           20, 'char'
'num_planes',               1, 'int16'
'num_frames',               1, 'int16'
'num_gates',                1, 'int16'
'num_bed_pos',              1, 'int16'
```

```
    'bed_position',              16, 'float32'
    'plane_separation',           1, 'float32'
    'lwr_sctr_thres',             1, 'int16'
    'lwr_true_thres',             1, 'int16'
    'upr_true_thres',             1, 'int16'
    'collimator',                 1, 'float32' % ecat6.4
    'user_process_code',         10, 'char'
    ... %'acquisition_mode', 1, 'int16'
    'fill2',                     20, 'int16' };
    %'bin_size', 1, 'float32'
    %'branching_fraction', 1, 'float32'
    %'dose_start_time', 1, 'int32'
    %'dosage', 1, 'float32'
    %'well_counter_corr_factor', 1, 'float32'
    %'data_units', 32, 'char'
    %'septa_state', 1, 'int16'
    %'fill', 6, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;
  hds.file_system = 'ecat6.4';
otherwise
  message = 'Illegal file system';
  hds = [];

end % switch file_system

function [ hds, message ] = getshs( file_system, file_type );
message = '';
hds = [];

switch file_system
case 'ecat7'

  switch file_type

    % implemented:
    % 01=Sinogram, 03=Attenuation Correction, 04=Normalization,
    % 05=Polar Map, 07=Volume 16, 11=3D Sinogram 16, 13=3D Normalization

    % not implemented:
    % 00=unknown, 02=Image-16, 06=Volume 8, 08=Projection 8,
    % 09=Projection 16, 10=Image 8, 12=3D Sinogram 8, 14=3D Sinogram Fit

  case 1
    % sinogram
    a = { ...
        'data_type',              1, 'int16'
      'num_dimensions',           1, 'int16'
      'num_r_elements',           1, 'int16'
      'num_angles',               1, 'int16'
      'corrections_applied',      1, 'int16'
```

```
    'num_z_elements',              1, 'int16'
    'ring_difference',             1, 'int16'
    'xyz_resolution',              3, 'float32'
    'w_resolution',                1, 'float32'
    'fill1',                    6, 'int16'
    'gate_duration',               1, 'int32'
    'r_wave_offset',               1, 'int32'
    'num_accepted_beats',             1, 'int32'
    'scale_factor',                1, 'float32'
    'scan_min',                    1, 'int16'
    'scan_max',                    1, 'int16'
    'prompts',                     1, 'int32'
    'delayed',                     1, 'int32'
    'multiples',                   1, 'int32'
    'net_trues',                   1, 'int32'
    'cor_singles',                16, 'float32'
    'uncor_singles',               16, 'float32'
    'tot_avg_cor',                 1, 'float32'
    'tot_avg_uncor',               1, 'float32'
    'total_coin_rate',             1, 'int32'
    'frame_start_time',            1, 'int32'
    'frame_duration',              1, 'int32'
    'deadtime_correction_factor',     1, 'float32'
    'physical_planes',             8, 'int16'
    'fill2',                    83, 'int16'
    'fill3',                    50, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;

case 3
  % attenuation
  a = { ...
      'data_type',                 1, 'int16'
    'num_dimensions',              1, 'int16'
    'attenuation_type',            1, 'int16'
    'num_r_elements',              1, 'int16'
    'num_angles',                  1, 'int16'
    'num_z_elements',              1, 'int16'
    'ring_difference',             1, 'int16'
    'x_resolution',                1, 'float32'
    'y_resolution',                1, 'float32'
    'z_resolution',                1, 'float32'
    'w_resolution',                1, 'float32'
    'scale_factor',                1, 'float32'
    'x_offset',                    1, 'float32'
    'y_offset',                    1, 'float32'
    'x_radius',                    1, 'float32'
    'y_radius',                    1, 'float32'
    'tilt_angle',                  1, 'float32'
    'attenuation_coeff',           1, 'float32'
    'attenuation_min',             1, 'float32'
```

```
    'attenuation_max',              1, 'float32'
    'skull_thickness',              1, 'float32'
    'num_additional_atten_coeff',   1, 'int16'
    'additional_atten_coeff',       8, 'float32'
    'edge_finding_threshold',       1, 'float32'
    'storage_order',                1, 'int16'
    'span',                         1, 'int16'
    'z_elements',                   64, 'int16'
    'fill1',                        86, 'int16'
    'fill2',                        50, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;

case 4
  % 6.5 normalization
  a = { ...
      'data_type',                  1, 'int16'
    'num_dimensions',               1, 'int16'
    'num_r_elements',               1, 'int16'
    'num_angles',                   1, 'int16'
    'num_z_elements',               1, 'int16'
    'ring_difference',              1, 'int16'
    'scale_factor',                 1, 'float32'
    'norm_min',                     1, 'float32'
    'norm_max',                     1, 'float32'
    'fov_source_width',             1, 'float32'
    'norm_quality_factor',          1, 'float32'
    'norm_quality_factor_code',     1, 'int16'
    'storage_order',                1, 'int16'
    'span',                         1, 'int16'
    'z_elements',                   64, 'int16'
    'fill1',                        123, 'int16'
    'fill2',                        50, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;

  % polar map
case 5
  a = { ...
      'data_type',                  1, 'int16'
    'polar_map_type',               1, 'int16'
    'num_rings',                    1, 'int16'
    'sectors_per_ring',             32, 'int16'
    'ring_position',                32, 'float32'
    'ring_angle',                   32, 'int16'
    'start_angle',                  1, 'int16'
    'long_axis_left',               3, 'int16'
    'long_axis_right',              3, 'int16'
    'position_data',                1, 'int16'
```

```
    'image_min',                1, 'int16'
    'image_max',                1, 'int16'
    'scale_factor',             1, 'float32'
    'pixel_size',               1, 'float32'
    'frame_duration',           1, 'int32'
    'frame_start_time',         1, 'int32'
    'processing_code',          1, 'int16'
    'quant_units',              1, 'int16'
    'annotation',              40, 'char'
    'gate_duration',            1, 'int32'
    'r_wave_offset',            1, 'int32'
    'num_accepted_beats',       1, 'int32'
    'polar_map_protocol',      20, 'char'
    'database_name',           30, 'char'
    'fill1',                   27, 'int16'
    'fill2',                   27, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;

case 7
  % image
  a = { ...
      'data_type',              1, 'int16'
    'num_dimensions',           1, 'int16'
    'xyz_dimension',            3, 'int16'
    'xyz_offset',               3, 'float32'
    'recon_zoom',               1, 'float32'
    'scale_factor',             1, 'float32'
    'image_min',                1, 'int16'
    'image_max',                1, 'int16'
    'xyz_pixel_size',           3, 'float32'
    'frame_duration',           1, 'int32'
    'frame_start_time',         1, 'int32'
    'filter_code',              1, 'int16'
    'xyz_resolution',           3, 'float32'
    'num_r_elements',           1, 'float32'
    'num_angles',               1, 'float32'
    'z_rotation_angle',         1, 'float32'
    'decay_corr_fctr',          1, 'float32'
    'processing_code',          1, 'int32'
    'gate_duration',            1, 'int32'
    'r_wave_offset',            1, 'int32'
    'num_accepted_beats',       1, 'int32'
    'filter_cutoff_frequency',  1, 'float32'
    'filter_resolution',        1, 'float32'
    'filter_ramp_slope',        1, 'float32'
    'filter_order',             1, 'int16'
    'filter_scatter_fraction',  1, 'float32'
    'filter_scatter_slope',     1, 'float32'
    'annotation',              40, 'char'
    'mt',                       9, 'mat1st'
```

```
    'rfilter_cutoff',              1, 'float32'
    'rfilter_resolution',          1, 'float32'
    'rfilter_code',                1, 'int16'
    'rfilter_order',               1, 'int16'
    'zfilter_cutoff',              1, 'float32'
    'zfilter_resolution',          1, 'float32'
    'zfilter_code',                1, 'int16'
    'zfilter_order',               1, 'int16'
    'mt',                      3, 'mat2nd'
    'scatter_type',                1, 'int16'
    'recon_type',                  1, 'int16'
    'recon_views',                 1, 'int16'
    'fill',                  136, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;

case { 11, 14 }
  % 3D sinogram
  a = { ...
      'data_type',                 1, 'int16'
    'num_dimensions',              1, 'int16'
    'num_r_elements',              1, 'int16'
    'num_angles',                  1, 'int16'
    'corrections_applied',         1, 'int16'
    'num_z_elements',             64, 'int16'
    'ring_difference',             1, 'int16'
    'storage_order',               1, 'int16'
    'axial_compression',           1, 'int16'
    'x_resolution',                1, 'float32'
    'v_resolution',                1, 'float32'
    'z_resolution',                1, 'float32'
    'w_resolution',                1, 'float32'
    'fill1',                   6, 'int16'
    'gate_duration',               1, 'int32'
    'r_wave_offset',               1, 'int32'
    'num_accepted_beats',          1, 'int32'
    'scale_factor',                1, 'float32'
    'scan_min',                    1, 'int16'
    'scan_max',                    1, 'int16'
    'prompts',                     1, 'int32'
    'delayed',                     1, 'int32'
    'multiples',                   1, 'int32'
    'net_trues',                   1, 'int32'
    'tot_avg_cor',                 1, 'float32'
    'tot_avg_uncor',               1, 'float32'
    'total_coin_rate',             1, 'int32'
    'frame_start_time',            1, 'int32'
    'frame_duration',              1, 'int32'
    'deadtime_correction_factor',      1, 'float32'
    'fill2',                   90, 'int16'
    'fill3',                   50, 'int16'
```

```
      'uncor_singles',              128, 'float32' };

    hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
      'type', a(:,3) );
    hds.header_size = 1024;

  case 13
    % 3D normalization
    a = { ...
        'data_type',                  1, 'int16'
      'num_r_elements',               1, 'int16'
      'num_transaxial_crystals',      1, 'int16'
      'num_crystal_rings',            1, 'int16'
      'crystals_per_ring',            1, 'int16'
      'num_geo_corr_planes',          1, 'int16'
      'uld',                        1, 'int16'
      'lld',                        1, 'int16'
      'scatter_energy',             1, 'int16'
      'norm_quality_factor',        1, 'float32'
      'norm_quality_factor_code',   1, 'int16'
      'ring_dtcor1',               32, 'float32'
      'ring_dtcor2',               32, 'float32'
      'crystal_dtcor',              8, 'float32'
      'span',                       1, 'int16'
      'max_ring_diff',              1, 'int16'
      'fill1',                     48, 'int16'
      'fill2',                     50, 'int16' };

    hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
      'type', a(:,3) );
    hds.header_size = 512;
  otherwise
    message = 'Illegal file type';

  end % switch file_type

  hds.file_system = 'ecat7sh';

case 'ecat6.4'

  switch file_type
    % implemented:
    % 01=.scn, 02=.img, 03=.atn, 04=.nrm

  case 1 % .scn

    a = { ...
        'fill1',                    63, 'int16'
      'data_type',                   1, 'int16'
      'fill2',                      2, 'int16'
      'dimension_1',                 1, 'int16'
      'dimension_2',                 1, 'int16'
      'smoothing',                   1, 'int16'
```

```
    'processing_code',          1, 'int16'
    'fill3',                 3, 'int16'
    'sample_distance',          1, 'float32'
    'fill4',                 8, 'int16'
    'isotope_halflife',         1, 'float32'
    'frame_duration_sec',          1, 'int16'
    'gate_duration',            1, 'int32'
    'r_wave_offset',            1, 'int32'
    'fill5',                 1, 'int16'
    'scale_factor',             1, 'float32'
    'fill6',                 3, 'int16'
    'scan_min',                 1, 'int16'
    'scan_max',                 1, 'int16'
    'prompts',                  1, 'int32'
    'delayed',                  1, 'int32'
    'multiples',                1, 'int32'
    'net_trues',                1, 'int32'
    'fill7',                52, 'int16'
    'cor_singles',              16, 'float32'
    'uncor_singles',            16, 'float32'
    'tot_avg_cor',              1, 'float32'
    'tot_avg_uncor',            1, 'float32'
    'total_coin_rate',          1, 'int32'
    'frame_start_time',         1, 'int32'
    'frame_duration',           1, 'int32'
    'loss_correction_fctr'      1, 'float32'
    'fill8',                22, 'int16' };

  hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
    'type', a(:,3) );
  hds.header_size = 512;

case 2 % .img

  a = { ...
      'fill1',                 63, 'int16'
    'data_type',                1, 'int16'
    'num_dimensions',              1, 'int16'
    'fill2',                 1, 'int16'
    'dimension_1',              1, 'int16'
    'dimension_2',              1, 'int16'
    'fill3',                12, 'int16'
    'x_origin',                 1, 'float32'
    'y_origin',                 1, 'float32'
    'recon_scale',              1, 'float32'
    'quant_scale',              1, 'float32'
    'image_min',                1, 'int16'
    'image_max',                1, 'int16'
    'fill4',                 2, 'int16'
    'pixel_size',               1, 'float32'
    'slice_width',              1, 'float32'
    'frame_duration',              1, 'int32'
    'frame_start_time',            1, 'int32'
```

```
    'slice_location',              1, 'int16'
    'recon_start_hour',            1, 'int16'
    'recon_start_min',             1, 'int16'
    'recon_start_sec',             1, 'int16'
    'recon_duration',              1, 'int32'
    'fill5',                  12, 'int16'
    'filter_code',                 1, 'int16'
    'scan_matrix_num',                1, 'int32'
    'norm_matrix_num',                1, 'int32'
    'atten_cor_mat_num',              1, 'int32'
    'fill6',                  23, 'int16'
    'image_rotation',              1, 'float32'
    'plane_eff_corr_fctr',         1, 'float32'
    'decay_corr_fctr',             1, 'float32'
    'loss_corr_fctr',              1, 'float32'
    'fill7',                  32, 'int16'
    'processing_code',             1, 'int16'
    'fill8',                   1, 'int16'
    'quant_units',                 1, 'int16'
    'recon_start_day',             1, 'int16'
    'recon_start_month',           1, 'int16'
    'recon_start_year',            1, 'int16'
    'ecat_calibration_fctr',       1, 'float32'
    'well_counter_cal_fctr',       1, 'float32'
    'filter_params',               6, 'float32'
    'annotation',                 40, 'char'
    'fill9',                  26, 'int16' };

    hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
      'type', a(:,3) );
    hds.header_size = 512;

case 3 % .atn

  a = { ...
      'fill1',                  63, 'int16'
    'data_type',                   1, 'int16'
    'attenuation_type',               1, 'int16'
    'fill2',                   1, 'int16'
    'dimension_1',                    1, 'int16'
    'dimension_2',                    1, 'int16'
    'fill3',                  23, 'int16'
    'scale_factor',                1, 'float32'
    'x_origin',                   1, 'float32'
    'y_origin',                   1, 'float32'
    'x_radius',                   1, 'float32'
    'y_radius',                   1, 'float32'
    'tilt_angle',                 1, 'float32'
    'attenuation_coeff',              1, 'float32'
    'sample_distance',                1, 'float32'
    'fill4',                 149, 'int16' };

    hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
```

```
        'type', a(:,3) );
      hds.header_size = 512;

  case 4 % .nrm

    a = { ...
        'fill1',                63, 'int16'
      'data_type',                1, 'int16'
      'fill2',                  2, 'int16'
      'dimension_1',              1, 'int16'
      'dimension_2',              1, 'int16'
      'fill3',                 23, 'int16'
      'scale_factor',             1, 'float32'
      'norm_hour',                1, 'int16'
      'norm_minute',              1, 'int16'
      'norm_second',              1, 'int16'
      'norm_day',                 1, 'int16'
      'norm_month',               1, 'int16'
      'norm_year',                1, 'int16'
      'fov_source_width',           1, 'float32'
      'fill4',                155, 'int16' };

    hds.s = struct( 'fieldname', a(:,1), 'noelements', a(:,2), ...
      'type', a(:,3) );
    hds.header_size = 512;
  otherwise
    message = 'Illegal file type';

  end % switch file_type

  hds.file_system = 'ecat6.4sh';

otherwise
  message = 'Illegal file system';

end % switch file_system

function [ datadescr, message ] = getdatadescr( file_system, file_type, data_type );

datadescr = [];
message = '';

switch file_system

case 'ecat6.4'
  switch file_type

  case { 1, 2, 3, 4 }
    % 1 = .scn, 2 = .img, 3 = .atn, 4 = .nrm
    switch data_type
    case 2 % VAX_Ix2
      datadescr.ftype = 'int16';
      datadescr.size = 2;
```

```
    case 4 % vax float
      datadescr.ftype = 'float32';
      datadescr.size = 4;
    case 5 % ieee float
      datadescr.ftype = 'float32';
      datadescr.size = 4;
    otherwise
      message = 'Data type not implemented';
    end
  otherwise
    message = 'File type not implemented';
  end %switch file_type

case 'ecat7'

  % implemented:
  % 01=Sinogram, 03=Attenuation Correction, 04=Normalization,
  % 05=Polar Map, 07=Volume 16, 11=3D Sinogram 16, 13=3D Normalization

  % not implemented:
  % 00=unknown, 02=Image-16, 06=Volume 8, 08=Projection 8,
  % 09=Projection 16, 10=Image 8, 12=3D Sinogram 8, 14=3D Sinogram Fit
  switch file_type
  case { 11, 14 }
    switch data_type
    case 1 % byte
      datadescr.ftype = 'int8';
      datadescr.size = 1;
    case { 2, 6 } % Sun short, sun_in
      datadescr.ftype = 'int16';
      datadescr.size = 2;
    case 5 % ieee float
      datadescr.ftype = 'float32';
      datadescr.size = 4;
    otherwise
      message = 'Data type not implemented';
    end
  case 13
    switch data_type
    case 1 % ieee-float
      datadescr.ftype = 'float32';
      datadescr.size = 4;
    case 5 % sun-float
      datadescr.ftype = 'float32';
      datadescr.size = 4;
    otherwise
      message = 'Data type not implemented';
    end
  case { 1, 3, 4, 5, 7 }
    % (0=Unkonwn Matrix Data Type, 1=Byte Data, 2=VAX_Ix2, 3=VAX_Ix4,
    %  4=VAX_Rx4, 5=IEEE Float, 6=Sun short, 7=Sun long)
    % DTYPE_BYTES, _I2, _I4, _VAXR4, _SUNFL, _SUNIN
    switch data_type
```

```
    case 1 % byte
      datadescr.ftype = 'int8';
      datadescr.size = 1;
    case 6 % Sun short
      datadescr.ftype = 'int16';
      datadescr.size = 2;
    case 5 % ieee float
      datadescr.ftype = 'float32';
      datadescr.size = 4;
    otherwise
      message = 'Data type not implemented';
    end
  otherwise
    error( 'File type not implemented' );
  end %switch file_type

otherwise
  message = 'File system not implemented';

end % switch file_system

% ------------------------------------------------------------

function [ hd, offsetfromheader, writefill, numdatablocks, message ] = ...
    set_xyz_dimension( hd, datadescr, xyz_dimension, selsegment  );
% implemented:
% 01=Sinogram, 03=Attenuation Correction, 04=Normalization,
% 05=Polar Map, 07=Volume 16, 11=3D Sinogram 16, 13=3D Normalization

message = '';
offsetfromheader = 0;
writefill = 0;
numdatablocks = 0;

switch hd.mh.file_system
case 'ecat6.4'

  switch hd.mh.file_type
  case { 01, 02, 03, 04 } % .scn, .img, .atn, .nrm
    hd.sh.dimension_1 = xyz_dimension( 1 );
    hd.sh.dimension_2 = xyz_dimension( 2 );
    numdatablocks = floor( ...
      ( prod( xyz_dimension ) * datadescr.size + 511 ) / 512 );
    writefill = ( 511 - rem( prod( xyz_dimension ) * datadescr.size + 511, 512 ) );

  otherwise
    message = 'Writing of the file type not implemented';
    hd = [];

  end

case 'ecat7'
```

```
switch hd.mh.file_type
case 07 % Volume 16
  hd.sh.xyz_dimension = xyz_dimension;
  numdatablocks = floor( ...
    ( prod( xyz_dimension ) * datadescr.size + 511 ) / 512 );
  writefill = ( 511 - rem( prod( xyz_dimension ) * datadescr.size + 511, 512 ) );

case { 11, 14 } % 3D Sinogram 16, FORE sinogram float
  if hd.sh.storage_order == 0
    hd.sh.num_r_elements = xyz_dimension( 1 );
    hd.sh.num_z_elements( selsegment ) = xyz_dimension( 2 );
    hd.sh.num_angles = xyz_dimension( 3 );
  else
    hd.sh.num_r_elements = xyz_dimension( 1 );
    hd.sh.num_angles = xyz_dimension( 2 );
    hd.sh.num_z_elements( selsegment ) = xyz_dimension( 3 );
  end

  nobytes = hd.sh.num_r_elements * hd.sh.num_angles * datadescr.size * ...
    hd.sh.num_z_elements;
  %  numdatablocksarr = floor( ( nobytes + 511 ) / 512 );
  %  offsetfromheader = sum( numdatablocksarr( 1:selsegment-1 ) ) * 512;
  offsetfromheader = sum( nobytes( 1:selsegment-1 ) );

  %  numdatablocks = sum( numdatablocksarr );
  numdatablocks = floor( ( sum( nobytes ) + 511 ) / 512 );

  if sum( hd.sh.num_z_elements( selsegment + 1 : end ) ) == 0
    writefill = ( 511 - rem( prod( xyz_dimension ) * datadescr.size + offsetfromheader + 511, 512 ) );
  else
    writefill = 0;
  end

case 3 % Attenuation
  if hd.sh.storage_order == 0
    hd.sh.num_r_elements = xyz_dimension( 1 );
    hd.sh.z_elements( selsegment ) = xyz_dimension( 2 );
    hd.sh.num_angles = xyz_dimension( 3 );
  else
    hd.sh.num_r_elements = xyz_dimension( 1 );
    hd.sh.num_angles = xyz_dimension( 2 );
    hd.sh.z_elements( selsegment ) = xyz_dimension( 3 );
  end
  if selsegment == 1
    hd.sh.num_z_elements = hd.sh.z_elements( 1 );
  end

  nobytes = hd.sh.num_r_elements * hd.sh.num_angles * datadescr.size * ...
    hd.sh.z_elements;
  %  numdatablocksarr = floor( ( nobytes + 511 ) / 512 );
  %  offsetfromheader = sum( numdatablocksarr( 1:selsegment-1 ) ) * 512;
  offsetfromheader = sum( nobytes( 1:selsegment-1 ) );
```

```
    %  numdatablocks = sum( numdatablocksarr );
    numdatablocks = floor( ( sum( nobytes ) + 511 ) / 512 );

    if sum( hd.sh.num_z_elements( selsegment + 1 : end ) ) == 0
      writefill = ( 511 - rem( prod( xyz_dimension ) * datadescr.size + offsetfromheader + 511, 512 ) );
    else
      writefill = 0;
    end

  otherwise
    message = 'Writing of the file type not implemented';
    hd = [];

  end

otherwise
  message = 'File system not implemented';
  hd = [];

end % switch hd.mh.file_system

% ------------------------------------------------------------
function [ xyz_dimension, offsetfromheader, message ] = get_xyz_dimension( hd, datadescr, selsegment )
% implemented:
% 01=Sinogram, 03=Attenuation Correction, 04=Normalization,
% 05=Polar Map, 07=Volume 16, 11=3D Sinogram 16, 13=3D Normalization

message = '';
xyz_dimension = [];
offsetfromheader = 0;

switch hd.mh.file_system
case 'ecat7'

  switch hd.mh.file_type
  case 07 % Volume 16
    xyz_dimension = hd.sh.xyz_dimension;

  case { 11, 14 } % 3D Sinogram 16, FORE sinogram float
    if hd.sh.storage_order == 0
      xyz_dimension = [ hd.sh.num_r_elements, ...
          hd.sh.num_z_elements( selsegment ), hd.sh.num_angles ];
    else
      xyz_dimension = [ hd.sh.num_r_elements, hd.sh.num_angles, ...
          hd.sh.num_z_elements( selsegment ) ];
    end
    nobytes = hd.sh.num_r_elements * hd.sh.num_angles * datadescr.size * ...
      hd.sh.num_z_elements;
    %  numdatablocksarr = floor( ( nobytes + 511 ) / 512 );
    %  offsetfromheader = sum( numdatablocksarr( 1:selsegment-1 ) ) * 512;
    offsetfromheader = sum( nobytes( 1:selsegment-1 ) );

  case 3 % Attenuation
```

```
    if hd.sh.storage_order == 0
      xyz_dimension = [ hd.sh.num_r_elements, ...
          hd.sh.z_elements( selsegment ), hd.sh.num_angles ];
    else
      xyz_dimension = [ hd.sh.num_r_elements, hd.sh.num_angles, ...
          hd.sh.z_elements( selsegment ) ];
    end
    nobytes = hd.sh.num_r_elements * hd.sh.num_angles * datadescr.size * ...
      hd.sh.z_elements;
    %  numdatablocksarr = floor( ( nobytes + 511 ) / 512 );
    %  offsetfromheader = sum( numdatablocksarr( 1:selsegment-1 ) ) * 512;
    offsetfromheader = sum( nobytes( 1:selsegment-1 ) );

  otherwise
    message = 'Reading of the file type not implemented';

  end

case 'ecat6.4'

  switch hd.mh.file_type
  case { 01, 02, 03, 04 } % .scn, .img, .atn, .nrm
    xyz_dimension = [ hd.sh.dimension_1, hd.sh.dimension_2, 1 ];

  otherwise
    message = 'Reading of the file type not implemented';

  end

otherwise
  message = 'File system not implemented';

end % switch hd.mh.file_system

% ------------------------------------------------------------

function n = firstno( sel )
n = [ find( sel(:) ); 0 ];
n = n( 1 );

% ------------------------------------------------------------
```

G: MRI2RAW.M : CONVERTS MRI IMAGES INTO RAW FORMAT SUITABLE FOR SERA

```
close all; clear all;
%                                                    MRI2RAW.m
%                                    Code to Read MRI image & Convert to 8 bit RAW format image for BNCT



sprintf('%s', '*********************************')
sprintf('%s', 'This is written by Mohammad K. Khan')
sprintf('%s', '       Copyright  February, 1999   ')
sprintf('%s',  '*********************************')

clear all
nn='MRI2RAW.in';
nn=input('Please enter the name of input file(MRI2RAW.in) ?', 's');
        inputfid = fopen(nn, 'r');
                %CHECK for the existence of file in the current directory
                 while (inputfid == -1)
                        sprintf('Attempt to find MRI2RAW.in in the current directory failed... Please try
again')
                        nn=input(' \n Enter the path to input (MRI2RAW.in) file ? \n', 's');
                        inputfid = fopen(nn, 'r');
                 end

        hh = 0;  %this is a counter that will count how many image files to read
              % and get pixel size
        % Read the contour file names from the input file
        global pixelsize plotsize FigureSelect INELfactor
        n=fgetl(inputfid);
        nt=fgetl(inputfid);
        nt=fgetl(inputfid);
        nt =fgetl(inputfid);
        nt =fgetl(inputfid);
        nt = fgetl(inputfid);
D = pwd;
nt=deblank(nt);
cd(nt);



while(feof(inputfid) ~= 1)
        hh= hh+1;
        n1=fgetl(inputfid);
                if n1 == -1,
                        break;
                end
        nt=deblank(n1); % Gets the image file names from the input file
        imagefiles(hh) = {nt};   %names of all the mri/pet image files
        imagefid=fopen(nt, 'r');
```

```
        %sprintf('Working on %s file... Please be patient! ..', nt)
        while (imagefid == -1)
                sprintf('Attempt to find %s failed... Please try again', nt)
                nt=input(' \n please enter the path to that file ? \n', 's');
                imagefid = fopen(nt, 'r');
        end

        say = fread(imagefid, 'int16');

        k=0;
        for i = 1:256,
                for j = 1:256,
                        k=k+1;
                        A(i,j) = say(3952+k);  % Header Lines NOT NEEDED

                end
        end
        imagematrix(:,1+(hh-1)*256:256*hh)=A;
        fclose(imagefid);
end

cd(D);
grayimages= ind2gray(imagematrix, gray(256));
        if min(min(imagematrix)') <0;
                grayimages = ind2gray(imagematrix/2, gray(256));
        end

figure(1)
montage(grayimages, [256 256 hh]);
sprintf('Image planes are number 1..2..3..4 .. etc right to left and into next row')
number = input('please select an image plane to convert from MRI to BMP 8 BIT ? ');

IM = grayimages(:, 1+(number-1)*256:256*number);
IM = flipud(IM);
nt = imagefiles(number);
nt2 = strcat(nt, '.img');
nt = strcat(nt, '.bmp');
output = fopen(char(nt2), 'wb');
imwrite(IM*255, gray(255), char(nt), 'bmp');
fwrite(output, IM'*255, 'int8');  % Convert to Raw 8 bit image


figure(2);
imshow(IM);
cd(D);
multiple= input('Would you like to convert other images Planes (n/y)?','s');
while(multiple == 'y' | 'Y')
        number = input('\n please select an image plane to convert from RAW to Bitmap ? ');
        IM = grayimages(:, 1+(number-1)*256:256*number);
        IM = flipud(IM);
        nt = imagefiles(number);
        nt2 = strcat(nt, '.img');
        nt = strcat(nt, '.bmp');
```

```
        output = fopen(char(nt2), 'wb');
        imwrite(IM*255, gray(255), char(nt), 'bmp');
        fwrite(output, IM'*255, 'int8');  % Convert to Raw 8 bit image

        figure(2);
        imshow(IM);
        cd(D);
        multiple= input('\n Would you like to continue converting other images Planes (n/y)?','s')
                if ((multiple == 'n') | (multiple == 'N'))
                        break;
                end
end
```

H: INPUT FILE FOR MRI2RAW.M

```
          MRICONVERT.in (R_in.m)
THIS IS AN INPUT FILE TO MRI2RAW.m: A code to read raw 8 bit
raw MRI/PET images
and convert a selected image into a raw format.

HERE IS A LIST OF PET/MRI IMAGES THAT I WOULD TO SEE:
/ne5/bnct/Godwin/MRI/10_15_97/
E5039S3I1.MR
E5039S3I2.MR
E5039S3I3.MR
E5039S3I4.MR
E5039S3I5.MR
E5039S3I6.MR
E5039S3I7.MR
E5039S3I8.MR
E5039S3I9.MR
E5039S3I10.MR
E5039S3I11.MR
E5039S3I12.MR
E5039S3I13.MR
E5039S3I14.MR
E5039S3I15.MR
E5039S3I16.MR
E5039S3I17.MR
E5039S3I18.MR
E5039S3I19.MR
E5039S3I20.MR
E5039S3I21.MR
```

I: READ_REGISTRATION.M

```
% program to Read Registered PET/MRI Images and to write out individual MRI Image Planes for SERA

close all; clear all;
%Phdr = 'rBates_c0_2e1_de7.hdr';
Phdr = input('Enter Name of registered PET header (.hdr) file in Analyze format ? ', 's'),
%Mhdr = 'T1_FLAIR2.hdr';
Mhdr = input('Enter Name of registered MRI header (.hdr) file in Analyze Format ? ', 's');
Pimg = strcat(Phdr(1:size(Phdr,2)-3), 'img');
Mimg = strcat(Mhdr(1:size(Mhdr,2)-3), 'img');

PFID = fopen(Pimg, 'r');
MFID = fopen(Mimg, 'r');

path(path, '/ne1/mkhan/registration/SPM_99/spm99');

[DIM,VOX,SCALE,TYPE,OFFSET,ORIGIN,DESCRIP ] = spm_hread(Phdr);
[MDIM, MVOX, MSCALE, MTYPE, MOFFSET, MORIGIN, MDESCRIP ] = spm_hread(Mhdr);

P=zeros(DIM);
M=zeros(MDIM);
type = spm_type(TYPE);
Mtype = spm_type(MTYPE);

for i = 1:DIM(3),
   figure(i);
    subplot(2,1, 1)
    A=fread(PFID, [DIM(1) DIM(2)], type);
    P(1:256, 1:256, i) = A;
    minA = min(min(A)); maxA = max(max(A));
    imshow((A-minA)/(maxA-minA));

    subplot(2,1,2)
    MA = fread(MFID, [MDIM(1) MDIM(2)], Mtype);
    M(1:256, 1:256, i) = MA;
    minMA = min(min(MA)); maxMA = max(max(MA));
    imshow((MA-minMA)/(maxMA-minMA));hold on;
    contour((A));


    Mimgout = strcat(Mimg, num2str(i));
    FID = fopen(Mimgout, 'w');
    fwrite(FID, MA, 'uint8');
    fclose(FID);

end
```

J: ECAT2SERA.M

```
%program to Convert ECAT 6.4 or 7.0 DATA into SERA Compatible Data
%It Strips out the last frame of study and all of the planes of the study
%Written by Mohammad Khurram Khan (c) copyright 2002

clear all; close all; fclose('all'); clear; pack;fclose('all');
filename= '/ne5/bnct/bnct2/BNCT_RAW_DATA/PET/Rose/Oct98/Rose_c7_796_de5.v';
%filename = '/ne5/bnct/bnct2/BNCT_RAW_DATA/PET/Gaddis/Nov95/gad1110j1d.img';
%filename= 'E:\BNCT_RAW_DATA\Gaddis\Pet\Nov95\gad1110j1d.img';
%filename=input('Enter Location and ECAT 6.4 Image FIle Name (E:\BNCT_RAW_DATA\Gaddis\..) ? ',
's');
Image_format = input('enter 1 for ECAT6.4 data and 2 for ECAT7.0 data format ? ');

if Image_format == 1
   file_system = 'ecat6.4' ;
   fpname = filename;
   [fid, message] = ecatfile('open', filename, file_system);
   [ mh, message ]       = ecatfile( 'mh', fid );
   selmatrix= [2 47 1 0 0];    %frame number num_planes
   [ vol, hd, message ]   = ecatfile( 'read', fid, selmatrix );
   I=zeros(hd.mh.num_planes, 128, 128);
   Psera = zeros(256,256);

  for i = 1:mh.num_planes,
    selmatrix= [hd.mh.num_frames i 1 0 0]   %frame number & plane number
    [ vol, hd, message]   = ecatfile( 'read', fid, selmatrix );
    I(i, 1:128, 1:128) = vol(1:128, 1:128);
     say = double(vol);
   Pimgout = strcat('Plane', num2str(i), '.img');
   FID = fopen(Pimgout, 'w');
   Psera = imresize(say, [256 256], 'bilinear');
   Psera = int16(Psera);
   fwrite(FID, Psera, 'int16');
   fclose(FID);
  end


elseif Image_format == 2
   file_system = 'ecat7';
   fpname = filename;
  [ fid, message ]      = ecatfile( 'open', filename, 'ecat7' );
  [ mh, message ]       = ecatfile( 'mh', fid );
  selmatrix= [1 1 1 0 0];    %frame number num_planes
  [ vol, hd, message ]   = ecatfile( 'read', fid, selmatrix );
  [ matranges, message ] = ecatfile( 'matranges', fid );
  I=zeros(mh.num_frames, 128, 128, hd.mh.num_planes);
```

```
    selmatrix=[mh.num_frames 1 1 0 0];
    [vol, hd, message]=ecatfile('read', fid, selmatrix);
      for i = 1:hd.mh.num_planes,
          PImage = squeeze(vol(:, :, i));
          say = double(PImage);
          Pimgout = strcat('Plane', num2str(i), '.img');
          FID = fopen(Pimgout, 'w');
          Psera = imresize(say, [256 256], 'bilinear');
          Psera = int16(Psera);
          fwrite(FID, Psera, 'int16');
           fclose(FID);
      end


else
   fprintf(' Could Not Recognize File Data');
end
```

K: EXAMPLE OF SERACALC INPUT FILE

```
VG
    2000     1000       1.000000E-02
     4.1100000E+00     -2.7300000E+00      9.7500000E+00
     1.8750000E+01     -6.5000000E+01     -1.4500000E+02
/nel/mkhan/BNCT/Sera/ST/Results/cg.geom
/nel/mkhan/registration/SPM_99/Gaddis/VG.uv
      30       3
      32       72       94
   -1.50000000E+01    -1.50000000E+01    -1.50000000E+01
    1.00000000E+00        0
none
/nel/mkhan/registration/SPM_99/Gaddis/VGsera.rst
/nel/mkhan/BNCT/Sera/ST/Results/BS.source
       1        0
NFGD             27_Nov_2001
$SERA_HOME/Resources/SeraMC/seraMC.mat
$SERA_HOME/Resources/SeraMC/seraMC.sigma
 521 508 513 512 505   0   0 6.0143E-08 6.3150E-02 7.9110E-04 7.2700E-03 2.6800E-02 0.0000E+00
0.0000E+00
run_seraMC

iop
 3 APERTURE BUFFER
 0.50

edit_dir
#patient VG UTMCK T1 MRI
#date May_9_1975
#beam BMRR_12_CM
 mw_min 1.0
 b10_blood 1.0
 refvol 1.0 0
 ref_reg brain tumor target
 ottocon 40 40 scalp tumor target brain
 delta 0.25
 eps 0.05
#Dose-Volume Histogram for Brain
 in_reg brain target
 b10_blood 13
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
#Dose-Volume Histogram for tumor
 in_reg tumor target
 b10_blood 45.5
 rbe 1.3 1.0 3.2 3.2 1.0 1.0
 DVbs
#Dose-Volume Histogram for Scalp
 in_reg scalp
 b10_blood 19
```

rbe 1.3 1.0 3.2 3.2 1.0 1.0
DVbs

L: EULER_TRANSFORM8.M PROGRAM TO IMPLEMENT RIGID BODY TRANSFORMATION/TRANSLATION

```
% This is a Non Linear Optimisation Code to Carry out Coordinate Transformation and Translation
% And to Find the angles of transformation.  This code is developed by
% Mohammad Khurram Khan.  Copy Righted 2000.
% LAST UPDATE 3/3/2000
% This Code is Extremely Robust

% Generation of Synthetic Data %
        x = rand(1, 10);
        y = rand(1, 10);
        z = rand(1, 10);

        originalpoints = [x; y; z];
        X = originalpoints;

        t = .2*pi;  % I theta (pitc)
        p = .25*pi;  %  phi  (Yaw)
        s = .3*pi;  %  psy  (Roll)
        T = [40 20 50];
        T = [T(1,1)*ones(1,size(x,2)); T(1,2)*ones(1,size(y,2)); T(1,3)*ones(1,size(z,2))];;

        R = [(cos(p)*cos(t)*cos(s)-sin(p)*sin(s)) (-cos(p)*cos(t)*sin(s)-sin(p)*cos(s)) cos(p)*sin(t),
      (sin(p)*cos(t)*cos(s)+cos(p)*sin(s)) (-sin(p)*cos(t)*sin(s)+cos(p)*cos(s)) sin(p)*sin(t),
          -sin(t)*cos(s)              sin(t)*sin(s)              cos(t)  ];


        Y = R*X+T;

 X0 = [ 0 0 0];
 T0 = [ 0 0 0];

 XG = [X0, T0];


% NONLINEAR UNCOSTRAINED (DIFFERENT FUNCTION)  (THIS WORKS ALSO)
options=optimset('LargeScale', 'off', 'Diagnostics', 'on', 'Display', 'iter', 'LevenbergMarquardt', 'on');
%[X1, fval, exitflag, output]= fminunc('objfun8', XG, options, X, Y)



lb = [-pi -pi -pi -256 -256 -256];
ub = [ pi  pi  pi  256  256  256 ];
[X1, fval, exitflag, output]= fmincon('objfun8', XG, [],[],[],[],lb, ub, [], options, X, Y)
T0 = [X1(4:6)];  %By Now I pretty much have my Translation Vector Nailed and will use the
          % and Use it again as input for the next iteration

% For Robustness, I feed X1 Results back into fmincon Just for a second iteration
XG = [X0, T0];
[X1, fval, exitflag, output]= fmincon('objfun8', XG, [],[],[],[],lb, ub, [], options, X, Y)
```

```
%     ********************     OPTIMISATION     and     TRANSFORMATION     FINISHED
**********************************
%
*****************************************************************************************
*********


% Re GENERATION OF NEWPOINTS AFTER THE THREE ANGLES ARE FOUND
 t1 = X1(1);
 p1= X1(2);
 s1= X1(3);
 Xt = X1(4);
 Yt = X1(5);
 Zt = X1(6);
T = [Xt Yt Zt];
T = [T(1,1)*ones(1,size(x,2)); T(1,2)*ones(1,size(y,2)); T(1,3)*ones(1,size(z,2))];;



 % FINAL MATRIX ROTATION MATRIX
        R1     =     [(cos(p1)*cos(t1)*cos(s1)-sin(p1)*sin(s1))     (-cos(p1)*cos(t1)*sin(s1)-sin(p1)*cos(s1))
cos(p1)*sin(t1),
       (sin(p1)*cos(t1)*cos(s1)+cos(p1)*sin(s1)) (-sin(p1)*cos(t1)*sin(s1)+cos(p1)*cos(s1)) sin(p1)*sin(t1),
          -sin(t1)*cos(s1)              sin(t1)*sin(s1)              cos(t1)  ];


% TO GET TO ORIGINAL UNTRANSFORMED POINTS:  points = R1'*Y
% TO GET TO TRANSFORMED POINTS  :        points = R1*X

points = (R1)'*(Y-T);
error = (points - originalpoints)./(originalpoints)*100;
```

```
 function F = objfun7(XG, X, Y)

 t1 = XG(1);
 p1= XG(2);
 s1= XG(3);
 Xt= XG(4);
 Yt = XG(5);
 Zt = XG(6);

  T = [Xt Yt Zt];
  T = [T(1,1)*ones(1,size(X,2)); T(1,2)*ones(1,size(X,2)); T(1,3)*ones(1,size(X,2))];


 % Guess Matrix

        R1     =     [(cos(p1)*cos(t1)*cos(s1)-sin(p1)*sin(s1))     (-cos(p1)*cos(t1)*sin(s1)-sin(p1)*cos(s1))
cos(p1)*sin(t1),
       (sin(p1)*cos(t1)*cos(s1)+cos(p1)*sin(s1)) (-sin(p1)*cos(t1)*sin(s1)+cos(p1)*cos(s1)) sin(p1)*sin(t1),
```

```
        -sin(t1)*cos(s1)        sin(t1)*sin(s1)        cos(t1) ];

  % Objective Function

  T_pnts = R1'*(Y-T);    % To take new points back into original axis (X)

  F = 0;

for k = 1:size(Y,1),
  for i = 1:size(Y,2)
        F = (T_pnts(k,i) - X(k,i) )^2 +F;
 end
end
F;
```

Results:

```
>> originalpoints'

ans =

   0.4514   0.6085   0.0841
   0.0439   0.0158   0.4544
   0.0272   0.0164   0.4418
   0.3127   0.1901   0.3533
   0.0129   0.5869   0.1536
   0.3840   0.0576   0.6756
   0.6831   0.3676   0.6992
   0.0928   0.6315   0.7275
   0.0353   0.7176   0.4784
   0.6124   0.6927   0.5548
```

Transformation/Translation Vector

[.2*pi .25*pi .3*pi 40 20 50];

[.6283 .7854 .9425 40 20 50];

Constrained Optimisation Results:

X1 =

0.6283 0.7853 0.9426 40.0000 20.0000 50.0000

Points' Determined based on initial guess of
[0 0 0 0 0 0]:

points'

ans =

0.4514 0.6085 0.0841
0.0439 0.0158 0.4544
0.0271 0.0164 0.4418
0.3126 0.1901 0.3533
0.0128 0.5869 0.1536
0.3839 0.0576 0.6757
0.6831 0.3676 0.6992
0.0928 0.6315 0.7275
0.0353 0.7176 0.4784
0.6124 0.6927 0.5549

% error'

ans =

-0.0090 0.0003 0.0194
-0.0921 0.0117 0.0036
-0.1487 0.0113 0.0037
-0.0129 0.0010 0.0046
-0.3142 0.0003 0.0106
-0.0105 0.0032 0.0024
-0.0059 0.0005 0.0023

-0.0435	0.0003	0.0022
-0.1144	0.0003	0.0034
-0.0066	0.0003	0.0029

M: MRI2RAW.M

```
close all; clear all;
%                                                MRI2RAW.m
%                                Code to Read MRI image & Convert to 8 bit RAW format image for BNCT


sprintf('%s', '*********************************')
sprintf('%s', 'This is written by Mohammad K. Khan')
sprintf('%s', '       Copyright  February, 1999   ')
sprintf('%s', '*********************************')

clear all
nn='MRI2RAW.in';
nn=input('Please enter the name of input file(MRI2RAW.in) ?', 's');
        inputfid = fopen(nn, 'r');
                %CHECK for the existence of file in the current directory
                 while (inputfid == -1)
                        sprintf('Attempt to find MRI2RAW.in in the current directory failed... Please try
again')
                        nn=input(' \n Enter the path to input (MRI2RAW.in) file ? \n', 's');
                        inputfid = fopen(nn, 'r');
                 end

        hh = 0;  %this is a counter that will count how many image files to read
              % and get pixel size
        % Read the contour file names from the input file
        global pixelsize plotsize FigureSelect INELfactor
        n=fgetl(inputfid);
        nt=fgetl(inputfid);
        nt=fgetl(inputfid);
        nt =fgetl(inputfid);
        nt =fgetl(inputfid);
        nt = fgetl(inputfid);
D = pwd;
nt=deblank(nt);
cd(nt);



while(feof(inputfid) ~= 1)
        hh= hh+1;
        n1=fgetl(inputfid);
                if n1 == -1,
                        break;
                end
        nt=deblank(n1); % Gets the image file names from the input file
        imagefiles(hh) = {nt};   %names of all the mri/pet image files
        imagefid=fopen(nt, 'r');
```

```
        %sprintf('Working on %s file... Please be patient! ..', nt)
        while (imagefid == -1)
                sprintf('Attempt to find %s failed... Please try again', nt)
                nt=input(' \n please enter the path to that file ? \n', 's');
                imagefid = fopen(nt, 'r');
        end

        say = fread(imagefid, 'int16');

        k=0;
        for i = 1:256,
                for j = 1:256,
                        k=k+1;
                        A(i,j) = say(3952+k);  % Header Lines NOT NEEDED

                end
        end
        imagematrix(:,1+(hh-1)*256:256*hh)=A;
        fclose(imagefid);
end

cd(D);
grayimages= ind2gray(imagematrix, gray(256));
        if min(min(imagematrix)') <0;
                grayimages = ind2gray(imagematrix/2, gray(256));
        end

figure(1)
montage(grayimages, [256 256 hh]);
sprintf('Image planes are number 1..2..3..4 .. etc right to left and into next row')
number = input('please select an image plane to convert from MRI to BMP 8 BIT ? ');

IM = grayimages(:, 1+(number-1)*256:256*number);
IM = flipud(IM);
nt = imagefiles(number);
nt2 = strcat(nt, '.img');
nt = strcat(nt, '.bmp');
output = fopen(char(nt2), 'wb');
imwrite(IM*255, gray(255), char(nt), 'bmp');
fwrite(output, IM'*255, 'int8');  % Convert to Raw 8 bit image


figure(2);
imshow(IM);
cd(D);
multiple= input('Would you like to convert other images Planes (n/y)?','s');
while(multiple == 'y' | 'Y')
        number = input('\n please select an image plane to convert from RAW to Bitmap ? ');
        IM = grayimages(:, 1+(number-1)*256:256*number);
        IM = flipud(IM);
        nt = imagefiles(number);
        nt2 = strcat(nt, '.img');
        nt = strcat(nt, '.bmp');
```

```
        output = fopen(char(nt2), 'wb');
        imwrite(IM*255, gray(255), char(nt), 'bmp');
        fwrite(output, IM'*255, 'int8');  % Convert to Raw 8 bit image

        figure(2);
        imshow(IM);
        cd(D); .
        multiple= input('\n Would you like to continue converting other images Planes (n/y)?','s')
                if ((multiple == 'n') | (multiple == 'N'))
                        break;
                end
end
```

N: PET2MRI.M

```
%Program to read registered PET/MRI Images and to Work with SERA Output
%To Post Process the Dose based on BNL Methodology and Dose Based on boron
%biodistribution as determined from a 18F-BPA PET Image

clear all; close all;
INELfactor=[45.5 13 19];  %BNL Scaling Factor for Dose
[FID, FID1, FID3, FID4, FID5, FID6, fname, fname1, fname3, uvval, r_name, Phdr, Pimg] =
file_locator;
[P_n, Mo, SO, Dim, Byt, Ndim, Sd0, Sd1, Sd2, Rloc, Xp, Yp, Zp, readas ]=qhd_read(FID);
[C, CC, z_v, xdim, ydim, l]= cdf_read2(FID3,fname3);

I=fread(FID4, [Sd1 Sd2*Sd0], 'uint8');     %Image Matrix
%I=fread(FID4,[Sd1 Sd2*Sd0], readas);      % Image Matrix
R=fread(FID1, [Sd1 Sd2*Sd0], 'uint8');    % Corresponding Regions Information
P=Reg_petread(FID6, Phdr);                % PET Image Matrix

minI = min(min(I)); maxI = max(max(I));
I2 = (I-minI)/(maxI-minI);
A=zeros(1:xdim, 1:ydim);
k=0;
for i = 1:xdim,
  for j = 1:ydim,
    k=k+1;
    A(i,j)=k;
  end
end

f=max(CC(:,6));    V= [5 4.5 4 3.5 3  2 1.75 1.5 1.25 1 .9 .8 .7 .6 .5 .4 .3 .2];      % controls %
Contour Labels
[X, Y] = meshgrid(1:xdim, 1:ydim);
[Xi, Yi] = meshgrid(1:Sd1, 1:Sd2);

ba=zeros(size(I2));
bo=zeros(size(I2));bb=zeros(1:256, 1:256, Sd0);

   for i = 1:l,
%step through each plane
  b=CC((i-1)*xdim*ydim+1:(i*xdim*ydim),6) ;b =b(A);
  b=imresize(b, [256 256],  'bilinear');
```

```
  bb(1:256, 1:256, i)=b;                                                    %256X256
Contour Grid Size
 [ba(1:256,  256*(i-1)+1:256*(i)),  bo(1:256,  256*(i-1)+1:256*(i))]=peak_to_average(Sd1,
Sd2, R, I2, P, i, uvval, r_name, b, INELfactor);
 % ba = PET2PET adjusted dosimetry; % bo = BNL adjusted dosimetry
  end
  %save ba ba;
  %save bo bo;
  %save bb bb;

bmax=max(max(max(bb)));  %maximum dose (cgy/mw-min/ppm) and unscaled boron dose
f=max(max(ba));
f2=max(max(bo));


  V= [ .9 .7 .5 .3 .1];
  Table1 = zeros(l, 4);
  Table2 = zeros(1,4);

for i = 1:l,
  format bank;
  figure(i);
  subplot(1,2,1)
  %imshow(rot90(I2(1:256, 256*(i-1)+1:256*(i))), 256); colorbar('horiz'); hold on;
  subimage(flipud((rot90((I2(1:256, 256*(i-1)+1:256*(i)))))));  %colorbar('horiz');
  bs=flipud(bb(1:256, 1:256, i)/bmax);                  %scale dose to BNL (for contour display)
  if ( max(max(bs)) >= .2 )
     format bank;
      V= [ .9 .7 .5 .3 .1];
   hold on;  [cs, h] = contour(bs, V);clabel(cs,h,V, 'fontsize', 10, 'color', 'r'); hold off;
  end
  title('Original B-10 Dose Contours '); colormap('bone');

  subplot(1,2,2);
  [xx yy] = meshgrid(1:256);
  plotat = .75*(min(min(bs*bmax))-max(max(bs*bmax)));
  h=surf(xx,yy,bs*bmax);
  set(h, 'Edgecolor', 'blue');
  hold on; %colormap('bone');
  I=flipud((rot90((I2(1:256, 256*(i-1)+1:256*(i))))));
  h2 = warp(1:256, 1:256, plotat*ones(size(bs*bmax)), I);
  hold off; colormap('bone');
  set(gca, 'zlim', [plotat max(max(bs*bmax))]); title('Original B-10 Dose Rate'); zlabel('cGy-
Eq/mw-min/ppm');
```

```
figure(30+i);
subplot(1,2,1)
title('MRI/PET Registered Scaled Boron Dose Rate');
subimage(flipud(rot90(I2(1:256, 256*(i-1)+1:256*(i))))); %colorbar('horiz');
%bas = ba(1:256, 256*(i-1)+1:256*(i))/bmax;
 bas = flipud((ba(1:256, 256*(i-1)+1:256*(i))))/f;

 [II, JJ]=find(bas>.9); say=size(II,1); say2=size(JJ,2);
area9=(say*str2num(Xp)*say2*str2num(Yp));
 [II2, JJ2]=find(bas>.7); say=size(II2,1); say2=size(JJ2,2);
area7=(say*str2num(Xp)*say2*str2num(Yp));
 [II3, JJ3]=find(bas>.5); say=size(II3,1); say2=size(JJ3,2);
area5=(say*str2num(Xp)*say2*str2num(Yp));
 [II3, JJ3]=find(bas>.3); say=size(II3,1); say2=size(JJ3,2);
area3=(say*str2num(Xp)*say2*str2num(Yp));

 Table1(i,1:4)=[area9 area7 area5 area3];

if (max(max(bas))>=.2 )
    format bank; v1=.7*f/f2;
   V= [.9 .7 .5 .3 .1];
hold on; [cs, h] = contour(bas, V);h=clabel(cs,h,V, 'fontsize', 10, 'color', 'r'); hold off;
end

subplot(1,2,2);
 [xx yy] = meshgrid(1:256);
 plotat = .75*(min(min(bas*f))-max(max(bas*f)));
 h=surf(xx,yy,bas*f);
 set(h, 'Edgecolor', 'blue');
 hold on; %colormap('bone');
 I=flipud((rot90((I2(1:256, 256*(i-1)+1:256*(i))))));
 h2 = warp(1:256, 1:256, plotat*ones(size(bas*f)), I);
 hold off; colormap('bone');
 set(gca, 'zlim', [plotat max(max(bas*f))]); title('MR1/PET Registered Scaled B-10 Dose');
zlabel('cGy-Eq/mw-min')


figure(60+i)
subplot(1,2,1)
title('Homogeneously Scaled Boron Dose');
subimage(flipud(rot90(I2(1:256, 256*(i-1)+1:256*(i))))); %colorbar('horiz');
```

```
  bos = flipud((bo(1:256, 256*(i-1)+1:256*(i))))/f2;
  if (max(max(bos))>=.2)
  V= [.9 .7 .5 .3 .1];
  hold on;  [cs, h] = contour(bos, V);h=clabel(cs,h,V, 'fontsize', 8, 'color', 'r'); hold off;
  end

  [II,        JJ]=find(bos>.9);          say=size(II,1);          say2=size(JJ,2);
area9=(say*str2num(Xp)*say2*str2num(Yp));
  [II2,       JJ2]=find(bos>.7);         say=size(II2,1);         say2=size(JJ2,2);
area7=(say*str2num(Xp)*say2*str2num(Yp));
  [II3,       JJ3]=find(bos>.5);         say=size(II3,1);         say2=size(JJ3,2);
area5=(say*str2num(Xp)*say2*str2num(Yp));
  [II3,       JJ3]=find(bos>.3);         say=size(II3,1);         say2=size(JJ3,2);
area3=(say*str2num(Xp)*say2*str2num(Yp));

  Table2(i,1:4)=[area9 area7 area5 area3];


  subplot(1,2,2);
  [xx yy] = meshgrid(1:256);
  plotat = .75*(min(min(bos*f2))-max(max(bos*f2)));
  h=surf(xx,yy,bos*f2);
  set(h, 'Edgecolor', 'blue');
  hold on; %colormap('bone');
  I=flipud((rot90((I2(1:256, 256*(i-1)+1:256*(i))))));
  h2 = warp(1:256, 1:256, plotat*ones(size(bos*f2)), I);
  hold off; colormap('bone');
  set(gca, 'zlim', [plotat max(max(bos*f2))]); title('Homogeneously Scaled B-10
Dose');zlabel('cGy-Eq/MW-min');


end

save Table1 Table1;
save Table2 Table2;

save Table1 Table1 -Ascii;
save Table2 Table2 -Ascii;
```

O:PEAK_TO_AVERAGE.M

```
function [ba, bo] = peak_to_average(Sd1, Sd2, R, I2, P, i, uvval, r_name, b, INELfactor)
% Compute Average over regions;
ii = i;
b=b;
bo=b;
reg=rot90(R(1:256, (i-1)*256+1:256*i)); b_10=b;
imagematrix=rot90(P(1:256, (i-1)*256+1:256*i)*255);  %Use PET Pixel Intensity to Scale the Dose Not the MRI
Image
MRIimage  =rot90(I2(1:256, (i-1)*256+1:256*i)*255);
buffer=strmatch('  buffer', r_name); buffer = uvval(buffer);
tumor_region=strmatch('  tumor', r_name);tumor_region= uvval(tumor_region);
brain_region=strmatch('  brain', r_name);brain_region= uvval(brain_region);
scalp_region=strmatch('  scalp', r_name); scalp_region=uvval(scalp_region);
target_region=strmatch('  target', r_name); target_region=uvval(target_region);
n1=0; sum1=0;
n2=0; sum2=0;
n3=0; sum3=0;
n4=0; sum4=0;

 tumorimage = zeros(256); targetimage=zeros(256); targetregion = zeros(256); tumorregion=zeros(256);
%first compute average intensities in various regions
 %for k = 1:2,                              %two iterations/ second scale the PET dose
  for i=1:256,
          for j=1:256,
     if (reg(i,j)==target_region)              %Target Region (main tumor mass)
        n4 = n4+1;
        sum4= sum4+imagematrix(i,j);           %target region PET pixel intensity sum
        targetimage(i,j)=imagematrix(i,j);
        targetregion(i,j)=reg(i,j);
        bo(i,j)=b(i,j)*INELfactor(1)/INELfactor(2); %Target Region BNL Dosimetry scaling
     end
                    if (reg(i,j)==tumor_region)              % Tumor Region
                          n1=n1+1;
        sum1=sum1+imagematrix(i,j);                                  % Tumor Region PET Pixel Intensity Sum
        bo(i,j)=b(i,j)*INELfactor(1)/INELfactor(2); % Tumor Region BNL Dosimetry Scaling
        tumorimage(i,j)=imagematrix(i,j);
        tumorregion(i,j)=reg(i,j);
    end
    if ( reg(i,j)== brain_region)              % Brain Region
                        n2=n2+1;
        sum2= sum2+imagematrix(i,j);                  % Brain Region PET Pixel Intensity Sum
        bo(i,j)=b(i,j)*INELfactor(2)/INELfactor(2); % Brain Region BNL Dosimetry
    end

    if(reg(i,j)==scalp_region) % Scalp/Skull Region
         n3=n3+1;
        sum3=sum3+imagematrix(i,j);                                  %Scalp/Skull PET Pixel Intensity
Sum
        bo(i,j) = b(i,j)*INELfactor(3)/INELfactor(2);  %Scalp/Skull Region BNL Dosimetry
            end
```

```
 end
        end
        tumoraverage= sum1/n1;        %tumor average PET pixel intensity
 targetaverage = sum4/n4;       %target average PET pixel intensity
        brainaverage= sum2/n2;        %brain average PET pixel intensity
 totalbrainaverage= (tumoraverage+targetaverage+brainaverage)/3;
 scalpskullav = sum3/n3;       %Scalp/Skull PET pixel intensity average
        entireregion= sum(sum(imagematrix)')/(256*256);

 adjusteddose2=zeros(256);
        adjusteddose2=b_10.*imagematrix/totalbrainaverage;   %Peak To Average Scaling Based on PET
Tumor Average
 ba=adjusteddose2;

 figure(100);
  subplot(6, 7, ii);
   imshow(MRIimage,[100 200]); hold on;
   contour(imagematrix); hold off;

  subplot(6,7, ii+21);
   imshow(targetimage/256); hold on;
   contour(tumorimage);
   hold off;
```

P: QHD_READ.M

```
function [P_n, Mod, SO, Dim, Byt, Ndim, Sdim0, Sdim1, Sdim2, Rloc, Xp, Yp, Zp, readas ]=qhd_read(FID)



i=0;
while feof(FID) ~= 1
  i=i+1;
  f = fgetl(FID);
  h = findstr(f, '=')+1;
        if  strmatch('Patient Name', f)==1
                                        P_n = f(h:size(f,2));
        elseif strmatch('Modality',f)==1;
          Mod = f(h:size(f,2));
        elseif strmatch('Slice Orientation', f)==1;
          SO = f(h:size(f,2));
        elseif strmatch('Number of Dimensions', f)==1;
          Dim = f(h:size(f,2));
        elseif strmatch('Bytes', f)==1;
          Byt= f(h:size(f,2));
        elseif strmatch('Number of Dimension', f) == 1;
          Ndim = f(h:size(f,2));
        elseif strmatch('Size of Dimension[0]', f)== 1;
          Sdim0 = f(h:size(f,2));    %Number of Z Planes
        elseif strmatch('Size of Dimension[1]', f)==1;
          Sdim1 = f(h:size(f,2));
```

```
        elseif strmatch('Size of Dimension[2]', f) == 1;
          Sdim2 = f(h:size(f,2));
        elseif strmatch('Refernce Location', f) == 1;
          Rloc = f(h:size(f,2));
        elseif strmatch('Uniform Spacing', f)==1 ;
            Usp = f(h:size(f,2));
        elseif strmatch('X Pixel Size', f)==1;
          Xp = f(h:size(f,2));
        elseif strmatch('Y Pixel Size', f) == 1;
          Yp = f(h:size(f,2));
        elseif strmatch('Image Location[0]',f) == 1,
            for k = 1:str2num(Sdim0)-1,
              if f == -1,
                break
              end
            Zp(k) = str2num(f(h:size(f,2)));
            f = fgetl(FID);
                    h = findstr(f, '=')+1;
          end
         break
       end
end


By = str2num(Byt);
Sdim1 = str2num(Sdim1);
Sdim2 = str2num(Sdim2);
```

```
Sdim0=str2num(Sdim0);


      fclose(FID);


if By == 1
  readas='int8';
elseif By == 2
  readas='int16';
else
 % readas='int8';
  sprintf('I Do Not Know the structure of the image file for sure')
end


      return
```

Q: CDF_WRITE.M

```
function [] = cdf_write(FID3,fname3, xdim, ydim, l, ba, CC)

CA=zeros(xdim*ydim*l, 1);
cdf_name = strcat(fname3(1:size(fname3,2)-4), 'adjusted.cdf');
cdf_name2 = strcat(fname3(1:size(fname3,2)-4), 'BNL.cdf');

FID = fopen(cdf_name, 'w');
FID2 = fopen(cdf_name2, 'w');

format short e;
L=0;
   for i = 1:l,                                %step through each plane
    b = imresize(ba(1:256, 256*(i-1)+1:256*i), [xdim ydim], 'bilinear');
     for k = 1:xdim,
        for j = 1:ydim,
            L = L+1;
            CA(L,1) = b(k,j);
            hi = isnan(b(k,j));
           if (hi == 1)
              CA(L,1) = hi-1;
           end



        end
     end
   end




 for i = 1:xdim*ydim*l,
 %C=fscanf(FID3, '%10e2, %10e2, %10e2, %5c %10e2 %10e2 %10e2 %10e2 %10e2 %10e2 %10e2 %10e2
%10e2', 125)

 %C = C';
 x = num2str(CC(i,1), '%6.3E');
   if size(x,2) < 10
      fprintf(FID, ' %s', x);
     fprintf(FID2, ' %s', x);
  else
      fprintf(FID, '%s', x);
     fprintf(FID2, '%s', x);
   end

 y = num2str(CC(i,2), '%6.3E');
  if size(y,2) <10
     fprintf(FID, ' %s', y);
    fprintf(FID2, ' %s', y);
  else
```

```
    fprintf(FID, '%s', y);
    fprintf(FID2, '%s', y);
  end

z = num2str(CC(i,3), '%6.3E');
  if size(z, 2) < 10
    fprintf(FID, ' %s', z);
    fprintf(FID2, ' %s', z);
  else
    fprintf(FID, '%s', z);
    fprintf(FID2, '%s', z);
  end

r = num2str(CC(i,4));
fprintf(FID, '   %s', r);
fprintf(FID2,'   %s', r);

t = num2str(CC(i,5), '%6.3E');
  if size(t, 2) < 10
    fprintf(FID, ' %s', t);
    fprintf(FID2,' %s', t);
  else
    fprintf(FID, '%s', t);
    fprintf(FID2,'%s', t);
  end

b = num2str(CA(i,1), '%6.3E');
b2 = num2str(CC(i,6),'%6.3E');
  if size(b, 2) < 10
    fprintf(FID, ' %s', b);
    fprintf(FID2,' %s', b2);
  else
    fprintf(FID, '%s', b);
    fprintf(FID2,'%s',b2);
  end


g = num2str(CC(i,7), '%6.3E');
  if size(g, 2) < 10
    fprintf(FID, ' %s', g);
    fprintf(FID2, ' %s', g);

  else
    fprintf(FID, '%s', g);
    fprintf(FID2,'%s', g);
  end

n = num2str(CC(i,8), '%6.3E');
  if size(n, 2) < 10
    fprintf(FID, ' %s', n);
    fprintf(FID2, ' %s', n);
  else
    fprintf(FID, '%s', n);
```

```
    fprintf(FID2, '%s', n);
  end
f = num2str(CC(i,9), '%6.3E');
  if size(f, 2) < 10
    fprintf(FID, ' %s', f);
    fprintf(FID2, ' %s', f);
  else
    fprintf(FID, '%s', f);
    fprintf(FID2, '%s', f);
  end

  g1 = num2str(CC(i,10), '%6.3E');
  if size(g1, 2) < 10
    fprintf(FID, ' %s', g1);
    fprintf(FID2, ' %s', g1);
  else
    fprintf(FID, '%s', g1);
    fprintf(FID2, '%s', g1);
  end

  g2 = num2str(CC(i,11), '%6.3E');
  if size(g2, 2) < 10
    fprintf(FID, ' %s', g2);
    fprintf(FID2, ' %s', g2);
  else
    fprintf(FID, '%s', g2);
    fprintf(FID2, '%s', g2);

  end

  th = num2str(CC(i,12), '%6.3E');
  if size(th, 2) < 10
    fprintf(FID, ' %s', th);
    fprintf(FID2, ' %s', th);
  else
    fprintf(FID, '%s', th);
    fprintf(FID2, '%s', th);
  end

  o = num2str(CC(i,13), '%6.3E');
  if size(o, 2) < 10
    fprintf(FID, ' %s\n', o);
    fprintf(FID2, ' %s\n', o);
  else
    fprintf(FID, '%s\n', o);
    fprintf(FID2, '%s\n', o);
  end


end


 fclose(FID);
```

R: FILE_LOCATOR.M

```
function [FID, FID1, FID3, FID4, FID5, FID6, fname, fname1, fname3, uvval, r_name, Phdr, Pimg] = file_locator


fname= input('please enter name of .qhd header file name ?', 's');
%fname = 'G1_MRI_Presurgery.qhd';
while isempty(findstr('.qhd', fname)) == 1
  fname = input('please enter a valid .qhd file name ?', 's');
end

FID = fopen(fname);
while FID == -1
  FID = fopen(fname);
  if FID == -1
    sprintf('Could Not Locate File in the present directory')
    fname = input('please enter name of qhd header file ?', 's')
  end
end

imname=strcat(fname(1:size(fname,2)-3), 'qim');
FID4=fopen(imname);
while FID4 == -1
  FID4 = fopen(imname);
  if FID4 == -1
    sprintf('Could Not Locate File in the present directory')
    imname = input('please enter name of image (.qim) file ?', 's')
  end
end


fname2= input('please enter name of .uvh file  ?', 's');
%fname1='G1_MRI_Presurgery.uvh'
while isempty(findstr('.uvh', fname2)) == 1
  fname2 = input('please enter a valid .uvh file name ?','s');
end

FID2 = fopen(fname2);

while FID2 == -1
  FID2 = fopen(fname2);
  if FID2 == -1
    sprintf('Could Not Locate File in the present directory')
    fname2 = input('please enter name of .uvh file ?', 's')
  end
end

fname1=fname2(1, 1:size(fname2,2)-1);
FID1=fopen(fname1);

while FID1 == -1
```

```
  FID1 = fopen(fname1);
  if FID1 == -1
    sprintf('Could Not Locate File in the present directory')
    fname1 = input('please enter name of .uv file ?', 's')
  end
end


fname3=input('please enter name of .cdf file ?', 's');
%fname3 = 'G1_MRI_Presurgery.cdf';
while isempty(findstr('.cdf', fname3)) == 1
  fname3 = input('please enter a valid .cdf file name ?', 's');
end

FID3 = fopen(fname3);
while FID3 == -1
  FID3 = fopen(fname3);
  if FID3 == -1
    sprintf('Could Not Locate File in the present directory')
    fname3 = input('please enter name of .cdf file ?', 's')
  end
end



[uvval, r_name] = uvh_read(FID2);


% get Name of SPM Registered PET Image File

Phdr= input('please enter name of registered PET image in Analyze Format (.hdr) ?', 's');

while isempty(findstr('.hdr', Phdr)) == 1
  Phdr = input('please enter a valid PET Analyze .hdr file name ?', 's');
end

FID5 = fopen(Phdr);
while FID5 == -1
  FID5 = fopen(Phdr);
  if FID5 == -1
    sprintf('Could Not Locate the registered PET Analyze Format File in the present directory')
    Phdr = input('please enter name of a valid PET .hdr header file ?', 's')
  end
end

Pimg=strcat(Phdr(1:size(Phdr,2)-3), 'img');
FID6 = fopen(Pimg);
while FID6 == -1
  FID6 = fopen(Pimg);
  if FID6 == -1
    sprintf('Could Not Locate the registered PET Analyze Format File in the present directory')
    Pimg = input('please enter name of a valid PET .hdr header file ?', 's')
  end
```

```
end



function [uvval, r_name]=uvh_read(FID2)

r=0;
while feof(FID2)==0

  line = fgetl(FID2);
    i=strmatch('begin:', line);
    if isempty(i)==0
      r=r+1;
      %input('found begin')
      r_name(r)=cellstr(line(1, 7:size(line,2)) );
      line = fgetl(FID2);
      uvval(r) = str2num(line(1, 10:size(line,2)));
    end
  end
```

S: CDF_WRITE.M

```
function [] = cdf_write(FID3,fname3, xdim, ydim, l, ba, CC)

CA=zeros(xdim*ydim*l, 1);
cdf_name = strcat(fname3(1:size(fname3,2)-4), 'adjusted.cdf');
cdf_name2 = strcat(fname3(1:size(fname3,2)-4), 'BNL.cdf');

FID = fopen(cdf_name, 'w');
FID2 = fopen(cdf_name2, 'w');

format short e;
L=0;
   for i = 1:l,                                %step through each plane
    b = imresize(ba(1:256, 256*(i-1)+1:256*i), [xdim ydim], 'bilinear');
     for k = 1:xdim,
        for j = 1:ydim,
            L = L+1;
            CA(L,1) = b(k,j);
            hi = isnan(b(k,j));
           if (hi == 1)
              CA(L,1) = hi-1;
           end



        end
     end
   end




 for i = 1:xdim*ydim*l,
 %C=fscanf(FID3, '%10e2, %10e2, %10e2, %5c %10e2 %10e2 %10e2 %10e2 %10e2 %10e2 %10e2 %10e2
%10e2', 125)

 %C = C';
 x = num2str(CC(i,1), '%6.3E');
   if size(x,2) < 10
      fprintf(FID, ' %s', x);
     fprintf(FID2, ' %s', x);
  else
      fprintf(FID, '%s', x);
     fprintf(FID2, '%s', x);
   end

 y = num2str(CC(i,2), '%6.3E');
  if size(y,2) <10
     fprintf(FID, ' %s', y);
    fprintf(FID2, ' %s', y);
```

```
  else
     fprintf(FID, '%s', y);
     fprintf(FID2, '%s', y);
   end

 z = num2str(CC(i,3), '%6.3E');
   if size(z, 2) < 10
     fprintf(FID, ' %s', z);
     fprintf(FID2, ' %s', z);
  else
     fprintf(FID, '%s', z);
     fprintf(FID2, '%s', z);
  end

r = num2str(CC(i,4));
fprintf(FID, '   %s', r);
fprintf(FID2,'   %s', r);

 t = num2str(CC(i,5), '%6.3E');
   if size(t, 2) < 10
     fprintf(FID, ' %s', t);
     fprintf(FID2,' %s', t);
  else
     fprintf(FID, '%s', t);
     fprintf(FID2,'%s', t);
  end

 b = num2str(CA(i,1), '%6.3E');
 b2 = num2str(CC(i,6),'%6.3E');
   if size(b, 2) < 10
     fprintf(FID, ' %s', b);
     fprintf(FID2,' %s', b2);
  else
     fprintf(FID, '%s', b);
     fprintf(FID2,'%s',b2);
  end


 g = num2str(CC(i,7), '%6.3E');
   if size(g, 2) < 10
     fprintf(FID, ' %s', g);
     fprintf(FID2, ' %s', g);

  else
     fprintf(FID, '%s', g);
     fprintf(FID2,'%s', g);
  end

n = num2str(CC(i,8), '%6.3E');
  if size(n, 2) < 10
     fprintf(FID, ' %s', n);
     fprintf(FID2, ' %s', n);
  else
```

```
    fprintf(FID, '%s', n);
    fprintf(FID2, '%s', n);
  end
f = num2str(CC(i,9), '%6.3E');
  if size(f, 2) < 10
    fprintf(FID, ' %s', f);
    fprintf(FID2, ' %s', f);
  else
    fprintf(FID, '%s', f);
    fprintf(FID2, '%s', f);
  end

  g1 = num2str(CC(i,10), '%6.3E');
  if size(g1, 2) < 10
    fprintf(FID, ' %s', g1);
    fprintf(FID2, ' %s', g1);
  else
    fprintf(FID, '%s', g1);
    fprintf(FID2, '%s', g1);
  end

  g2 = num2str(CC(i,11), '%6.3E');
  if size(g2, 2) < 10
    fprintf(FID, ' %s', g2);
    fprintf(FID2, ' %s', g2);
  else
    fprintf(FID, '%s', g2);
    fprintf(FID2, '%s', g2);

  end

  th = num2str(CC(i,12), '%6.3E');
  if size(th, 2) < 10
    fprintf(FID, ' %s', th);
    fprintf(FID2, ' %s', th);
  else
    fprintf(FID, '%s', th);
    fprintf(FID2, '%s', th);
  end

  o = num2str(CC(i,13), '%6.3E');
  if size(o, 2) < 10
    fprintf(FID, ' %s\n', o);
    fprintf(FID2, ' %s\n', o);
  else
    fprintf(FID, '%s\n', o);
    fprintf(FID2, '%s\n', o);
  end
end
fclose(FID);
```

T: DOSE CONTOURS AND 3D IMAGES OF OTHER PATIENTS.

Patient BP:

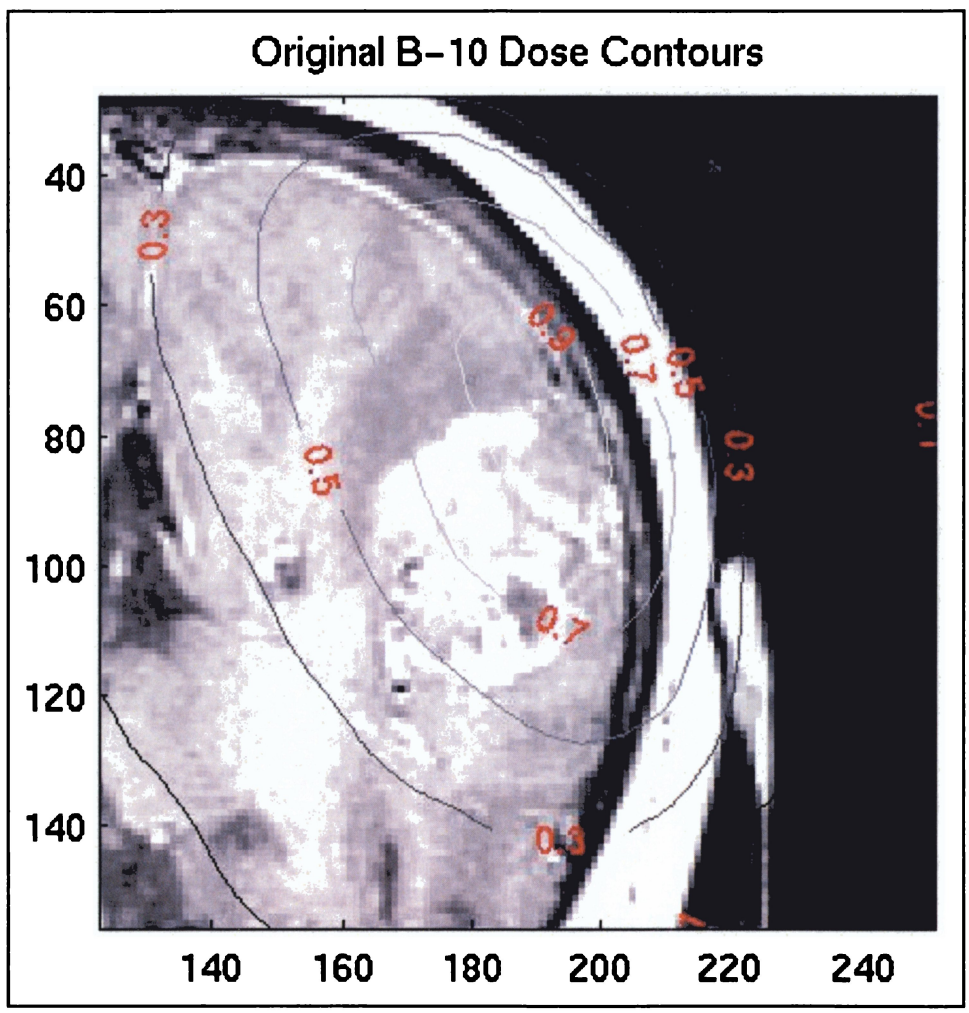


Figure 1A: Unscaled/Original Boron Dose Contours from SERA For Patient BP

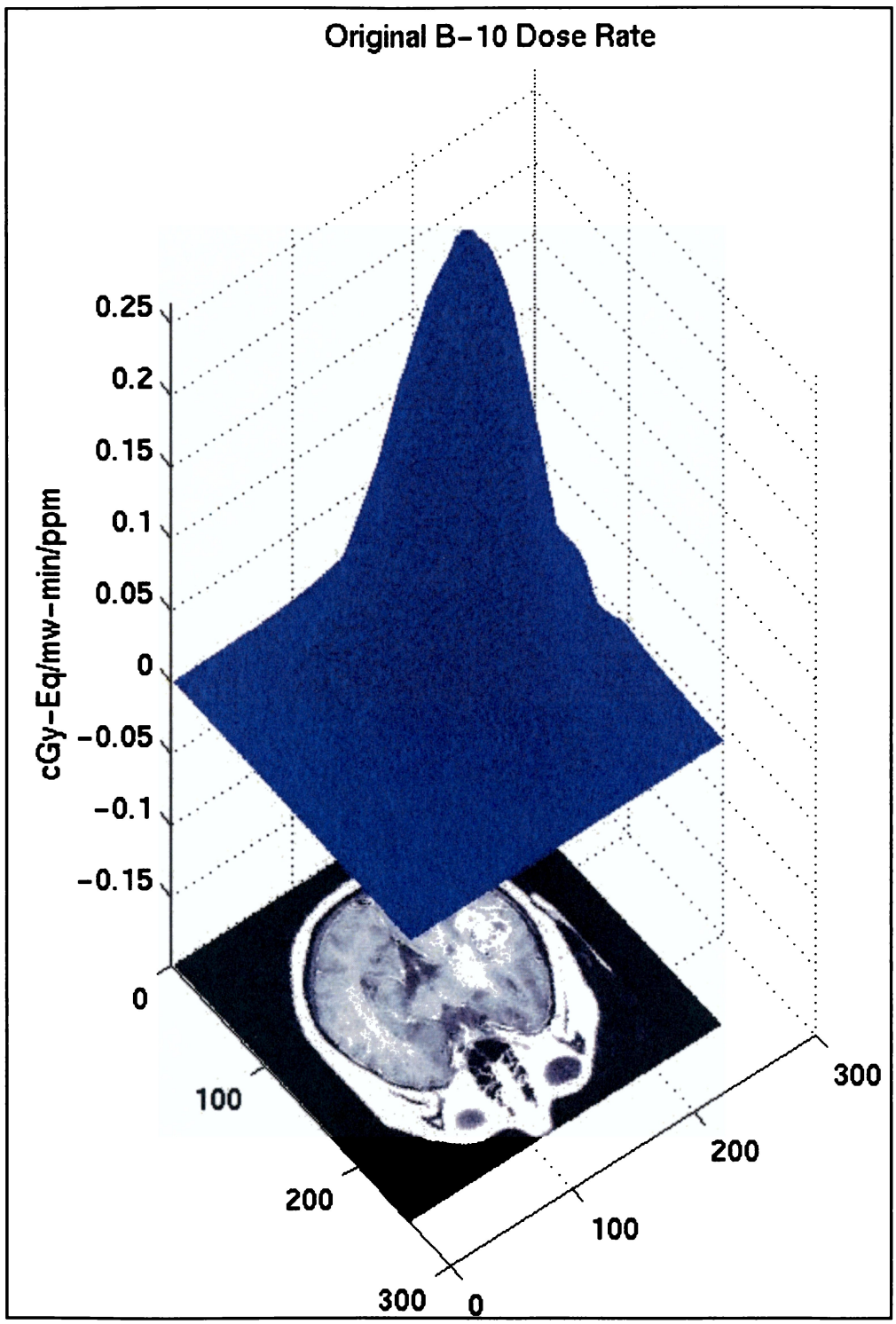


Figure 1B: 3D View of Unscaled/Original Boron Dose (Dose Rate) From SERA for Patient BP

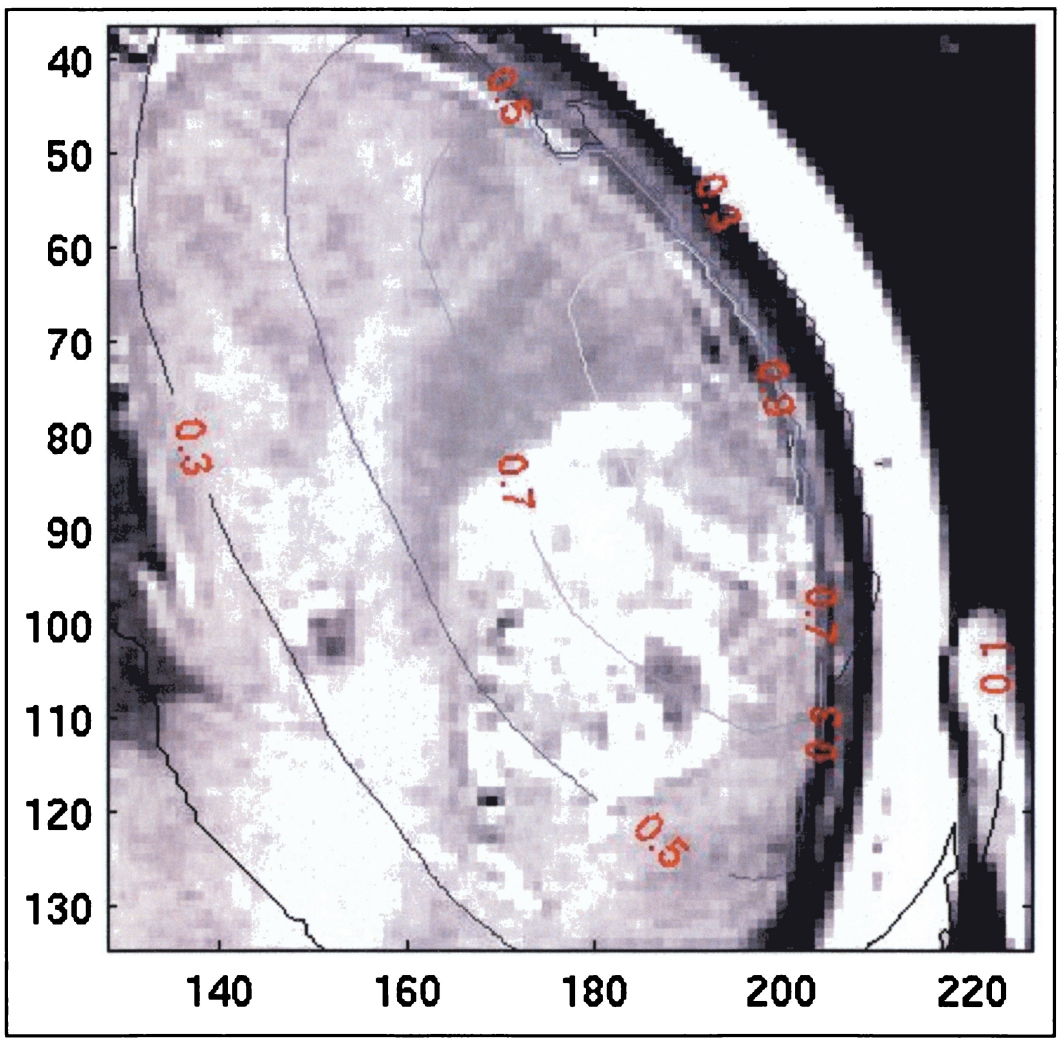


Figure 2A: Homogeneous Scaled Boron Dose Contours (Patient BP)

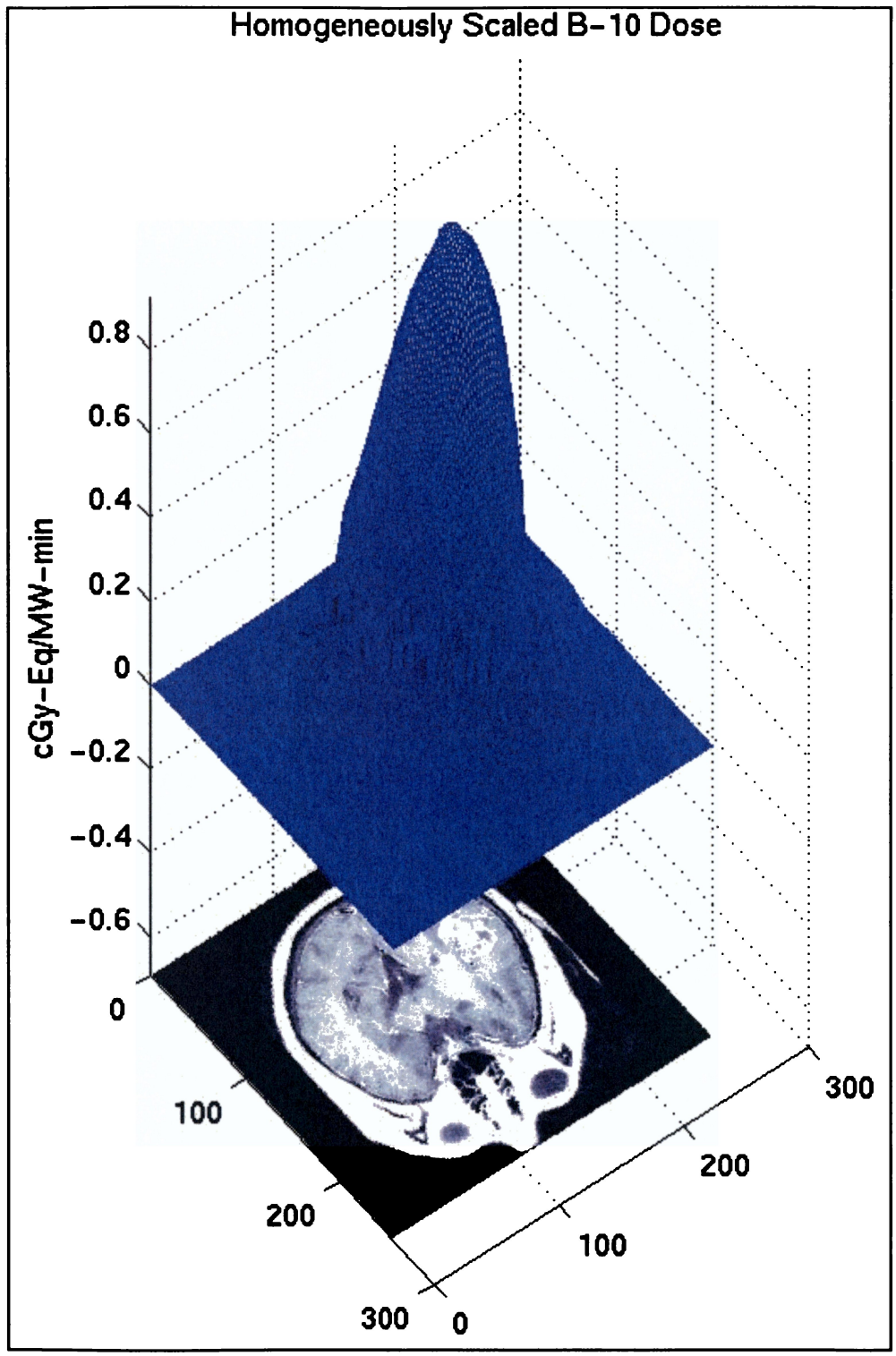


Figure 2B: 3D View of The Homogeneously Scaled Boron Dose (Dose Rate) Contours for Patient BP

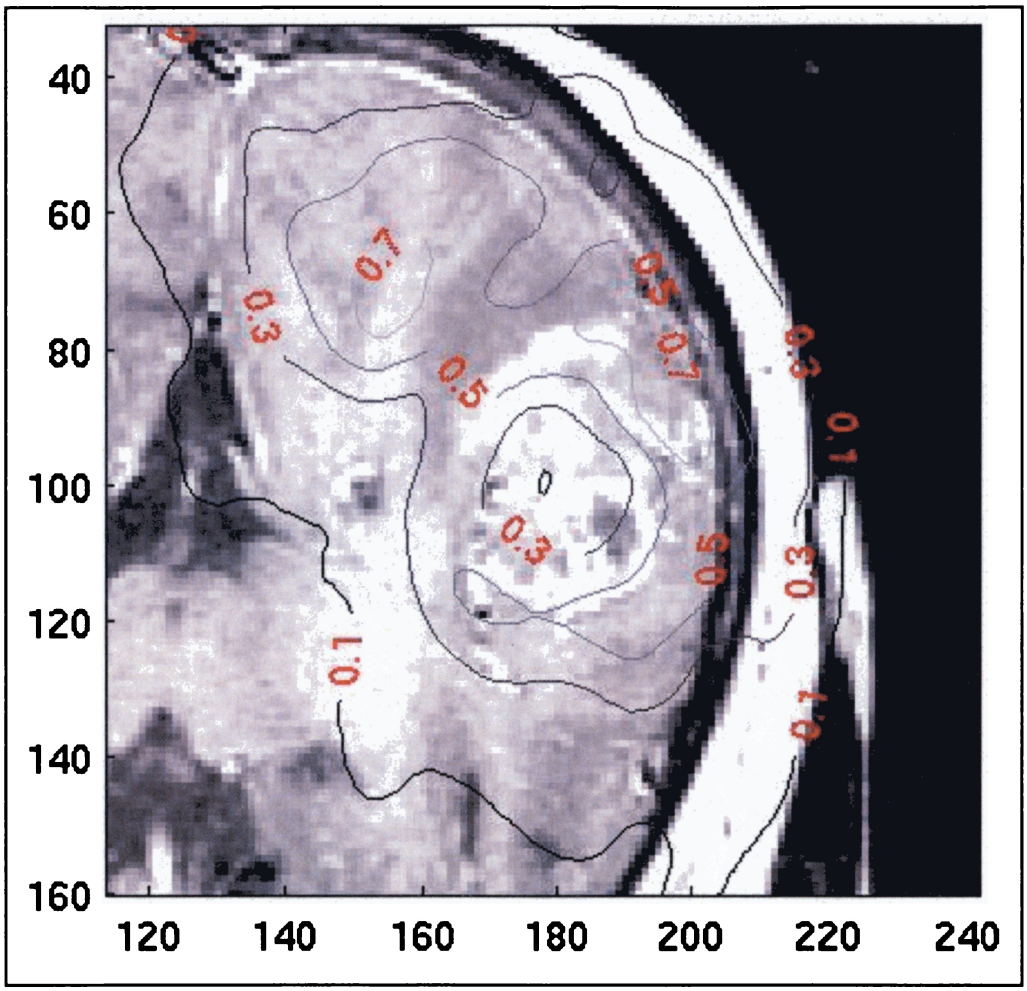


Figure 3A: PET Adjusted Boron Dose contours for Patient BP

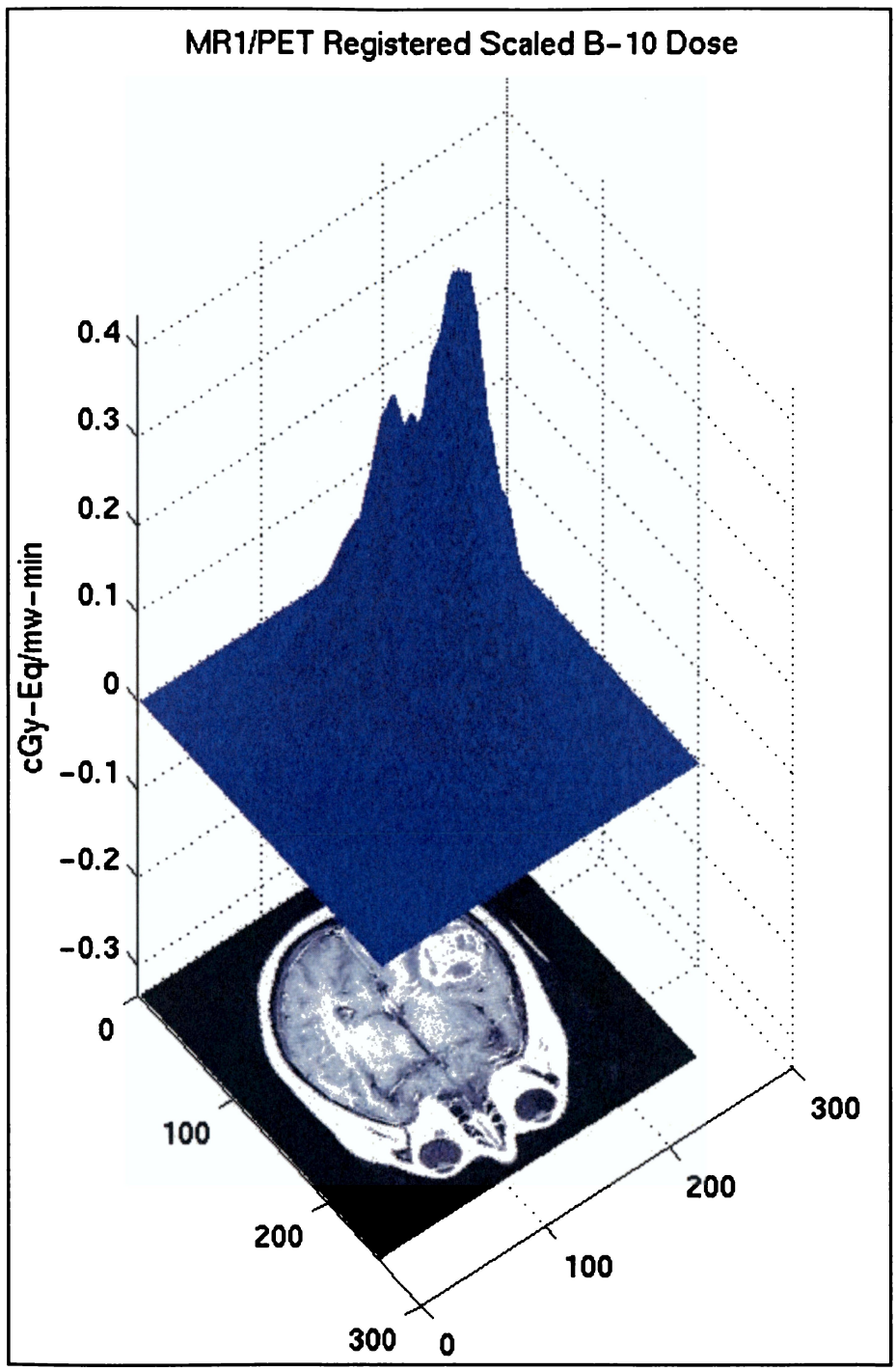


Figure 3B: 3D View of the PET Adjusted Boron Dose Rate (or Dose) for Patient BP

Patient EG:

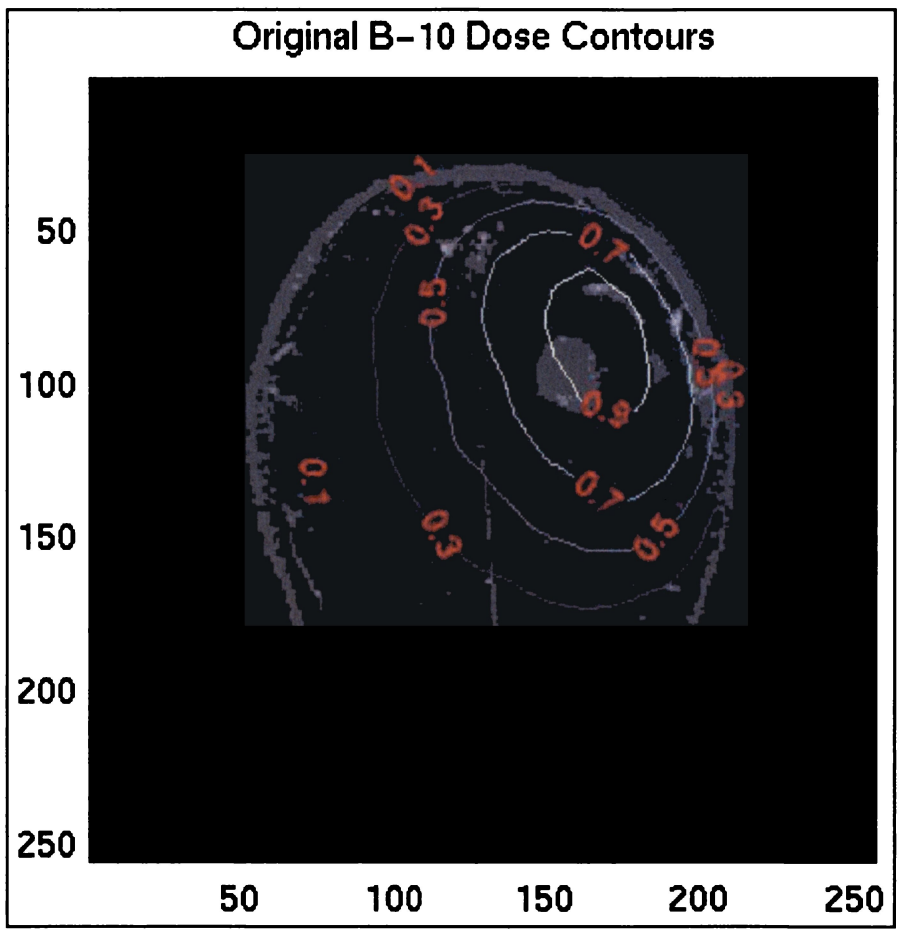


Figure 4A: Original Boron Dose Contours for Patient EG

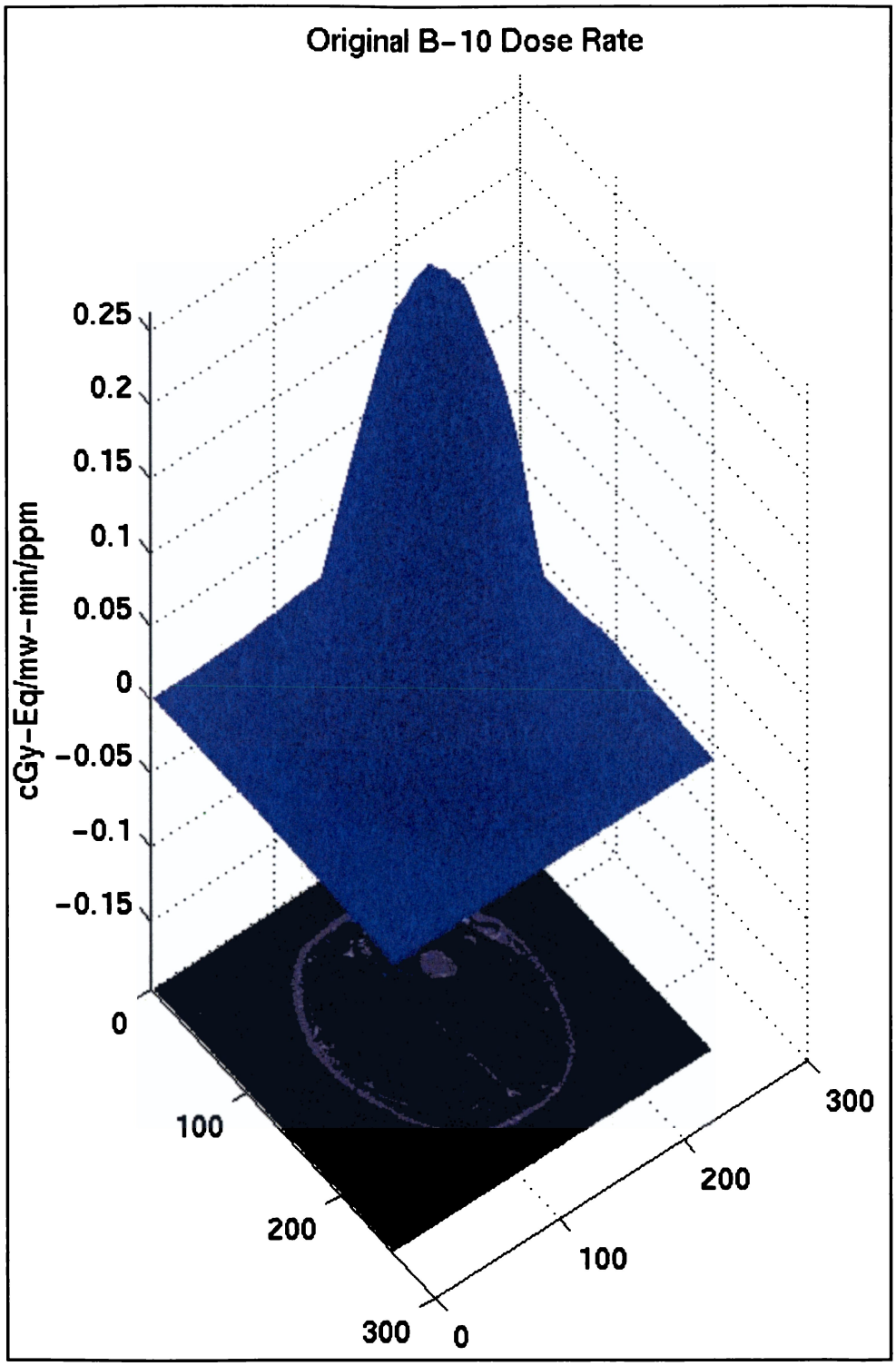


Figure 4B: 3D View of Original Boron Dose Rate for Patient EG

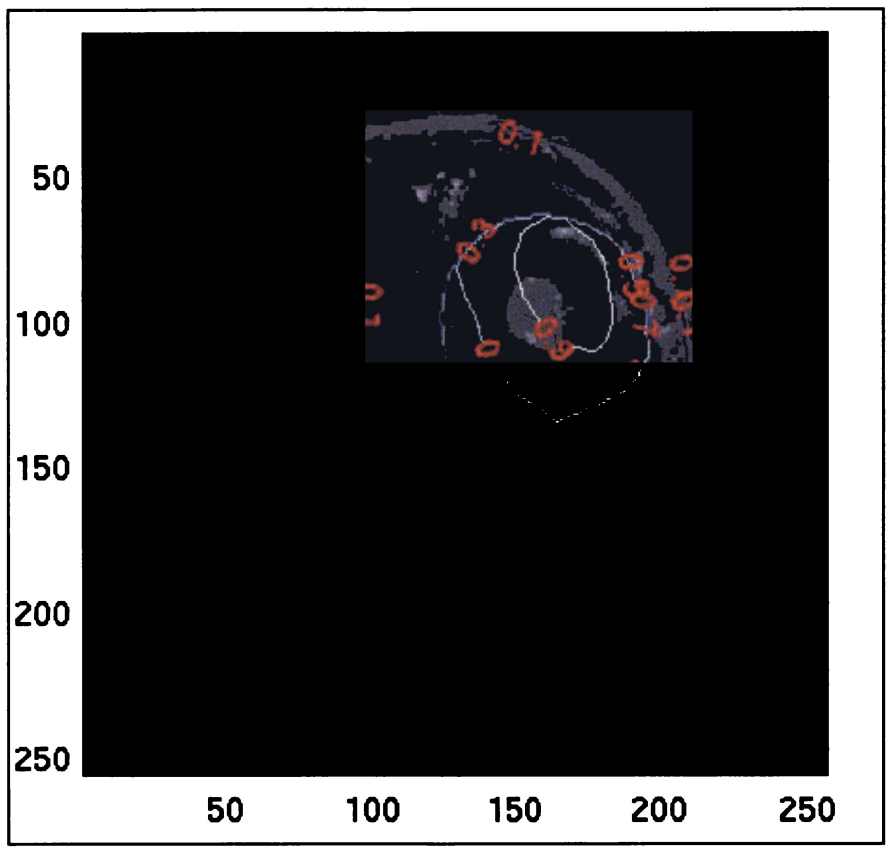


Figure 5A: Homgeneously Scaled Boron Dose (Dose Rate) Contours For Patient EG

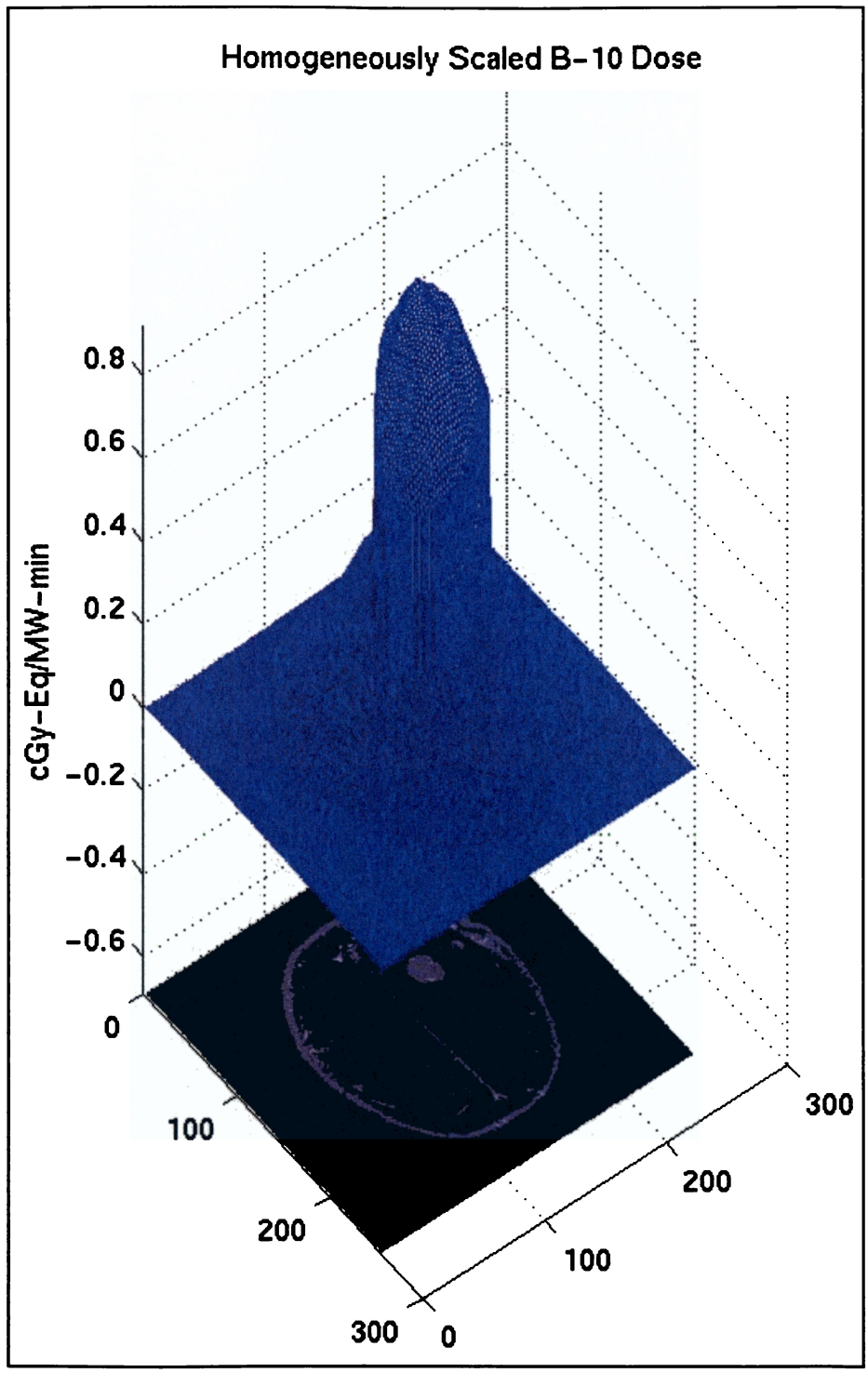


Figure 5B: 3D View of The Heterogeneously Scaled Boron Dose Rate for Patient EG

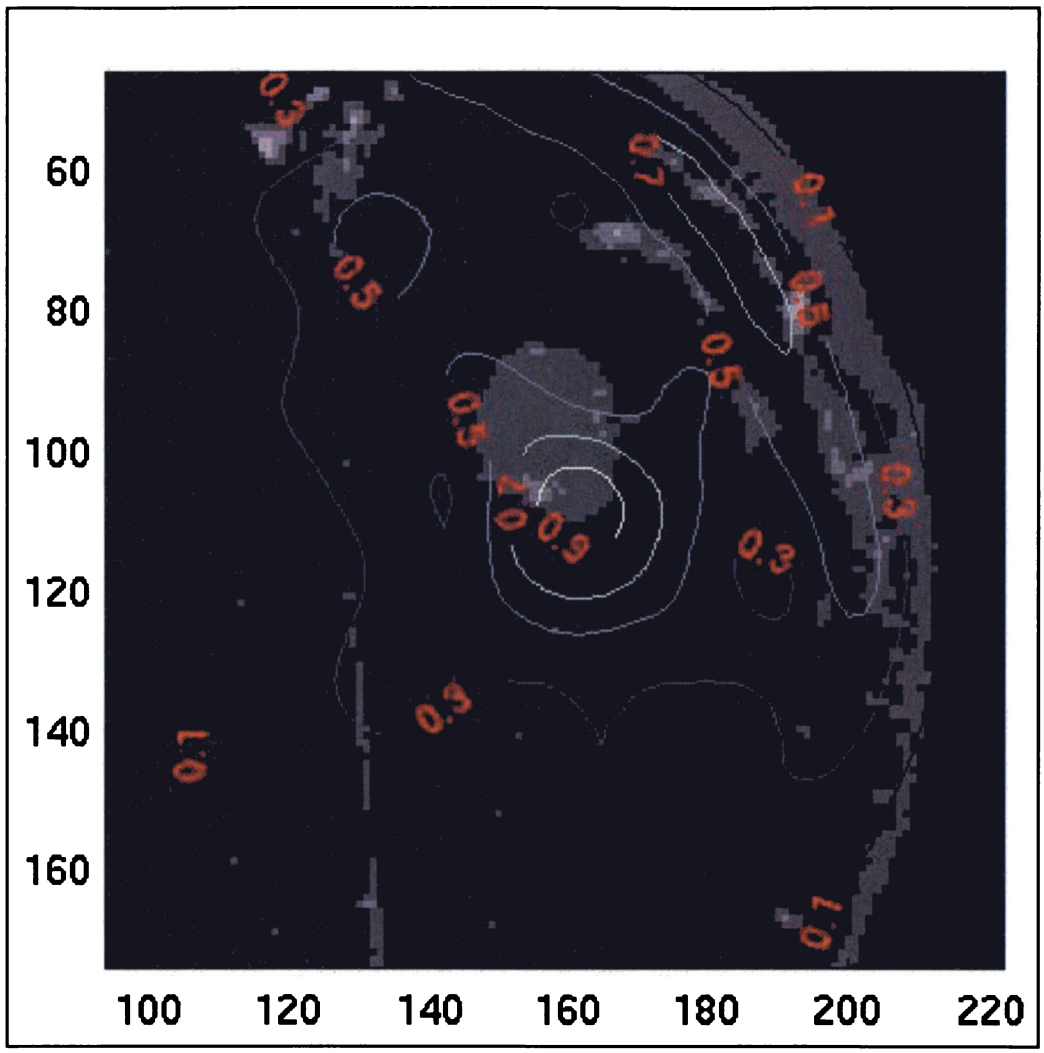


Figure 6A: Registered ^{18}F-BPA PET Based Boron Dose (Dose Rate) Contours

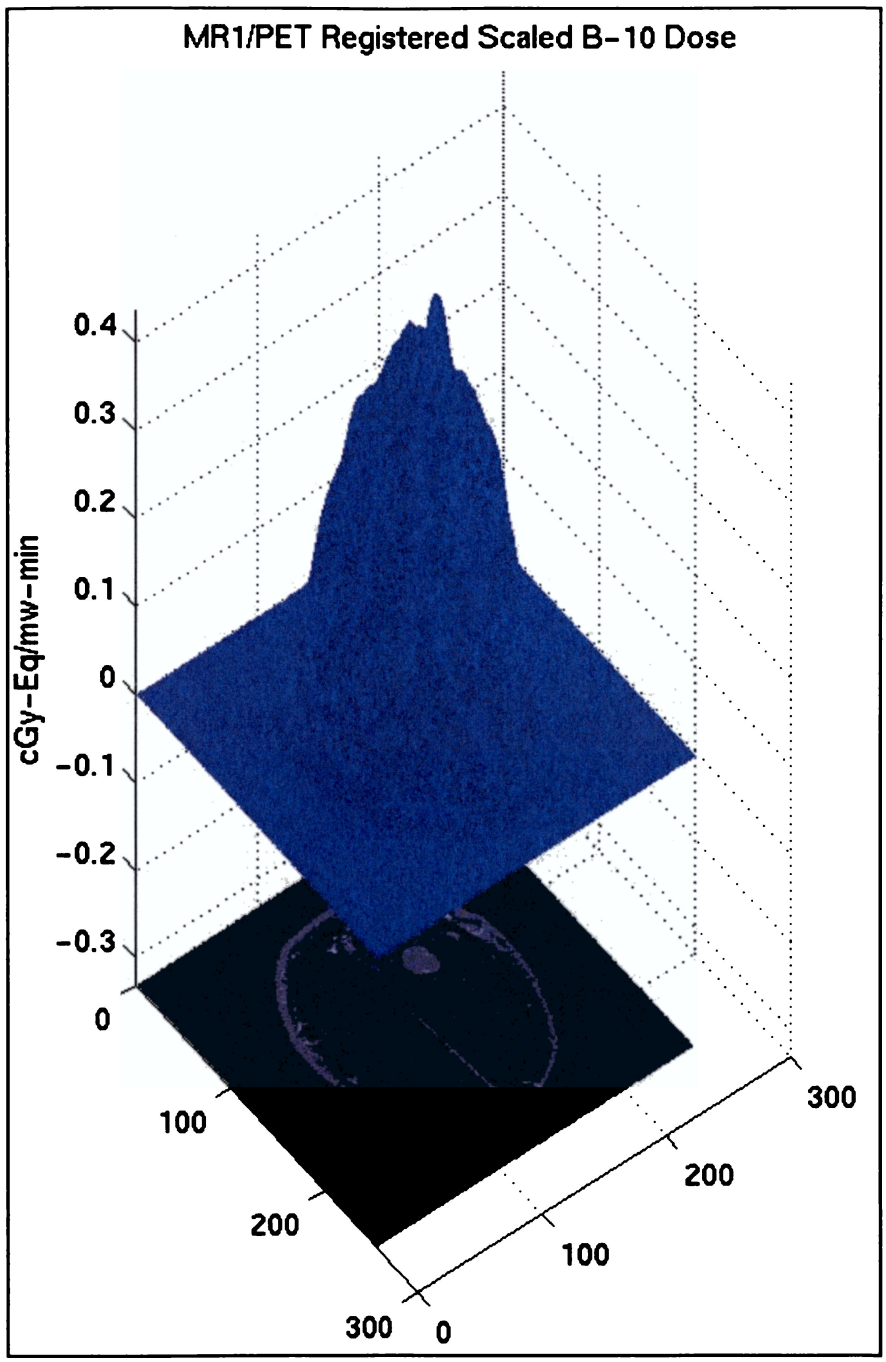


Figure 6B: 3D View of The PET Adjusted Boron Dose Rate for Patient EG

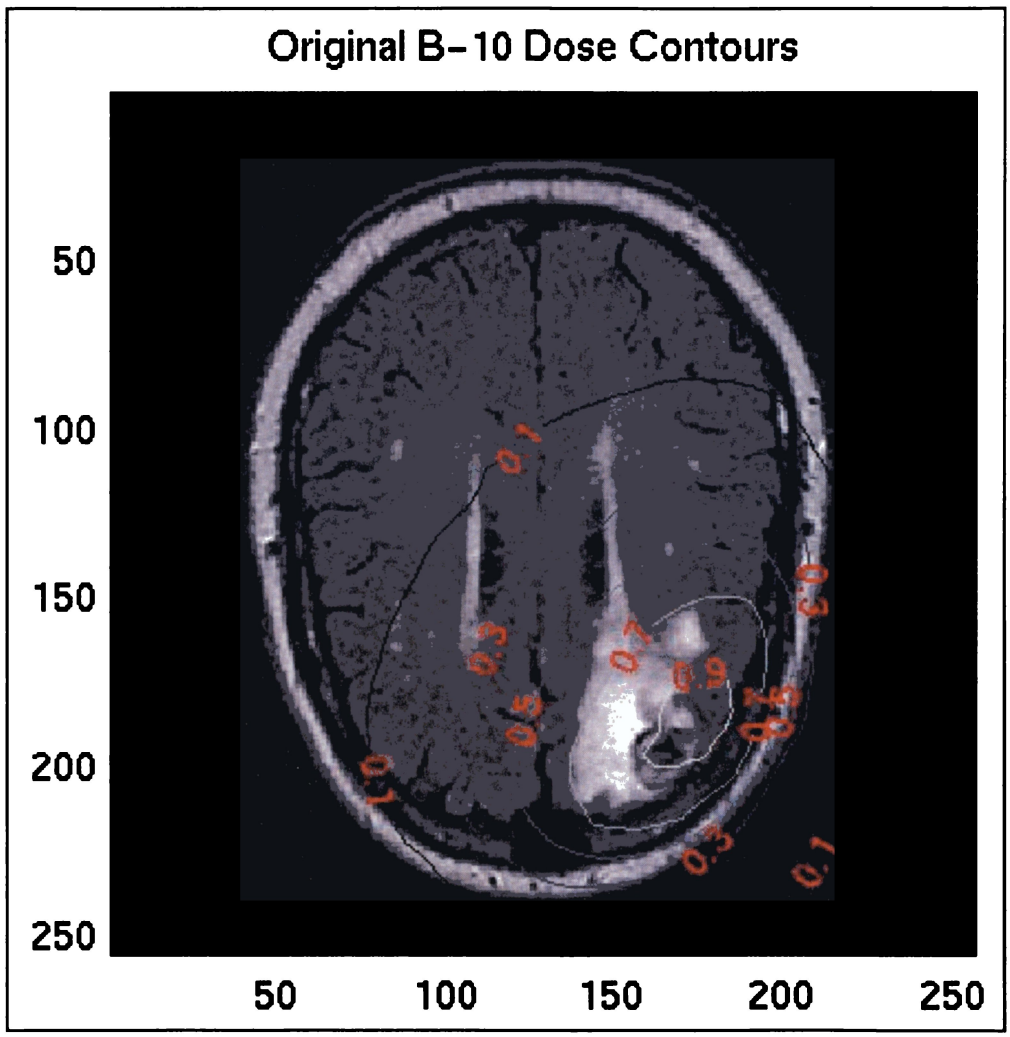


Figure 7A: Original Boron Dose (Dose Rate) Contours As Calculated by SERA for Patient CR

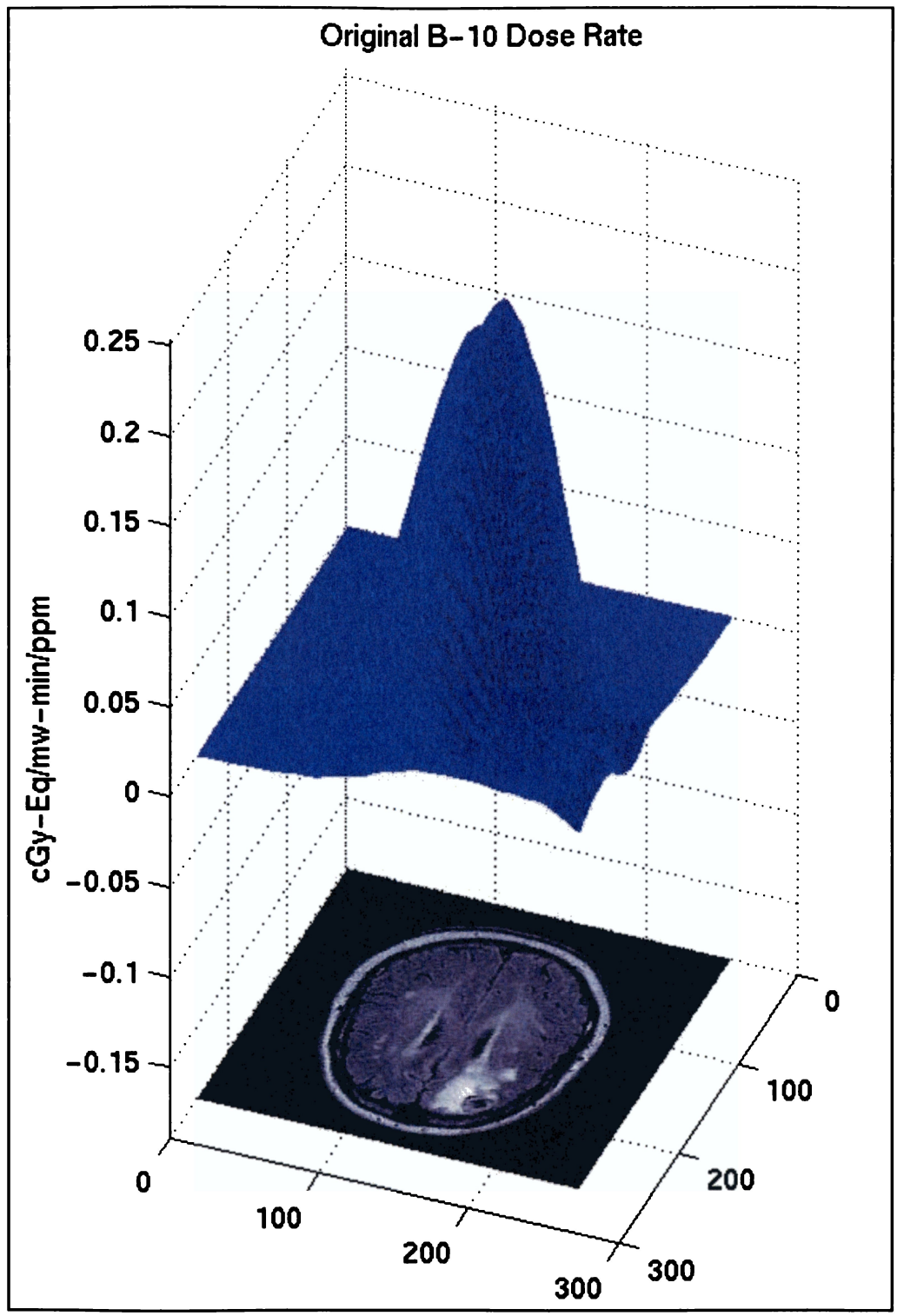


Figure 7B: 3D View of the Original Boron Dose Rate for Patient CR

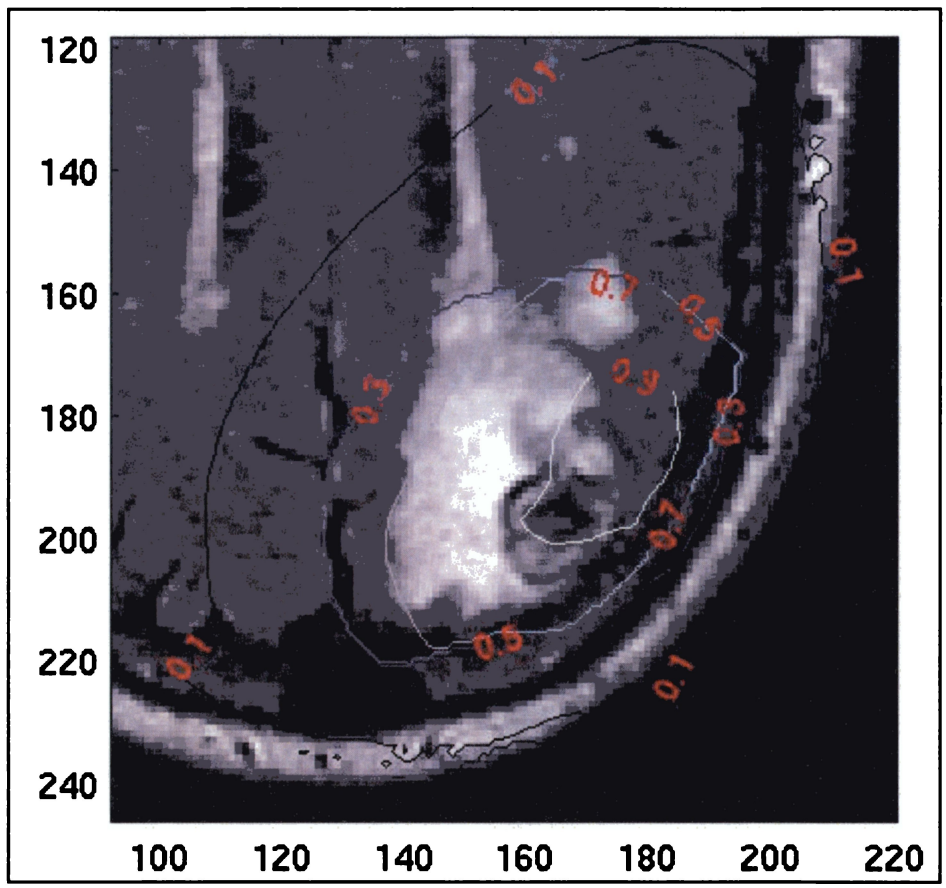


Figure 8A: Homogeneously Scaled Boron Dose (Dose Rate) Contours for Patient CR

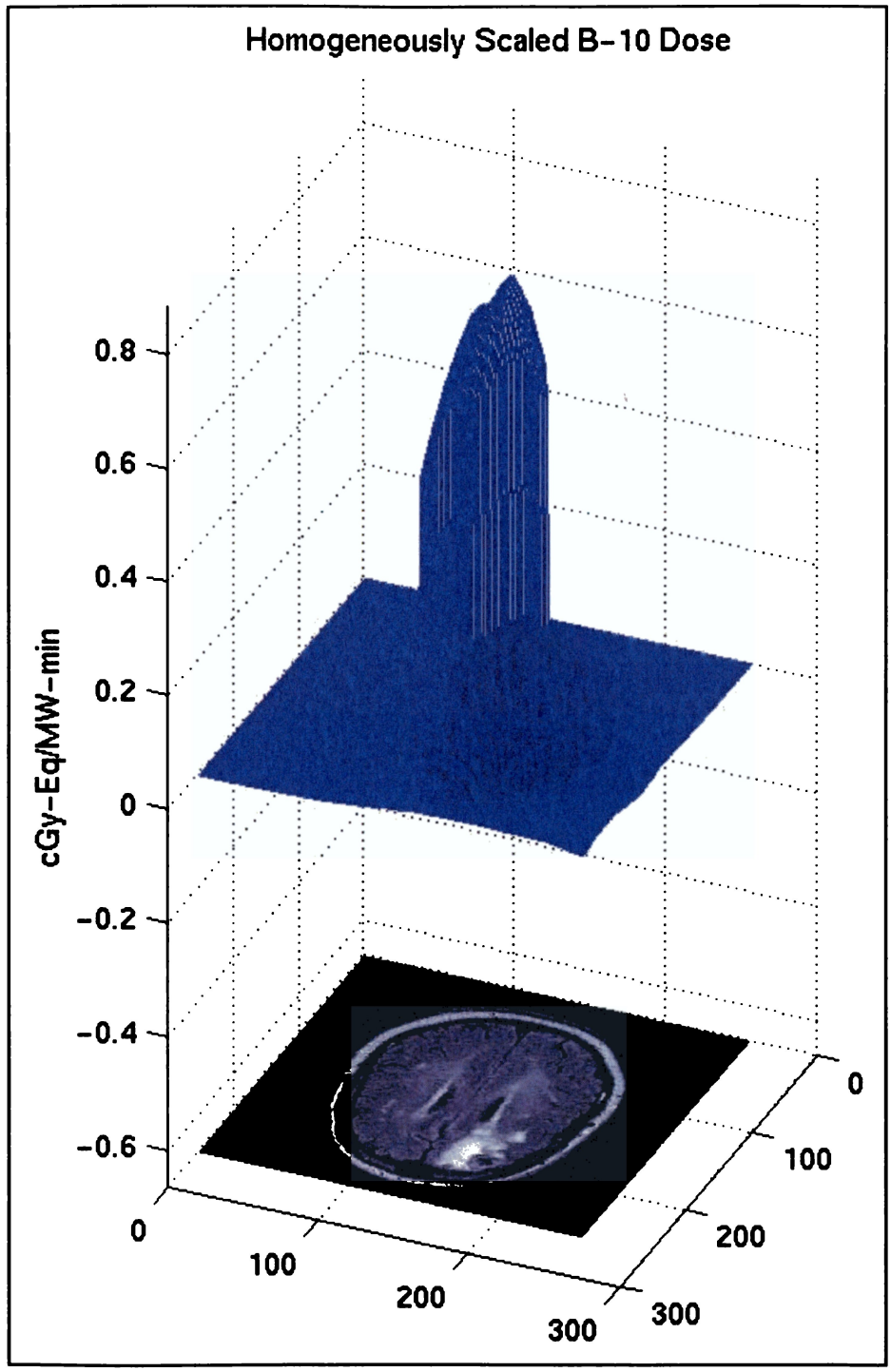


Figure 8B: 3D View of Homogeneously Scaled Boron Dose Rate for Patient CR

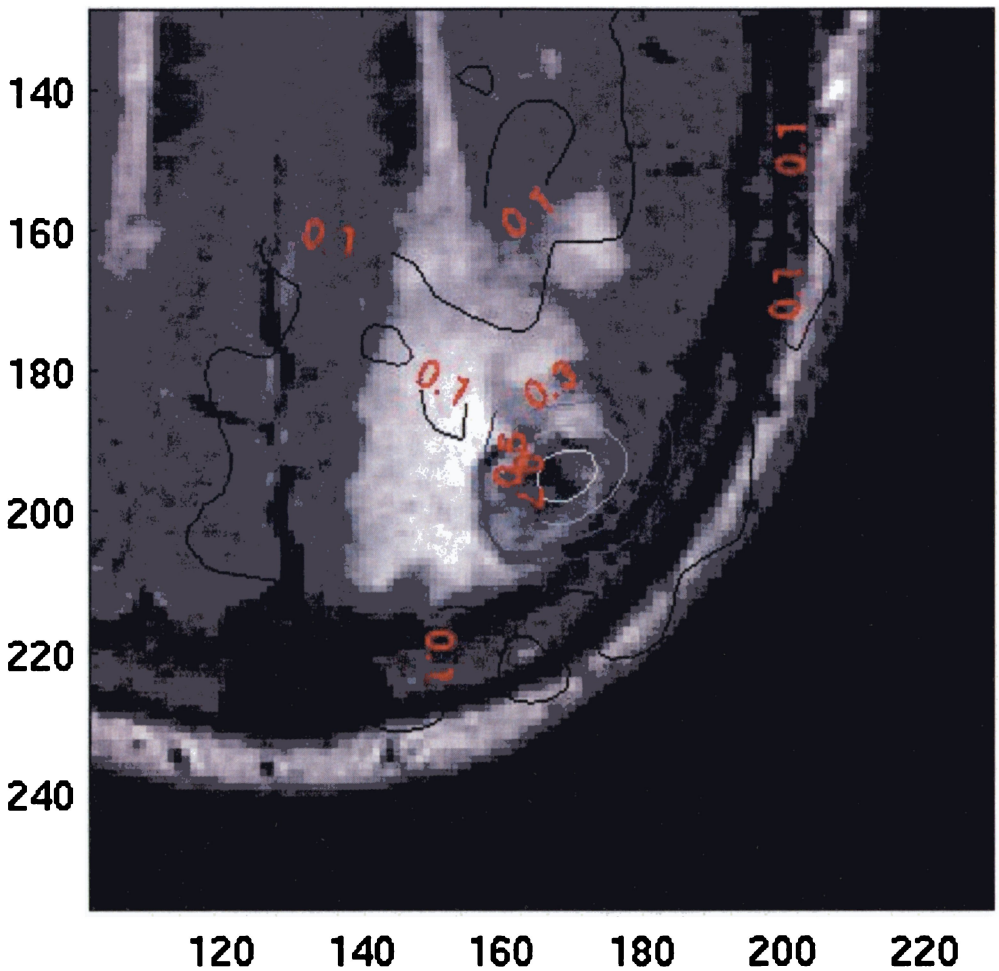


Figure 9A: PET Adjusted Boron Dose (Dose Rate) Contours for Patient CR

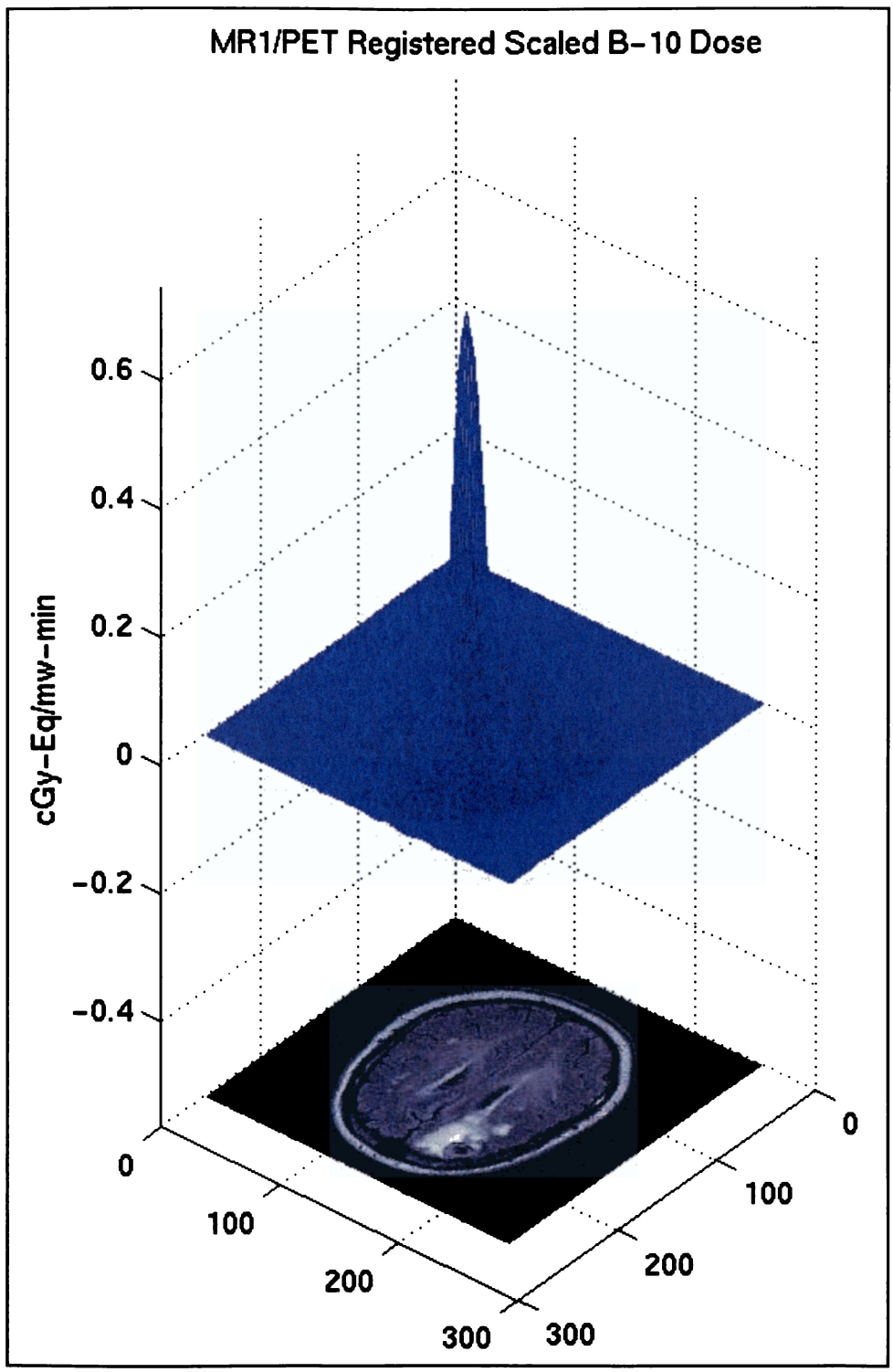


Figure 9B: 3D View of PET Adjusted Boron Dose Rate for Patient CR

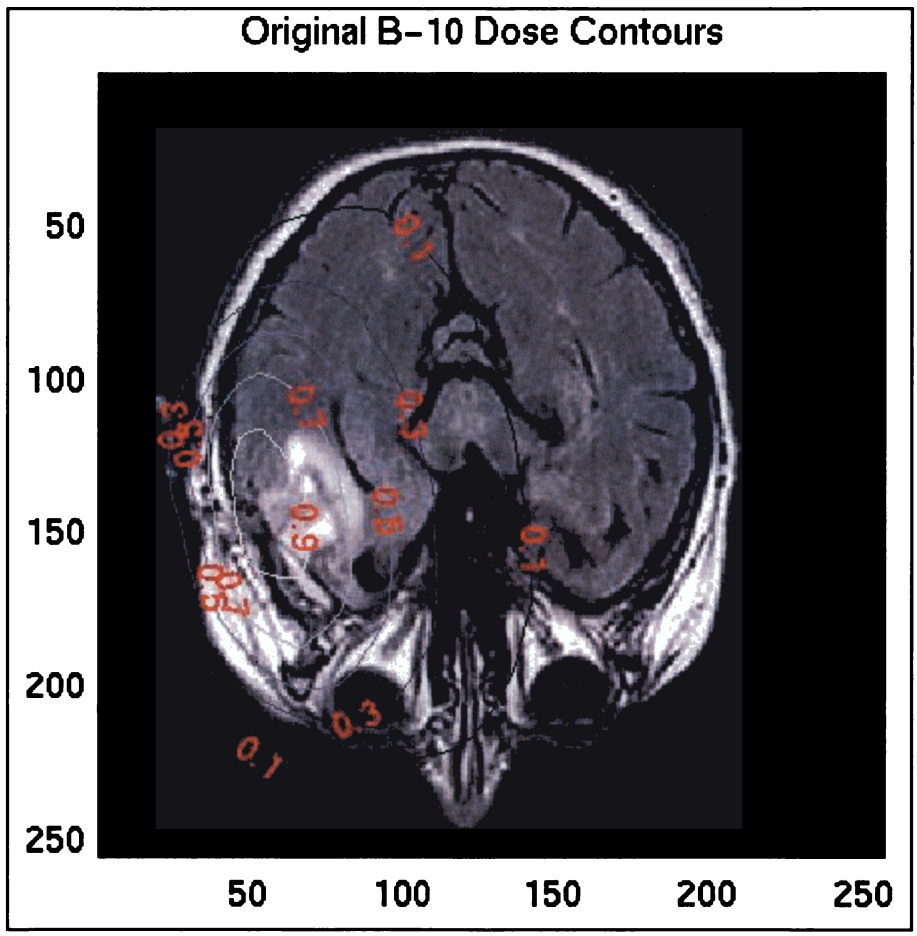


Figure 10 A: Original Boron Dose Contours from SERA for Patient LB

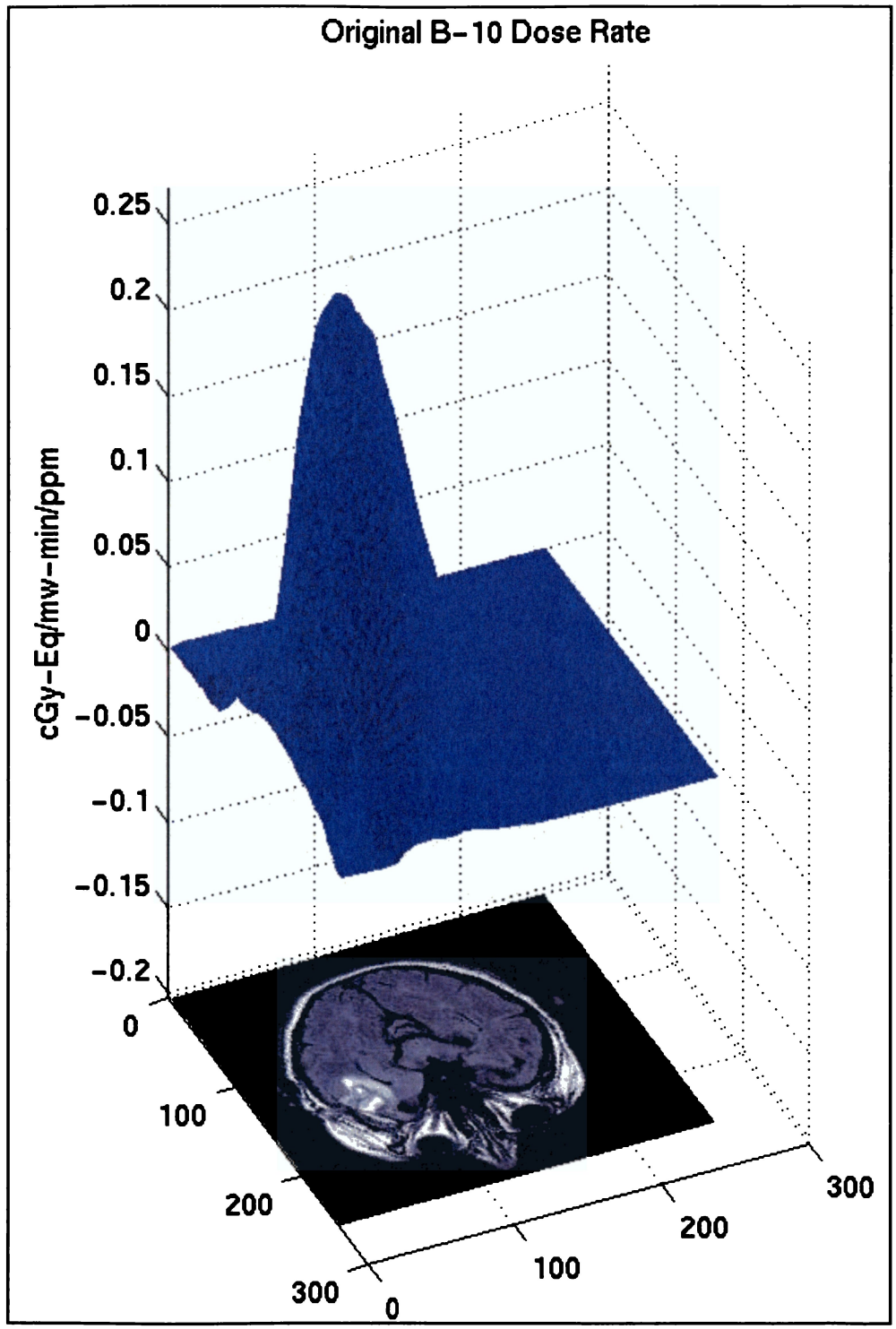


Figure 10B: 3D View of Original Boron Dose Rate from SERA for Patient LB

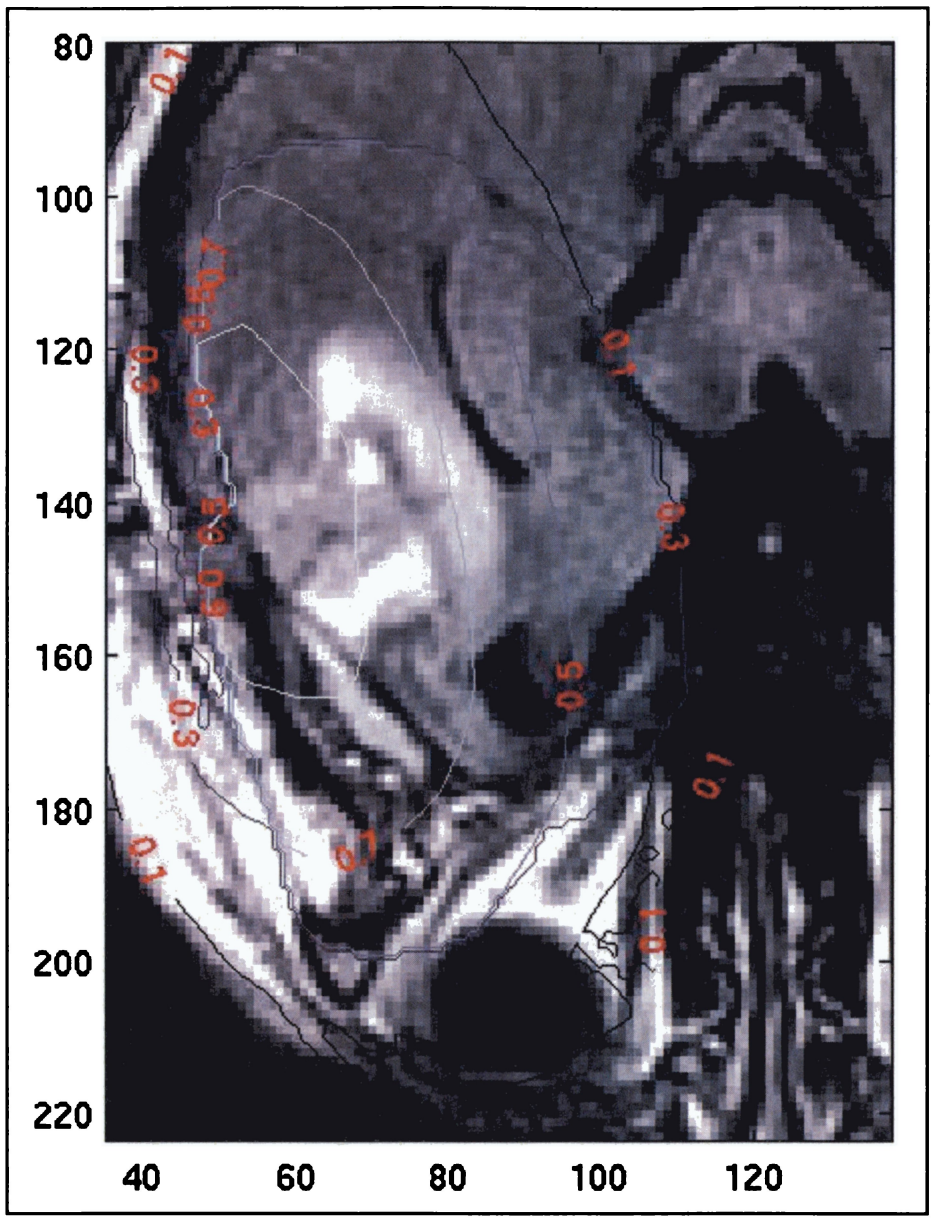


Figure 11A: Homogeneously Scaled Boron Dose (Dose Rate) Contours for Patient LB

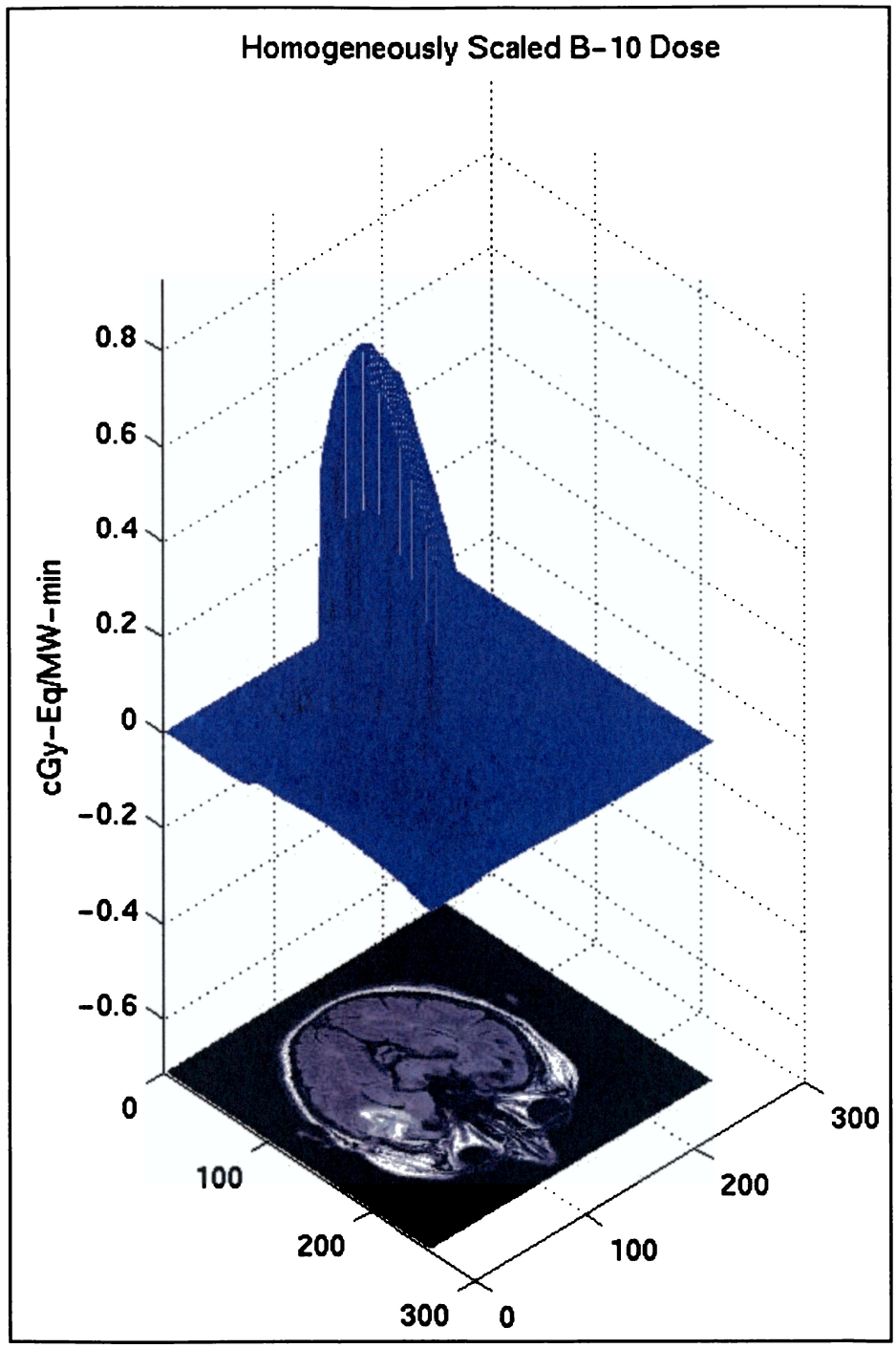


Figure 11B: 3D View of Homogeneousl Scaled Boron Dose Rate for Patient LB

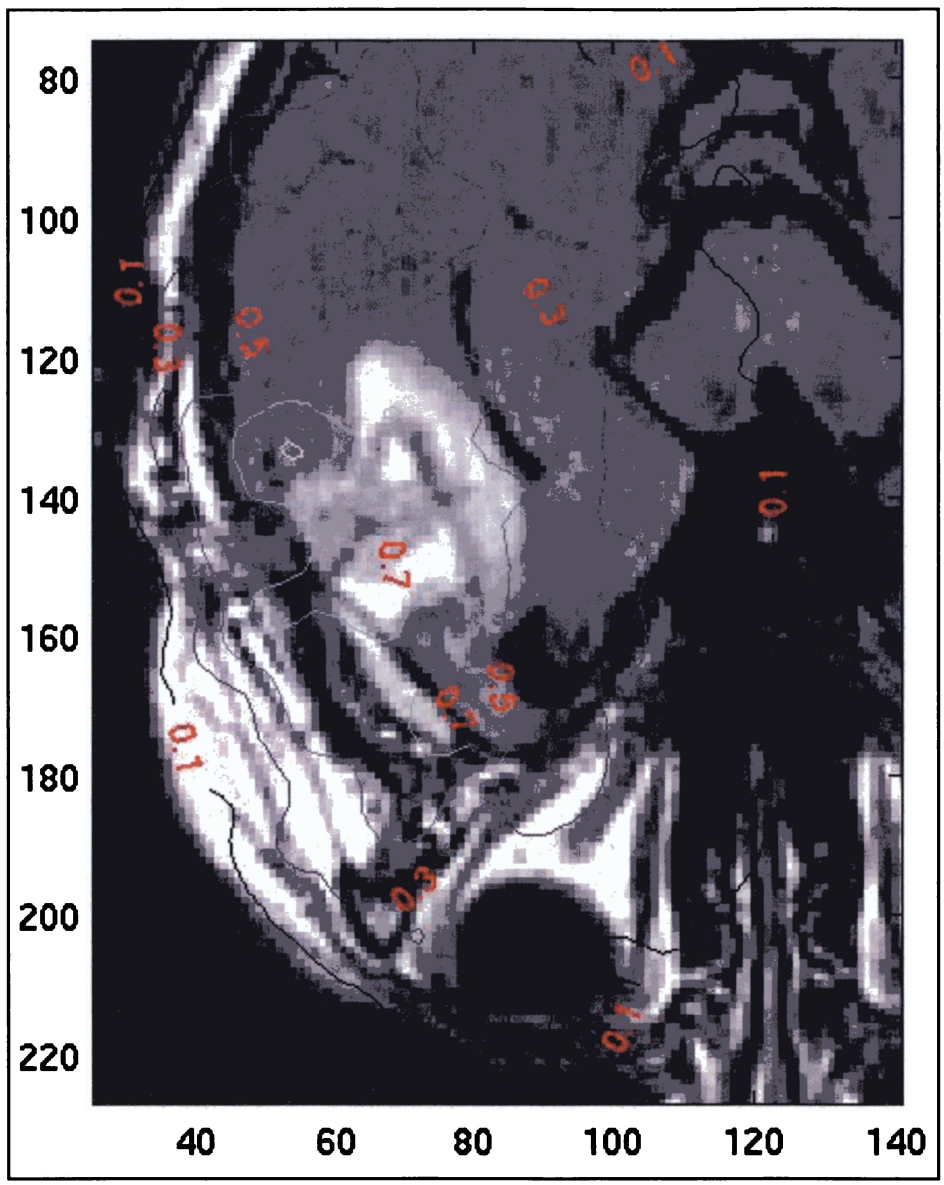


Figure 12A: Registered ^{18}F-BPA PET Adjusted Boron Dose Contours for Patient LB

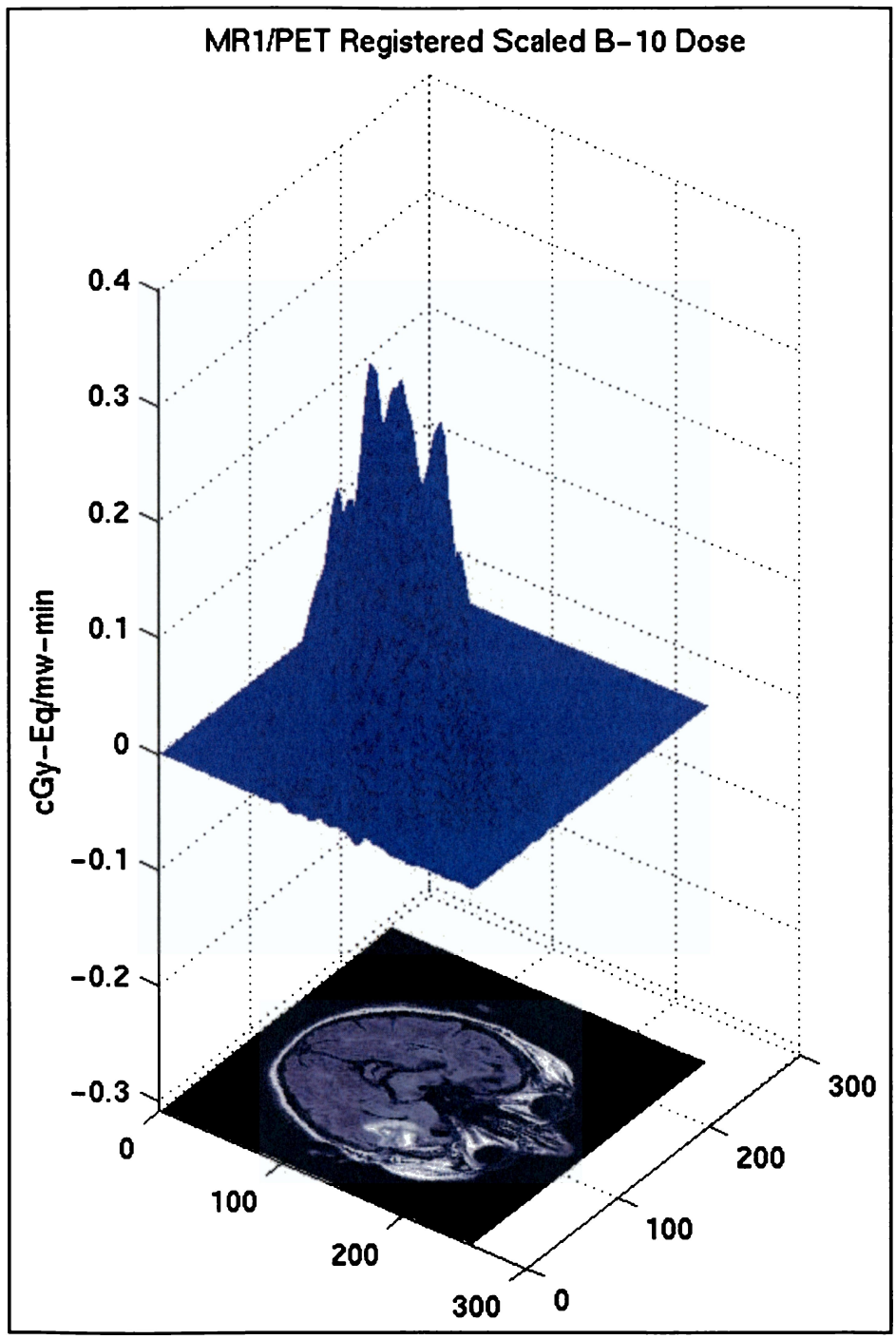


Figure 12B: 3D View of PET Adjusted Boron Dose Contours for Patient LB

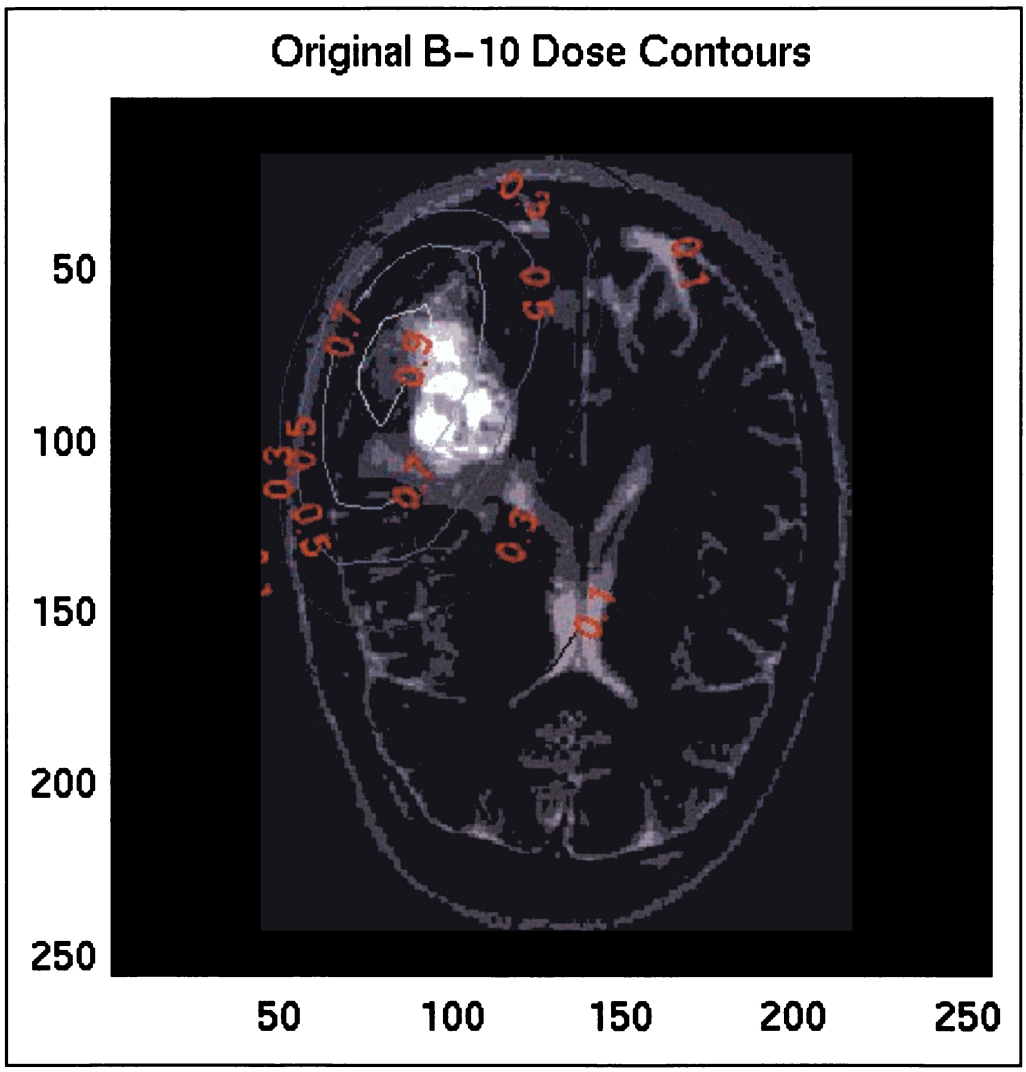


Figure 13 A: Original Boron Dose Contours As Determined from SERA for Patient MB

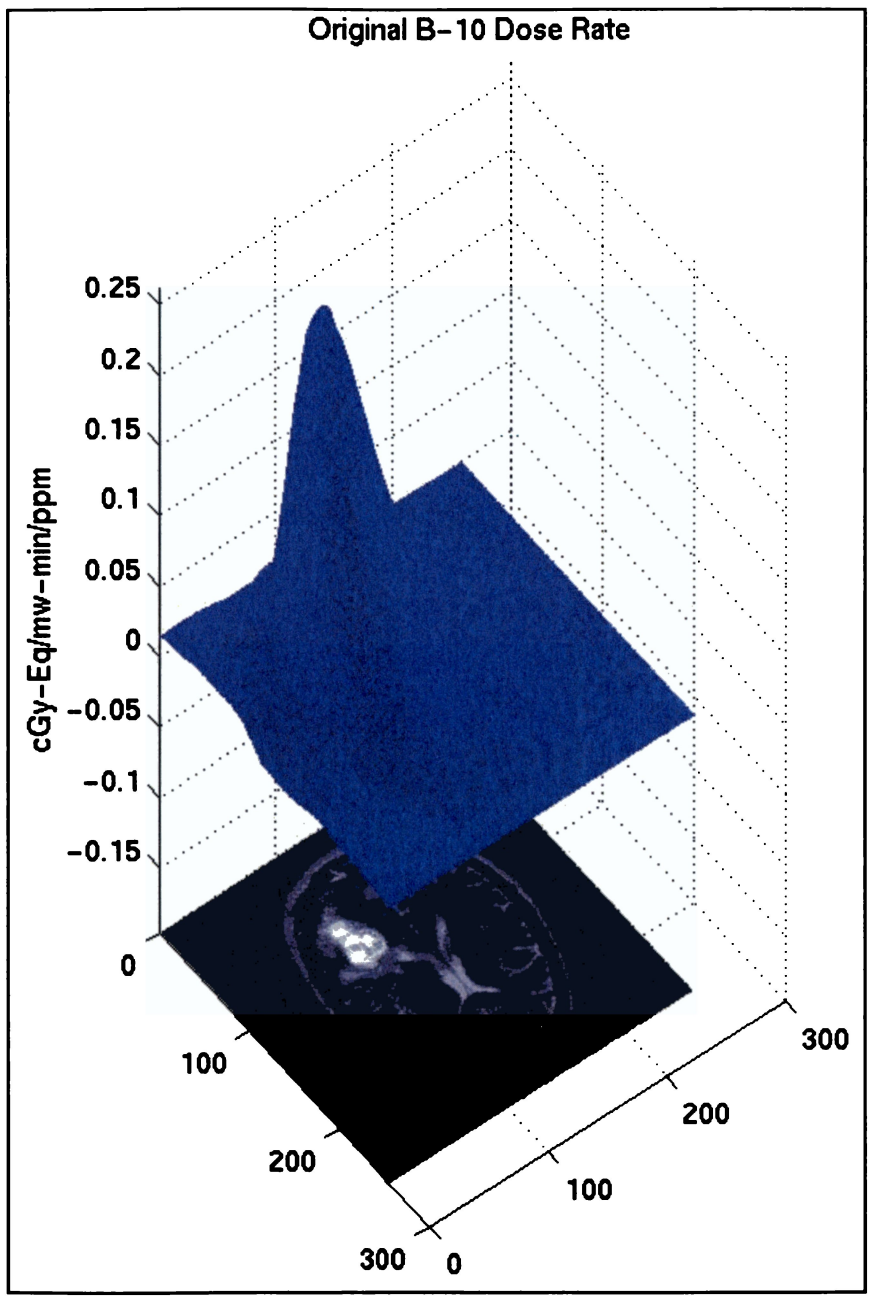


Figure 13B: 3D View of Original Boron Dose Rate from SERA for Patient MB

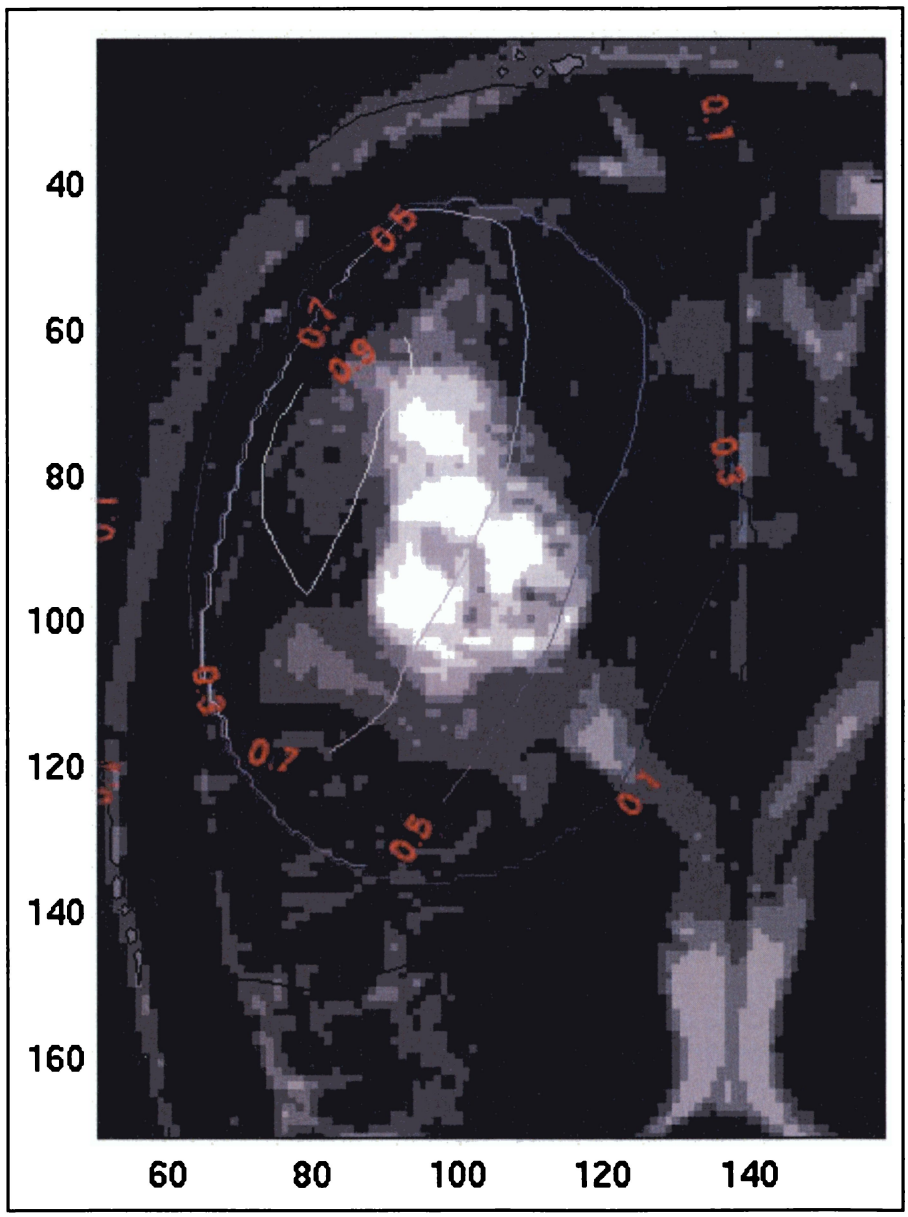


Figure 14A: Homogeneously Scaled Boron Dose Contours for Patient MB

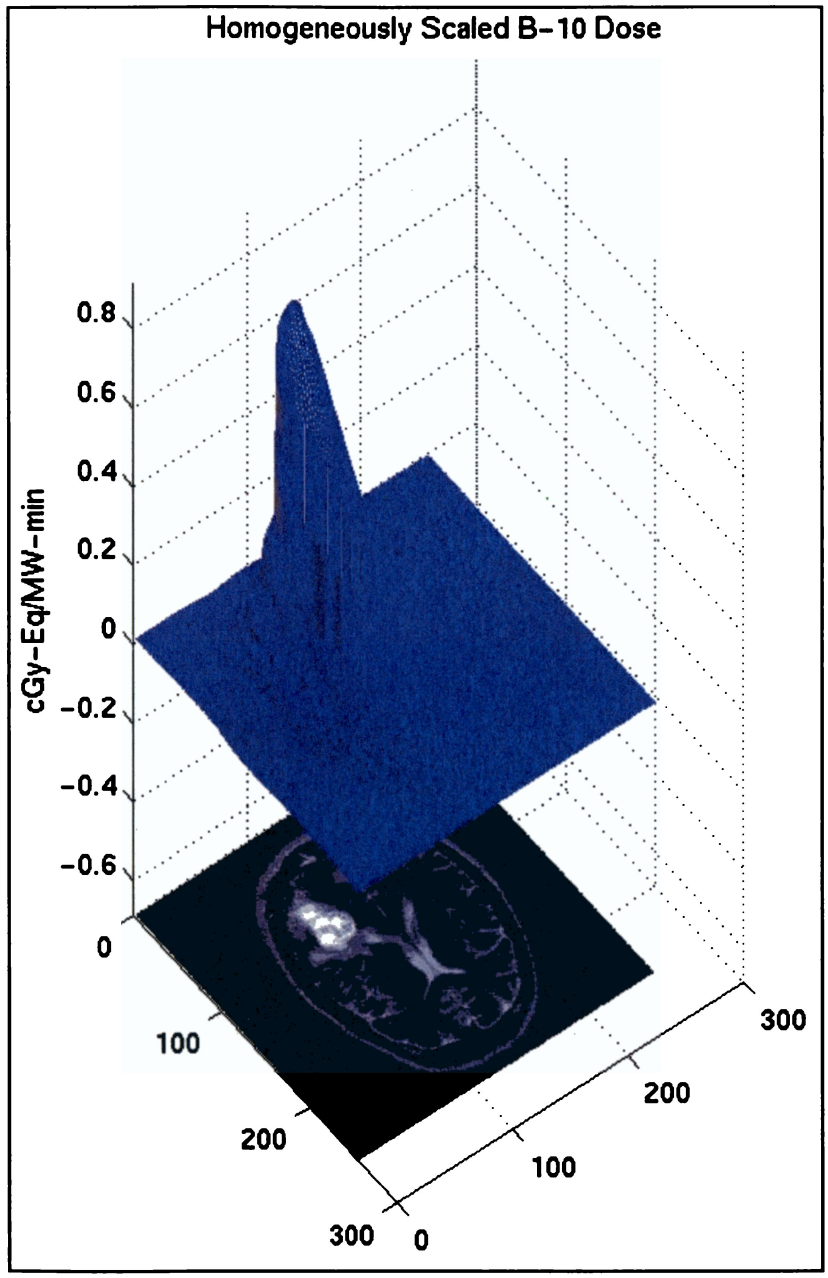


Figure 14B: 3D View of Homogeneously Scaled Boron Dose Rate for Patient MB

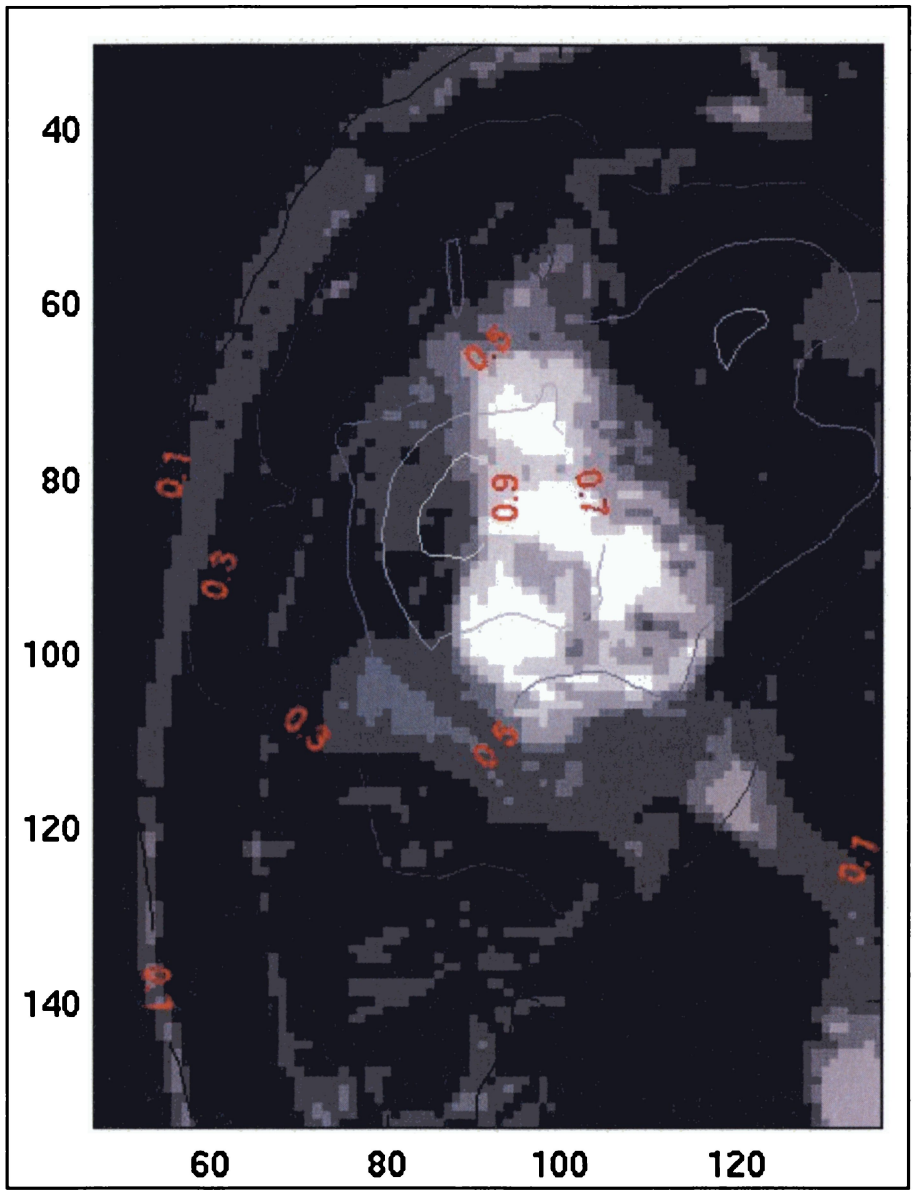


Figure 15A: Registered ^{18}F-BPA PET Scaled Boron Dose Contours for Patient MB

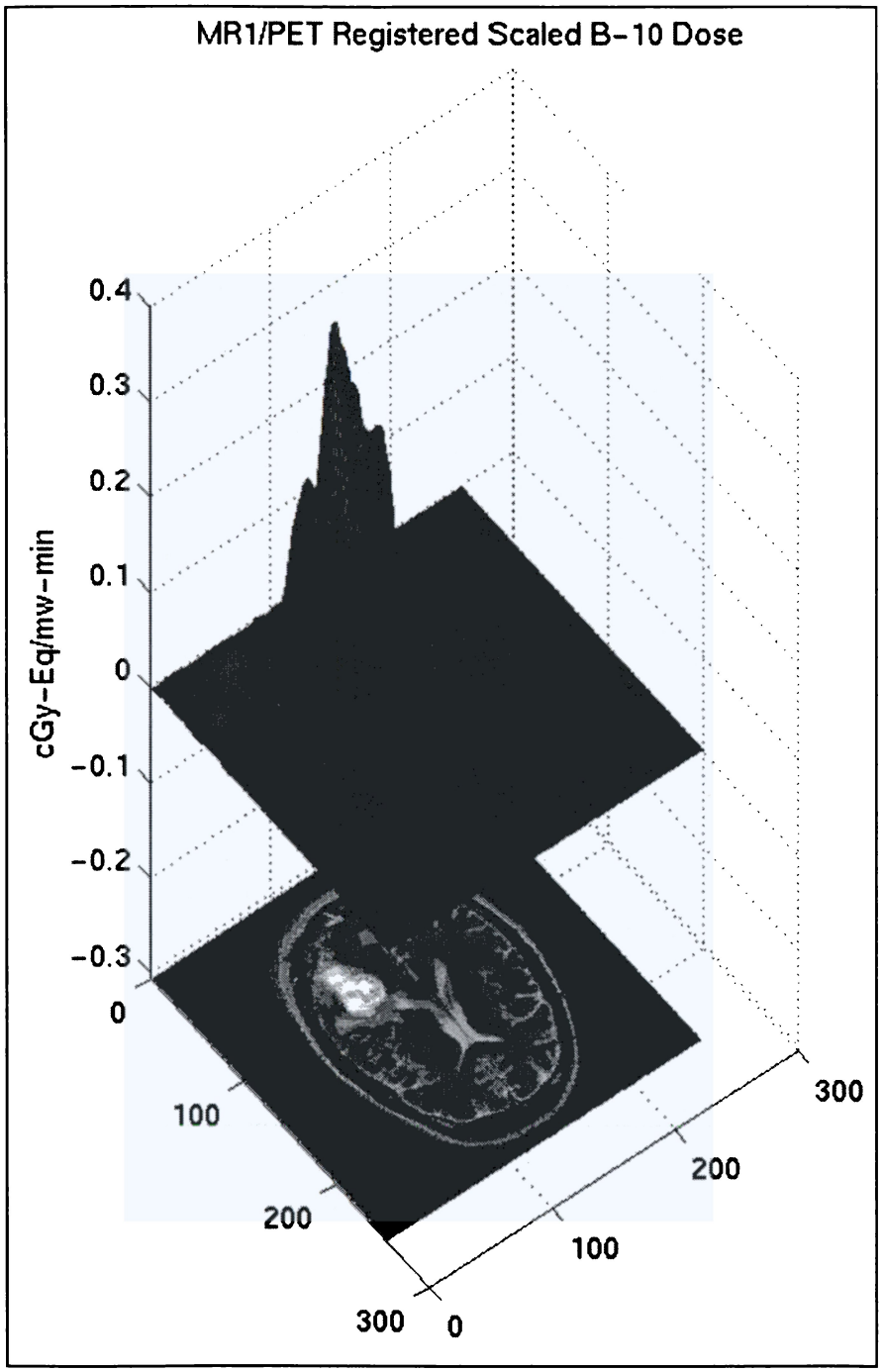


Figure 15B: 3D View of PET Adjusted Boron Dose Rate for Patient MB

Vita

Mohammad Khurram Khan was born in Lahore, Pakistan on May 9, 1975. During his first seven years, he was raised and educated in Sheikhupura, Pakistan. After completing his second grade, he continued his education from 2-12 years in public schools in New York city, in Marshville, N.C., and finished his High School degree from Maryville, T.N in 1993.

He then enrolled at the University of Tennessee, Knoxville (UTK), where he obtained his bachelor of science (B.S.) degree in May, 1997 and his Master's of Science (M.S.) degree in Nuclear Engineering in May, 1999. After completing his B.S. and M.S. degrees, he stayed for another three years to simultaneously work on his second bachelor of science (B.S.) degree in Biology with a Minor in Biochemsitry which he obtained in 2001 and on his Doctorate Degree (Ph.D.) in Nuclear Engineering, which he obtained in May, 2002. During his Master's Degree and his Doctorate Degree, he continued to focus on using Positron Emission Tomography (PET) Images to improve radiation treatment planning for Boron Neutron Capture Therapy (BNCT) patients.

His research experiences, his contacts with the patients, and his experiences with the physicians at the University of Tennessee Medical Center have encouraged him to pursue the path of a medical scientist in the future. Thus, starting in August, 2002, he will continue his education by attending the University of Tennessee, Medical School to obtain his M.D. degree.

3238 2553 57
07-24-02 MRB