

TRUE T_1 , PD IMAGE COMPUTATION FROM A SET OF T_1 WEIGHTED IMAGES

by

Bora Büyüksaraç

B.S. in Electrical and Electronics Engineering,
Boğaziçi University, 1997

Submitted to the Institute of Biomedical Engineering
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Biomedical Science

Bogazici University Library



39001100865693

14

Boğaziçi University
June, 2000

ACKNOWLEDGEMENTS

Firstly I would like to thank Assoc. Prof. Mehmed Özkan, for his generous help and for guiding me in this research. I feel very lucky indeed for working with him because he always behaved very supportive and I believe this thesis is mostly the product of his cooperative attitude. I would also like to thank Assoc. Prof. Ahmet Ademoğlu for sharing his computer knowledge with me during my confused hours.

As a research assistant I also would like to thank to our director Prof. Yorgo I Stefanopulos who always encouraged me at the institute.

As a student I would like to thank so much to my family and to my lovely Güldem for their limitless trust to me at every moment of my research.

As a final word, I would like to thank to all my dearest friends who helped me in finalizing this thesis.

TRUE T_1 , PD IMAGE COMPUTATION FROM A SET OF T_1 WEIGHTED IMAGES

ABSTRACT

Segmentation of tissues in magnetic resonance images is essential especially for a radiologist to be able to identify a disease, tumors, or any tissue. In any magnetic resonance image there exists many different types of tissues each with characteristic T_1 and T_2 decay times and proton densities. If these parameters of tissues can be calculated from the regular magnetic resonance images, the type of tissue could also be determined on any MR image independent of MR hardware characteristics. One such important hardware limitation is the varying sensitivity of an imaging coil spatially. Segmentation algorithms can not distinguish between an intensity variation caused by the imaging coil sensitivity or a variation by tissue change. Calculated T_1 , T_2 , and PD images provide consistent pixel intensity corresponding to the same tissue therefore easier to utilize in conventional segmentation algorithms. To be able to calculate true T_1 and PD parameters, a slice of human head were imaged sixteen times by holding TE fixed and changing TR each time. Levenberg-Marquardt Method is applied to the data and T_1 and PD values were estimated. The true T_1 and true PD images were produced. The maximum likelihood classification is then applied successfully to four MR images of different slices of human head and the robustness of this method in segmenting CSF, WM, and GM is illustrated.

Keywords: T_1 , T_2 , PD, segmentation, Levenberg-Marquardt, maximum likelihood classification

T₁ AĞIRLIKLI MANYETİK REZONANS GÖRÜNTÜLERİ KULLANILARAK GERÇEK T₁ VE GERÇEK PROTON YOĞUNLUĞU GÖRÜNTÜLERİNİN ELDE EDİLMESİ

ÖZET

Manyetik rezonans görüntülerinde doku sınıflandırması özellikle radyologlar için hastalıkların, tümörlerin veya herhangi bir dokunun teşhis edilmesi bakımından çok önem taşımaktadır. Manyetik rezonans görüntülerinde tüm dokuların kendilerine özgü T₁, T₂, ve proton yoğunluğu parametreleri vardır. Bu parametreler tüm manyetik rezonans görüntülemelerinde karakteristik olduğu için, eğer önceden hesaplanırlarsa herhangi bir görüntüde doku sınıflaması kolaylıkla yapılabilir. Görüntüleme bobinindeki herhangi bir duyarlılık değişimi yüzünden değişkenlik gösteren görüntülerde sınıflandırma algoritmaları başarısız olurlar. Gerçek T₁, T₂ ve proton yoğunluğu görüntüleri bu duyarlılık değişimlerinden etkilenmedikleri için sınıflandırma algoritmalarıyla çalışmak daha kolaydır. Bu gerçek parametreleri hesaplamak için insan beyninden seçilen herhangi bir dilim TE'yi sabit tutarak ve TR'yi değiştirerek 16 kez görüntülendi. Ardından Levenberg-Marquardt methodu bu görüntülere uygulandı ve T₁ ve proton yoğunluğu değerleri ortaya çıkarıldı. Böylelikle gerçek T₁ ve gerçek proton yoğunluğu görüntüleri elde edildi. İnsan beyninden seçilen 4 farklı dilime maksimum olasılıkla sınıflandırma methodu başarıyla uygulandı ve bu methodun CSF, WM ve GM sınıflandırılmasındaki güvenilirliği gösterildi.

Anahtar kelimeler: T₁, T₂, proton yoğunluğu, sınıflandırma, Levenberg-Marquardt, maksimum olasılıkla sınıflandırma.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS.....	iii
ABSTRACT.....	iv
ÖZET	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES	vii,viii
LIST OF TABLES.....	ix
LIST OF ABBREVIATIONS AND SYMBOLS	x
1. INTRODUCTION	1
2. MRI	3
2.1 T ₁ Processes	3
2.2 Precession	5
2.3 T ₂ Processes	6
2.4 Rotating Frame of Reference.....	8
2.5 Pulsed Magnetic Fields.....	8
2.6 T ₁ , T ₂ , & ρ Images.....	11
3. TISSUE CLASSIFICATION	13
4. SPIN ECHO.....	17
5. TRUE T ₁ AND TRUE PD IMAGES.....	25
6. SEGMENTATION	29
6.1 Segmentation of Weighted and Calculated Images... ..	29
6.2 Test of Segmentation by Confusion Matrix... ..	36
7. DISCUSSION AND CONCLUSION	37
7.1 Future Directions... ..	37
APPENDIX.....	38
REFERENCES	76

LIST OF FIGURES

		Page
FIGURE 2.1	Applied magnetic field B_0 and the equilibrium magnetization M_0	1
FIGURE 2.2	Net magnetization is zero, $M_Z=0$	2
FIGURE 2.3	Longitudinal magnetization, M_Z , returns to its equilibrium value, M_0 .	2
FIGURE 2.4	Longitudinal magnetization, M_Z , returns to its equilibrium value, M_0 starting at $-M_Z$	3
FIGURE 2.5	Transverse magnetization, M_{XY} , returns to its equilibrium value.	4
FIGURE 2.6	Longitudinal magnetization grows until we have M_0 along Z.	5
FIGURE 2.7	A coil of wire that produces magnetic field around X axis	6
FIGURE 2.8	The magnetic field is constant in a rotating frame of reference.	7
FIGURE 2.9	A 90° pulse rotates the equilibrium magnetization down to the Y' axis.	8
FIGURE 2.10	A 180° pulse rotates the equilibrium magnetization down to along the -Z axis.	8
FIGURE 2.11	The signal versus TE and the best fit line.	9
FIGURE 2.12	The signal versus TR and the best fit line.	10
FIGURE 3.1	A spin echo image that shows intensity of pixels of tissues.	11
FIGURE 3.3	The histogram and look up table for the image in Figure 3.1.	12
FIGURE 3.3	Lookup table to segment data values greater than 865 from others	12
FIGURE 3.4	Segmented image by using the lookup table in Figure 3.3.	13
FIGURE 3.5	Three-dimensional histogram of tissues in the brain	14
FIGURE 3.6	Segmented image of the brain	14
FIGURE 4.1	Spin echo pulse sequence	15
FIGURE 4.2	T_2 decay curves for CSF, GM, and WM are determined by the amplitude of M_{XY} as a function of TE.	16
FIGURE 4.3	A spin echo brain image showing the signal intensities of different tissues	17
FIGURE 4.4	Regrowth of M_Z and decay of M_{XY} for an extremely long TR.	17
FIGURE 4.5	Use of an intermediate TR does not allow full regrowth of M_Z .	18
FIGURE 4.6	With extremely short TR, little M_Z is reformed, resulting in more T_1 -weighted images at all clinically employed echo times.	19

FIGURE 4.7	Spin echo images produced at various TR times with a long TE result in T_2 -weighted images	20
FIGURE 4.8	Spin echo images produced at various TR times with short TE result in varying PD - and T_1 -weighted images.	21
FIGURE 5.1	Spin echo images produced at successive TR times with short TE.	23
FIGURE 5.2	Pixel value versus TR.	23
FIGURE 5.3	A sample dataset of estimated parameters and the best fit.	24
FIGURE 5.4	Calculated and weighted T_1 images.	25
FIGURE 5.5	Calculated T_1 and calculated PD images.	25
FIGURE 5.6	Window-level mechanism that is used to display the images in gray scale.	26
FIGURE 6.1	Scatter diagram of pixels chosen from T_1 -weighted and PD-weighted images.	27
FIGURE 6.2	Scatter diagram of pixels chosen from true T_1 and true PD images.	28
FIGURE 6.3	The multivariate Gaussian density function of the weighted dataset	29
FIGURE 6.4	The multivariate Gaussian density function of the calculated dataset	29
FIGURE 6.5	Segmented slice1 by using weighted dataset	30
FIGURE 6.6	Segmented slice1 by using calculated dataset	30
FIGURE 6.7	Segmented slice2 by using weighted dataset	31
FIGURE 6.8	Segmented slice2 by using calculated dataset	31
FIGURE 6.9	Segmented slice3 by using weighted dataset	32
FIGURE 6.10	Segmented slice3 by using calculated dataset	32
FIGURE 6.11	Segmented slice4 by using weighted dataset	33
FIGURE 6.12	Segmented slice4 by using calculated dataset	33

LIST OF TABLES

		Page
TABLE 4.1	Relationship of T_1 , T_2 , and PD weighted images with TR and TE	19
TABLE 6.1	Segmentation results of 400 test pixels (by using weighted dataset)	34
TABLE 6.2	Segmentation results of 400 test pixels (by using calculated dataset)	34

LIST OF ABBREVIATIONS AND SYMBOLS

T_1	Spin lattice relaxation time
T_2	Spin-spin relaxation time
PD	Proton density
TR	Repetition time
TE	Echo time
A_E	Amplitude of echo
B_0	Applied magnetic field
M_Z	Longitudinal magnetization
M_0	Equilibrium magnetization
M_{XY}	Transverse magnetization
CSF	Cerebrospinal fluid
WM	White matter
GM	Gray matter

1. INTRODUCTION

Magnetic resonance imaging (MRI) provides detailed images of tissues and is used for both human brain and body studies. Data obtained from MR images is used for detecting tissue deformities such as cancers and injuries; MR is also used extensively in studies of brain pathology, where regions of interest (ROI's) are often examined in detail, for example in multiple sclerosis (MS) studies [1], [2]. In order to perform good quantitative studies, ROI's within the brain must be well defined. In traditional methods, a skilled operator manually outlines the ROI's using a mouse or cursor. More recently, computer-assisted methods have been used for specific tasks such as extraction of MS lesions from MRI brain scans [3], [4], or extraction of the cerebral ventricles in schizophrenia studies [5]. Many of these computer-assisted tasks require segmentation of the whole brain from the head, either because the whole brain is the ROI such as in Alzheimer's studies [6] or because automatic ROI extraction using statistical methods is made easier if the skull and scalp have been removed [3].

Many techniques have been utilized over the years for the segmentation of biological images into tissue components. Classical methods range from simple thresholding to more sophisticated techniques including methods based on local features such as the median, the variance, or the gradient [7]. These techniques, however, do not take advantage of the multidimensional nature of the data. MRI is inherently multidimensional as it provides information about three tissue-dependent parameters: T1 (the spin-lattice relaxation time), T2 (the spin-spin relaxation time), and the proton density (PD).

Multidimensional data classification has been used extensively in the area of remote sensing and the parallel between multi-spectral images of the earth and MRI was made as early as 1985 [8]. Since that time a number of studies have explored the potential of various supervised and unsupervised pattern recognition techniques for the segmentation of MRI data [9],[10],[11]-[21]. Generally, the results are found to be in good visual agreement with expert opinion, but a number of factors affecting the classifiers are reported:

1. intensity variations introduced by the radio frequency (RF) coil,
2. imager-imposed inter-patient variations,
3. inter-slice intensity variations, and
4. the difficulty in assessing the accuracy of the methods.

The latter is due to the absence of gold standards to which the results of the automatic segmentation can be compared and to the large inter-rater variability observed when manual delineation is performed by human experts.

To overcome the intensity variation problem in segmentation, true T1 and true PD images are calculated in this work by using a series of images taken with different TR values. In chapter 2, the relaxation processes are mentioned. In chapter 3, different tissue classification techniques are discussed. In chapter 4, the spin echo pulse sequence and the magnetization curves of cerebro spinal fluid, white matter and gray matter are illustrated. In chapter 5 calculation of T1 and PD images are explained. Finally, in chapter 6 the segmentation algorithm is examined and the segmented brain slices are evaluated using confusion matrices.

2. MRI

Magnetic resonance imaging (MRI) is an imaging technique used primarily in medical settings to produce high quality images of the inside of the human body. MRI is based on the principles of nuclear magnetic resonance (NMR), a spectroscopic technique used by scientists to obtain microscopic chemical and physical information about molecules. The technique was called magnetic resonance imaging rather than nuclear magnetic resonance imaging (NMRI) because of the negative connotations associated with the word “nuclear” in the late 1970's. MRI started out as a tomographic imaging technique, that is, it produced an image of the NMR signal in a thin slice through the human body. MRI has advanced beyond a tomographic imaging technique to a volume imaging technique.

2.1 T_1 Processes

At equilibrium, the net magnetization vector lies along the direction of the applied magnetic field B_0 and is called the equilibrium magnetization M_0 (Figure 2.1). In this configuration, the Z component of magnetization M_Z equals M_0 . M_Z is referred to as the longitudinal magnetization. There is no transverse (M_X or M_Y) magnetization here.

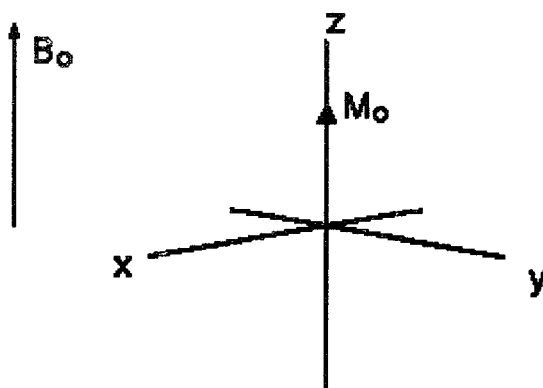


Figure 2.1 Applied magnetic field B_0 and the equilibrium magnetization M_0 .

It is possible to change the net magnetization by exposing the nuclear spin system to energy of a frequency equal to the energy difference between the spin states. If enough energy is put into the system, it is possible to saturate the spin system and make $M_Z = 0$ (Figure 2.2).

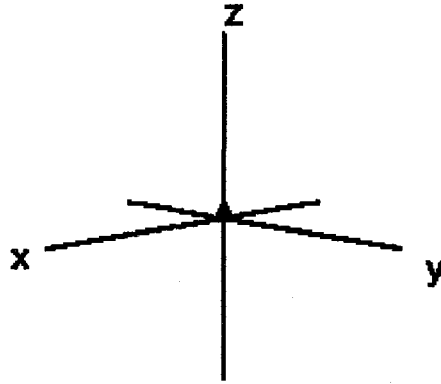


Figure 2.2 Net magnetization is zero, $M_Z = 0$.

The time constant that describes how M_Z returns to its equilibrium value is called the spin lattice relaxation time (T_1) (Figure 2.3). The equation governing this behavior as a function of the time t after its displacement is:

$$M_Z = M_0 (1 - e^{-t/T_1}) \quad (2.1)$$

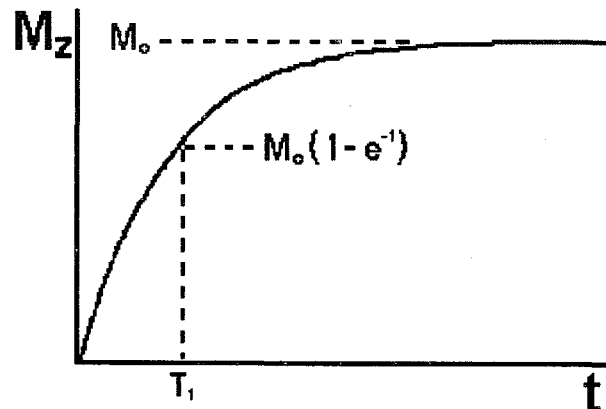


Figure 2.3 Longitudinal magnetization, M_Z , returns to its equilibrium value, M_0 .

T_1 is therefore defined as the time required to change the Z component of magnetization by a factor of 'e'.

If the net magnetization is placed along the -Z axis, it will gradually return to its equilibrium position along the +Z axis at a rate governed by T_1 (Figure 2.4). The equation governing this behavior as a function of time t after its displacement is:

$$M_Z = M_0 (1 - 2e^{-t/T_1}) \quad (2.2)$$

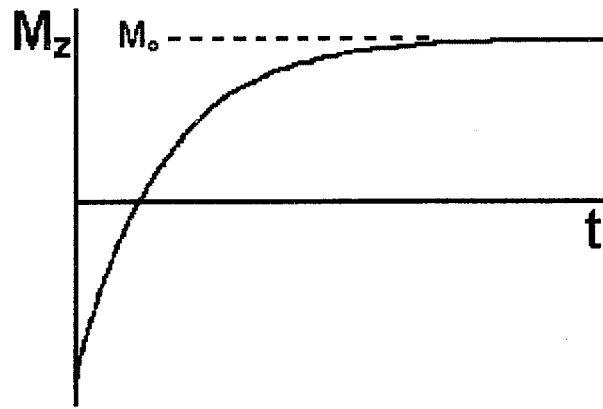


Figure 2.4 Longitudinal magnetization, M_Z , returns to its equilibrium value, M_0 starting at $-M_Z$.

The spin-lattice relaxation time (T_1) is the time to reduce the difference between the longitudinal magnetization (M_Z) and its equilibrium value by a factor of 'e'.

2.2 Precession

If the net magnetization is placed in the XY plane, it will rotate about the Z axis at a frequency equal to the frequency of the photon which would cause a transition between the two energy levels of the spin. This frequency is called the Larmor frequency.

2.3 T_2 Processes

In addition to the rotation, the net magnetization starts to dephase because each of the spin packets making it up is experiencing a slightly different magnetic field and rotates at its own Larmor frequency. The longer the elapsed time, the greater the phase difference is. Here the net magnetization vector is initially along +Y. For this and all dephasing examples think of this vector as the overlap of several thinner vectors from the individual spin packets.

The time constant that describes the return to equilibrium of the transverse magnetization, M_{XY} , is called the spin-spin relaxation time, T_2 (Figure 2.5).

$$M_{XY} = M_{XY0} e^{-t/T_2} \quad (2.3)$$

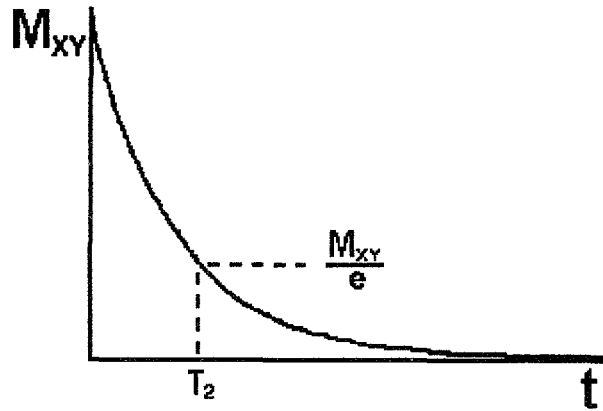


Figure 2.5 Transverse magnetization, M_{XY} , returns to its equilibrium value.

T_2 is always less than or equal to T_1 . The net magnetization in the XY plane goes to zero and then the longitudinal magnetization grows until we have M_0 along Z (Figure 2.6). Any transverse magnetization behaves the same way.

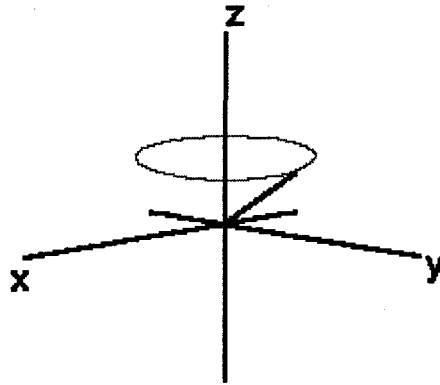


Figure 2.6 Longitudinal magnetization grows until we have M_0 along Z.

The transverse component rotates about the direction of applied magnetization and dephases. T_1 governs the rate of recovery of the longitudinal magnetization.

In summary, the spin-spin relaxation time, T_2 , is the time to reduce the transverse magnetization by a factor of 'e'. In the previous sequence, T_2 and T_1 processes are shown separately for clarity. That is, the magnetization vectors are shown filling the XY plane completely before growing back up along the Z axis. Actually, both processes occur simultaneously with the only restriction being that T_2 is less than or equal to T_1 . Two factors contribute to the decay of transverse magnetization.

- 1) molecular interactions (said to lead to a *pure* T_2 molecular effect)
- 2) variations in B_0 (said to lead to an *inhomogeneous* T_2 effect)

The combination of these two factors is what actually results in the decay of transverse magnetization. The combined time constant is called T_2 star and is given the symbol T_2^* . The relationship between the T_2 from molecular processes and that from inhomogeneities in the magnetic field is as follows.

$$1/T_2^* = 1/T_2 + 1/T_{2\text{inhomo}} \quad (2.4)$$

2.4 Rotating Frame of Reference

We have just looked at the behavior of spins in the laboratory frame of reference. It is convenient to define a rotating frame of reference, which rotates about the Z axis at the Larmor frequency. We distinguish this rotating coordinate system from the laboratory system by primes on the X and Y axes, X' and Y'.

A magnetization vector rotating at the Larmor frequency in the laboratory frame appears stationary in a frame of reference rotating about the Z axis. In the rotating frame, relaxation of M_Z magnetization to its equilibrium value looks the same as it did in the laboratory frame.

A transverse magnetization vector rotating about the Z axis at the same velocity as the rotating frame will appear stationary in the rotating frame. A magnetization vector traveling faster than the rotating frame rotates clockwise about the Z axis. A magnetization vector traveling slower than the rotating frame rotates counter-clockwise about the Z axis.

2.5 Pulsed Magnetic Fields

A coil of wire placed around the X axis will provide a magnetic field along the X axis when a direct current is passed through the coil (Figure 2.7).

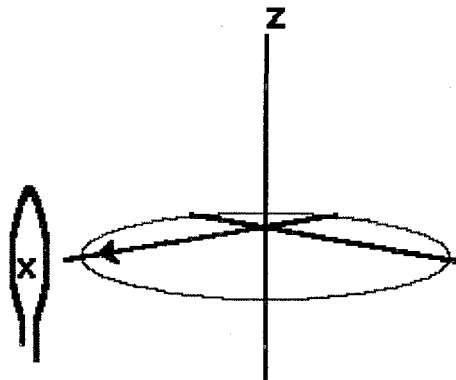


Figure 2.7 A coil of wire that produces magnetic field around X axis.

An alternating current will produce a magnetic field, which alternates in direction.

In a frame of reference rotating about the Z axis at a frequency equal to that of the alternating current, the magnetic field along the X' axis will be constant, just as in the direct current case in the laboratory frame (Figure 2.8).

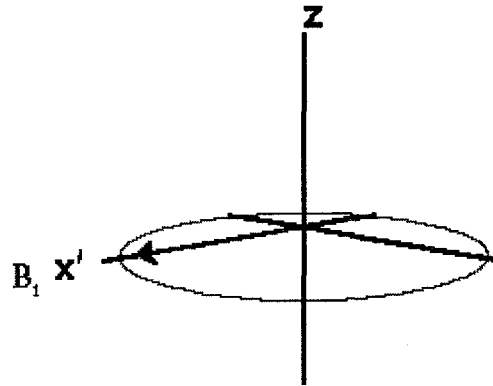


Figure 2.8 The magnetic field is constant in a rotating frame of reference.

This is the same as moving the coil about the rotating frame coordinate system at the Larmor Frequency. In magnetic resonance, the magnetic field created by the coil passing an alternating current at the Larmor frequency is called the B_1 magnetic field. When the alternating current through the coil is turned on and off, it creates a pulsed B_1 magnetic field along the X' axis.

The spins respond to this pulse in such a way as to cause the net magnetization vector to rotate about the direction of the applied B_1 field. The rotation angle depends on the length of time the field is on, τ , and its magnitude B_1 .

$$\theta = 2\pi\gamma\tau B_1 \quad (2.5)$$

In our examples, τ will be assumed to be much smaller than T_1 and T_2 .

A 90° pulse is one which rotates the magnetization vector clockwise by 90 degrees about the X' axis, down to the Y' axis (Figure 2.9).

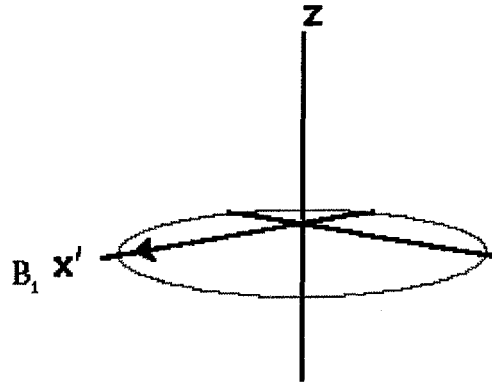


Figure 2.9 A 90° pulse rotates the equilibrium magnetization down to the Y' axis.

In the laboratory frame the equilibrium magnetization spirals down around the Z axis to the XY plane. You can see why the rotating frame of reference is helpful in describing the behavior of magnetization in response to a pulsed magnetic field.

A 180° pulse will rotate the magnetization vector by 180 degrees and the equilibrium magnetization points towards $-Z$ axis (Figure 2.10).

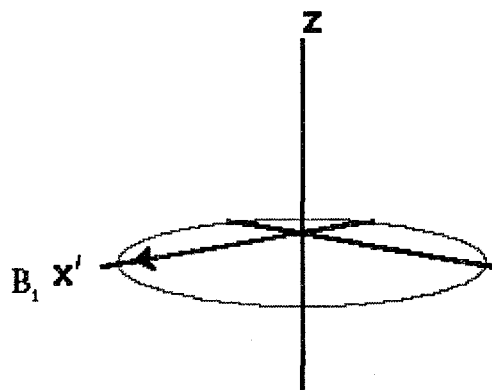


Figure 2.10 A 180° pulse rotates the equilibrium magnetization down to along the $-Z$ axis.

The net magnetization at any orientation will behave according to the rotation equation. For example, a net magnetization vector along the Y' axis will end up along the Y' axis when acted upon by a 180° pulse of B_1 along the X' axis.

2.6 T_1 , T_2 , & PD Images

The spin-lattice relaxation time (T_1), spin-spin relaxation time (T_2), and the proton density (PD) are properties of the spins in tissues. The values of these quantities change from one normal tissue to the next, and from one diseased tissue to the next. They are therefore responsible for contrast between tissues in the various image types.

There are several methods of calculating T_1 , T_2 , and PD values. These methods are applied to individual pixels to produce a calculated T_1 , T_2 , or PD image. The smaller the voxel corresponding to a pixel, the more likely the T_1 , T_2 , and PD values are to represent values for a single tissue. The larger the voxel, the more likely the calculated values are to represent that of a combination of tissue components.

The calculation of T_1 , T_2 , or PD starts with the collection of a series of images. For example, if you wish to produce a T_2 image, a spin-echo pulse sequence is used and a series of images are recorded with varying TE. The signal for a given pixel can be plotted for each TE value and the best fit line from the spin-echo equation drawn through the data to find T_2 (Figure 2.11).

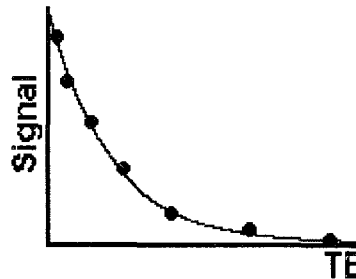


Figure 2.11 The signal versus TE and the best fit line.

A T_1 image can be created from the same pulse sequence using a series of images with varying TR. The signal for a given pixel can be plotted for each TR value and the best fit line from the spin-echo equation drawn through the data to find T_1 (Figure 2.12).

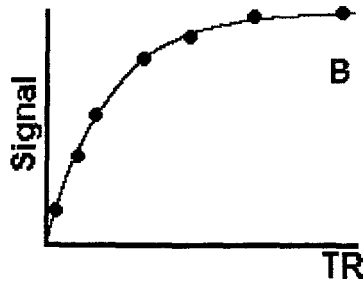


Figure 2.12 The signal versus TR and the best fit line.

The spin density can be calculated once T_1 and T_2 are found using the spin echo signal equation and any spin echo signal. [22]

3. TISSUE CLASSIFICATION

Tissue classification, or image segmentation as it is also called, is the identification of tissues in a magnetic resonance image. The classification is based on a property of the tissue in the image. For example, in the spin-echo image in Figure 3.1, where cerebral spinal fluid (CSF) and gray matter (GM) are bright compared to other tissues, the intensity of the pixel could be used to classify the tissue as CSF and GM or other tissues.

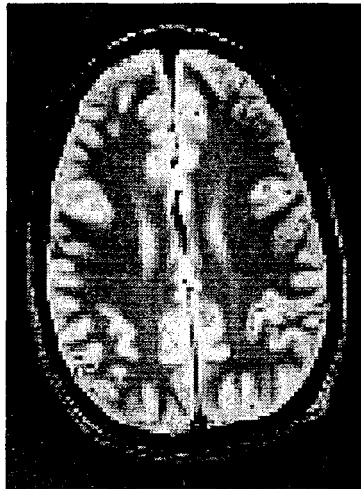


Figure 3.1 A spin echo image that shows intensity of pixels of tissues.

The histogram and look up table for this image is shown in Figure 3.2.

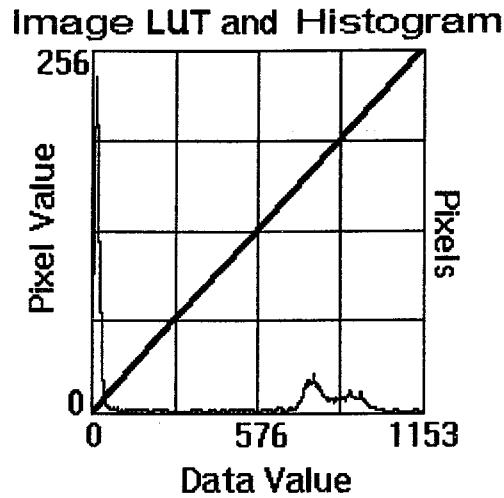


Figure 3.2 The histogram and look up table for the image in Figure 3.1.

Typically we would use a linear relationship between data value and pixel intensity. Furthermore the red, green, and blue content of each pixel would always be the same so as to get shades of gray for the pixels. We can segment the CSF and GM from other tissues in this example by modifying the lookup table such that the green and blue components of a pixel are turned off for data values greater than 865 (Figure 3.3).

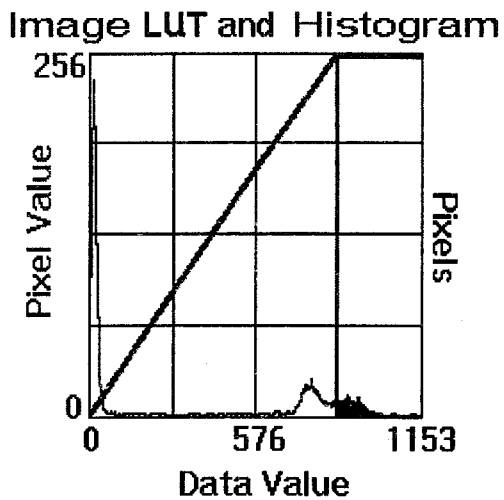


Figure 3.3 Lookup table to segment data values greater than 865 from other values.

This procedure will create red CSF and GM pixels as seen in Figure 3.4. These images are used to create a three-dimensional histogram as seen in Figure 3.5. Similar tissue types

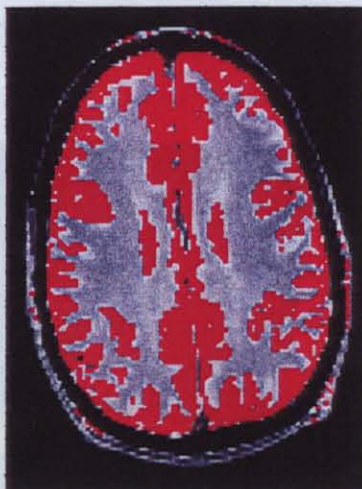


Figure 3.4 Segmented image by using the lookup table in Figure 3.3.

The image is therefore segmented into two classes of tissues: (1) GM and CSF, and (2) tissues other than GM and CSF.

The segmentation process is done with the aid of computer algorithms. These algorithms can segment with more advanced logic than the simple greater than a given data value example given above. Many different types of images, or spectral regions, can be used to segment the tissues. Some of the possible spectral regions include: T_1 , T_2 , and PD weighted; pure T_1 , T_2 , and PD; angiography; diffusion; chemical shift; and functional images. Some of these images are more difficult to work with. Images that show variations in sensitivity of an imaging coil can not be used because segmentation algorithms can not distinguish between an intensity variation caused by the imaging coil sensitivity and the tissue. Calculated T_1 , T_2 , and PD images are easier to work with because they do not show the variation in intensity caused by variations in imaging coil sensitivity.

In the above example it was not possible to segment GM from CSF because the two tissues have similar intensities in the spin-echo image. The more independent spectral regions we work with, the easier it is to segment tissues. For example, segmentation of the

tissues in the brain can be accomplished with calculated T_1 , T_2 , and PD images of the human brain. [23] These images are used to create a three-dimensional histogram as seen in Figure 3.5. Similar tissue types display clusters in the histogram.

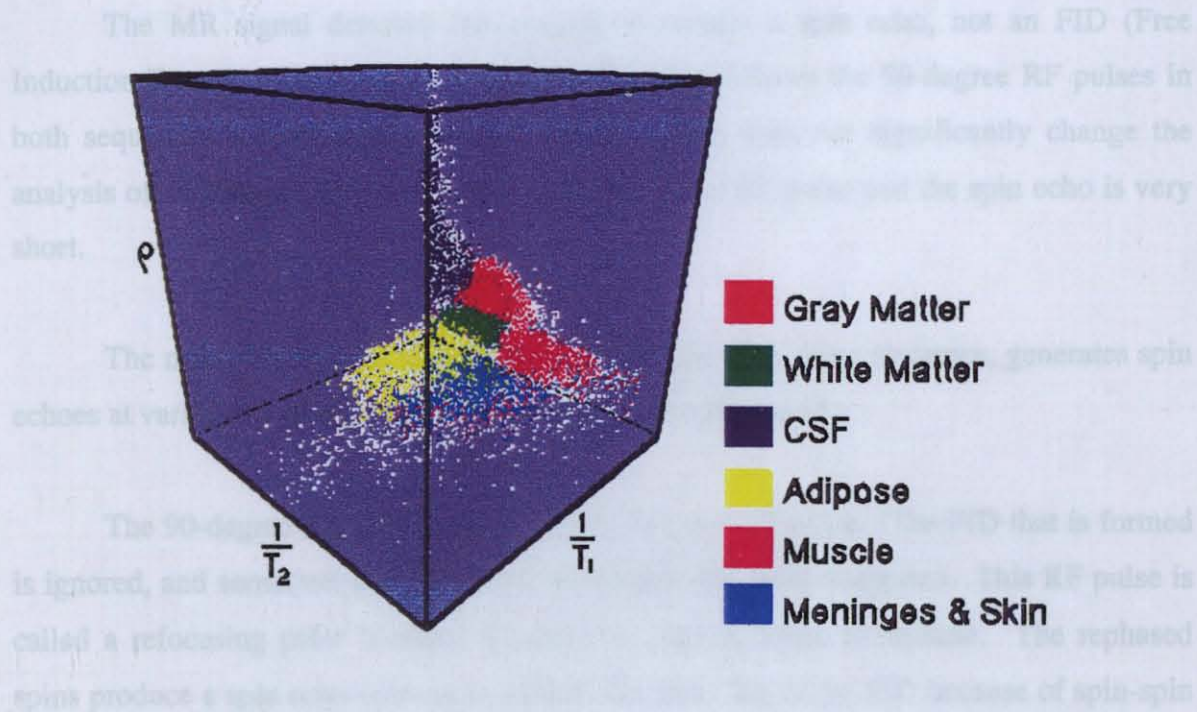


Figure 3.5 Three-dimensional histogram of tissues in the brain.

We can assign pixels in a given range of T_1 , T_2 , and PD values a particular color. The resultant image in Figure 3.6 displays the segmented tissues.

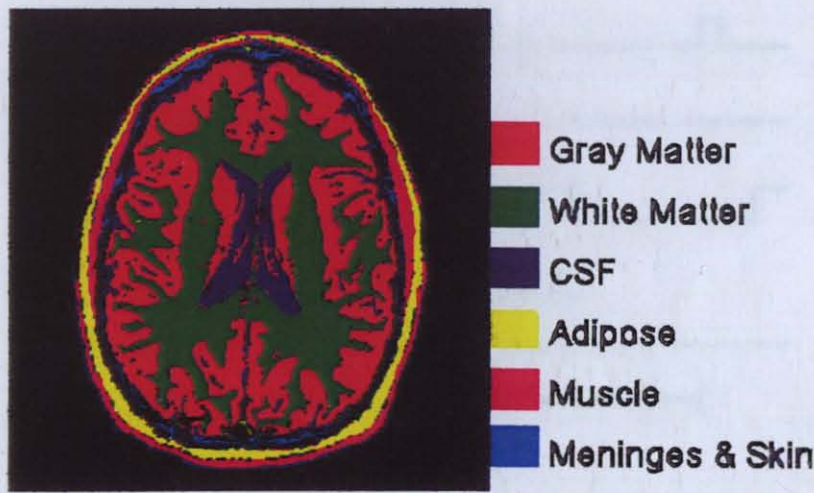


Figure 3.6 Segmented image of the brain.

4. SPIN ECHO

The MR signal detected for imaging is usually a spin echo, not an FID (Free Induction Decay). Therefore a 180-degree RF pulse follows the 90-degree RF pulses in both sequences so that a spin echo is formed. This does not significantly change the analysis of an image if the time between the 90-degree RF pulse and the spin echo is very short.

The most widely used pulse sequence, the spin echo pulse sequence, generates spin echoes at various times after the 90-degree RF pulse (Figure 4.1).

The 90-degree RF pulse rotates the M_z onto the XY plane. The FID that is formed is ignored, and sometime later, $\frac{1}{2} TE$, a 180-degree RF pulse is applied. This RF pulse is called a refocusing pulse because it causes the nuclear spins to rephase. The rephased spins produce a spin echo with an amplitude less than that of the FID because of spin-spin interactions and the resulting loss of transverse magnetization according to the T_2 relaxation time of the tissue. [22]

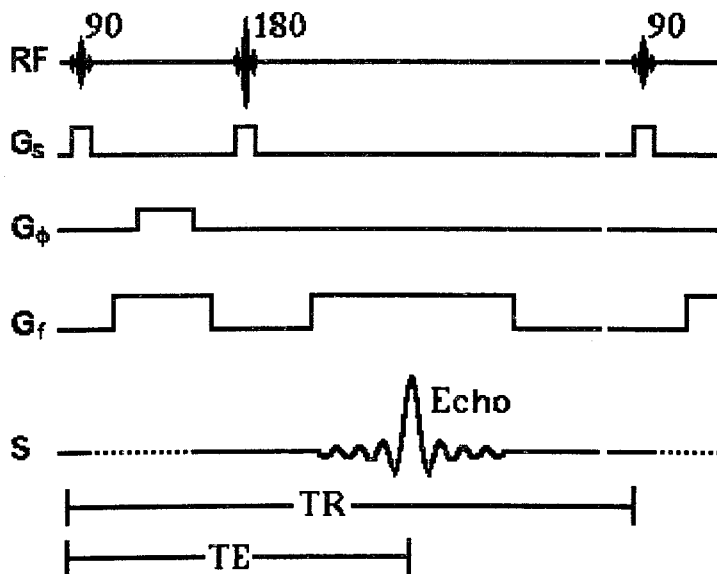


Figure 4.1 Spin echo pulse sequence.

Spin echo images can be PD, T_1 or T_2 weighted, depending on the values of TR and TE.

Figure 4.2 illustrates the T_2 for the three tissues CSF, GM, and WM.

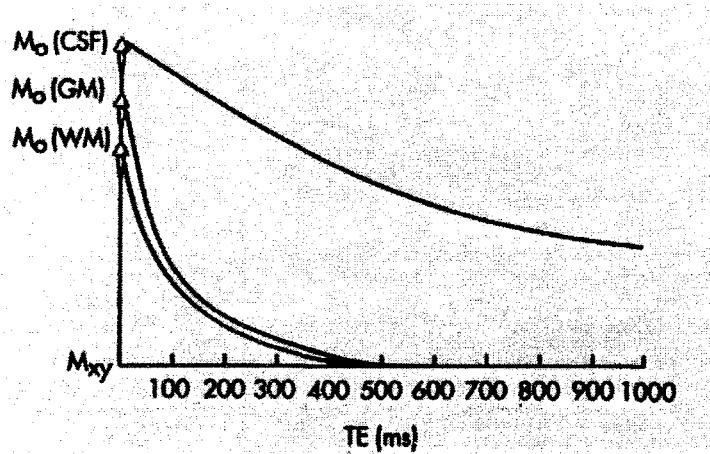


Figure 4.2 T_2 decay curves for CSF, GM, and WM are determined by the amplitude of M_{XY} as a function of TE.

As with the other pulse sequences, the initial signal intensity from CSF, GM, and WM is determined by the equilibrium magnetization, M_0 , for each. Loss of magnetization in the XY plane, M_{XY} and therefore MR signal intensity are determined by the T_2 relaxation time of each tissue. CSF has a relatively long T_2 , approximately 1500 ms. GM and WM have shorter T_2 relaxation times, both approximately 100ms. Therefore the CSF signal remains intense, as seen in Figure 4.3, resulting in a bright pixel for all echo times.

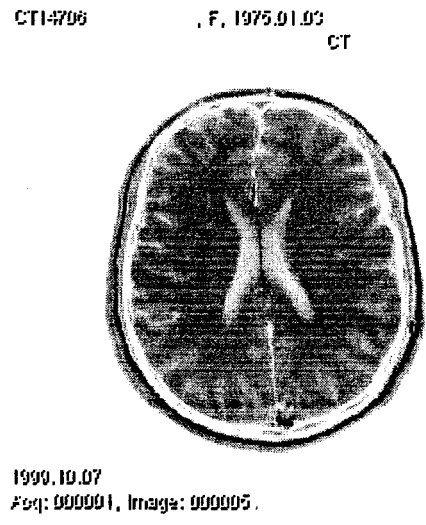


Figure 4.3 A spin echo brain image showing the signal intensities of different tissues.

If TR's for a spin echo pulse sequence were very long, the relationship between spin echoes of the three tissues would be like that shown in Figure 4.4.

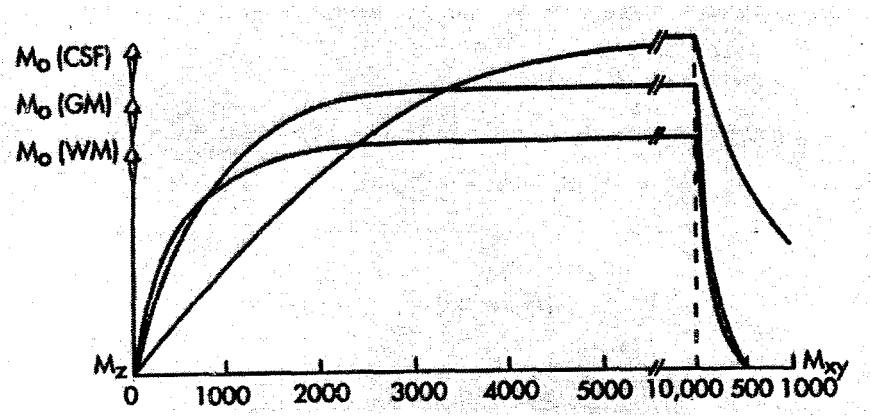


Figure 4.4 Regrowth of M_z and decay of M_{xy} for an extremely long TR.

CSF would always be brightest and WM would always be darkest because with TR at least five times the longest T_1 , all the proton spins have returned to equilibrium before each new pulse sequence.

Figure 4.5 shows the growth of M_z back to equilibrium and the loss of M_{xy} after a 90-degree RF pulse, both plotted on the same scale.

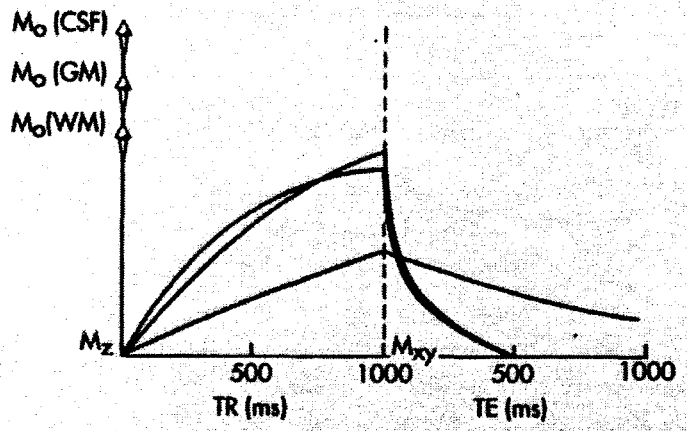


Figure 4.5 Use of an intermediate TR does not allow full regrowth of M_z .

It is clear that the loss of M_{xy} is far more rapid than the regrowth of M_z . At a very long TR, for example, 5000 ms, M_z has returned to equilibrium for nearly all tissues. Subsequent 90-degree RF pulses followed by 180-degree RF refocusing pulses results in images that are PD and T_2 weighted. At long TR, as the echo time is reduced, the image becomes more PD weighted.

If the TR is shortened, more T_1 influence is brought into the image because all tissues will not have recovered to M_0 . A review of Figure 4.4 shows how magnetization among these tissues change with reduced TR. Not only is the initial signal amplitude reduced for subsequent 180-degree refocusing pulses, but the relative amplitudes of the tissues also change.

If the TR is reduced to 1000 ms, GM exhibits a higher initial MR signal than CSF or WM (see Figure 4.4). Therefore GM appears brightest on an early echo, and CSF appears brightest on a late echo. This is a contrast reversal resulting from a crossover.

If the TR is made very short, for example, 300 ms, there is little growth of M_z due to the T_1 relaxation times being too long (Figure 4.6).

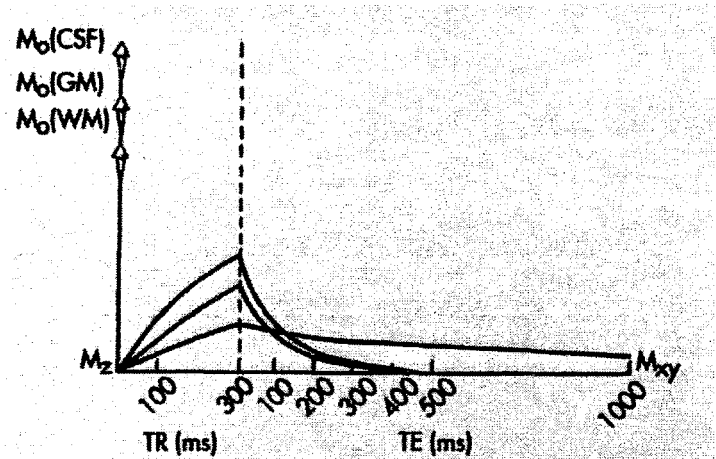


Figure 4.6 With extremely short TR, little M_z is reformed, resulting in more T_1 -weighted images at all clinically employed echo times.

White matter now appears brightest on an early echo. CSF is still brightest on a late echo. These contrast reversals result from crossovers of the T_2 relaxation curves.

- With a long TR and a long TE, the result is a T_2 -weighted image.
- A long TR and a short TE result in a T_1 -weighted image.
- At short TR, both long and short TE results in T_1 -weighted images.

Short TE produces early echoes, and long TE produces late echoes. All spin echo images have a PD contribution. These relationships are shown in Table 4.1.

Table 4.1 Relationship of T_1 , T_2 , and PD weighted images with TR and TE.

	Repetition Time (TR)		
	Short (<300 ms)	Intermediate (300-1000 ms)	Long (>2000)
Short TE (<40 ms)	T_1 weighted	PD and T_1 weighted	PD weighted
Long TE (>80 ms)	T_1 and T_2 weighted	PD, T_1 , and T_2 weighted	T_2 weighted

Depending on the T_1 and T_2 times of adjacent tissues, there will be many opportunities for obscuring contrast at crossovers.

From these graphs and Table 4.1, a little thought shows that the amplitude of the spin echo from any tissue increases as the TR is *lengthened*, as TE is *shortened*, as T_1 *decreases*, and as T_2 *increases*. Stated differently, in general, a long T_1 means *less* signal and a long T_2 means *more* signal.

The signal intensity of successive spin echoes decreases regardless of TR, therefore the SNR also decreases. Long TE results in a dim, noisy image. As a general rule, short TE (early echoes) results in a T_1 -weighted image, and long TE (late echoes) results in a T_2 -weighted image. Both images also have some contribution from PD.

The spin echo images shown in Figure 4.7 represent variations of this most common imaging sequence. When considering spin echo imaging, the interpreter must specify whether the image is from an "early" echo or a "late" echo. These images were made with four different TR times, ranging from 2000 ms to 4000 ms, at a TE of 130 ms. They are all late spin echo images. Late spin echoes are viewed primarily as providing a T_2 -weighted image because the character of a pixel is primarily related to the T_2 of the tissue.

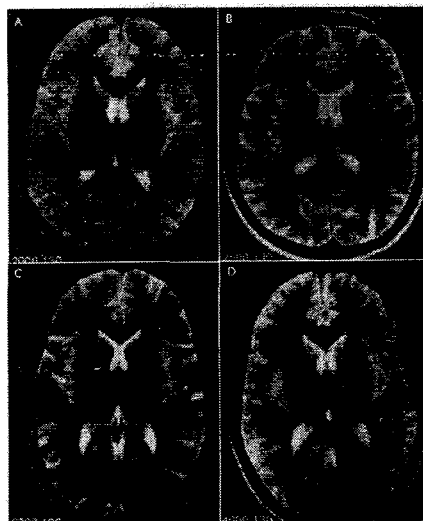


Figure 4.7 Spin echo images produced at various TR times with a long TE result in T_2 -weighted images.

These images, however, are considerably different from the pure T_2 images discussed earlier. The pure T_2 image is dull and has little contrast. On these late spin echo images, there is at least some differentiation between GM and WM. Consequently, some other parameter must contribute to contrast. The additional parameter is PD, which always influences pixel intensity. Longer TR images are brighter because of the return of all tissues to their equilibrium magnetization.

Figure 4.8 shows spin echo images produced at various TR times varying from 350 ms to 4000 ms with a short TE of 35 ms. At long TR, in excess of 1000 ms, the image is PD weighted. At very short TR, about 300 ms, the image is principally T_1 weighted. [24]

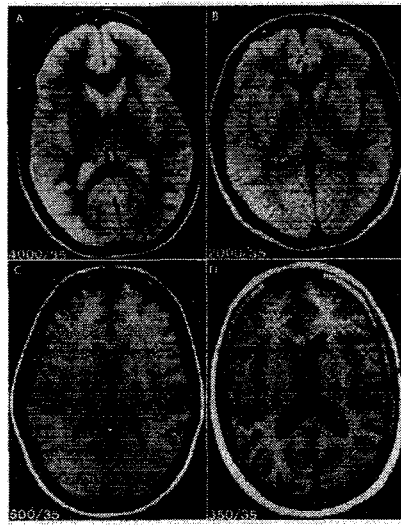


Figure 4.8 Spin echo images produced at various TR times with short TE result in varying PD- and T_1 -weighted images.

The mathematical expression for a saturation-recovery spin-echo sequence is as follows in Equation 4.1.

$$A_E = M_z^0 (1 - e^{-TR/T_1}) e^{-TE/T_2} \quad (4.1)$$

The signal expression in Equation 4.1 indicates that the image intensity of the spin echo sequence carries a T_1 -weighting, T_2 -weighting and spin-density-weighting simultaneously, in general. However, we can selectively emphasize one of contrast mechanisms by properly choosing the sequence parameters TR and TE. For instance, if a short TE is used, the term e^{-TE/T_2} approaches 1 and the T_2 -weighting factor can then be ignored. Effects of different sequence parameter values on image contrast for this excitation scheme were summarized in Table 4.1. [25]

5. TRUE T₁ AND TRUE PD IMAGES

For this research, the TR was varied from 400 ms to 1150 ms by 50 ms increments. By keeping TE constant, 16 images in Figure 5.1 were obtained with different TR/T₁ exponential constants.

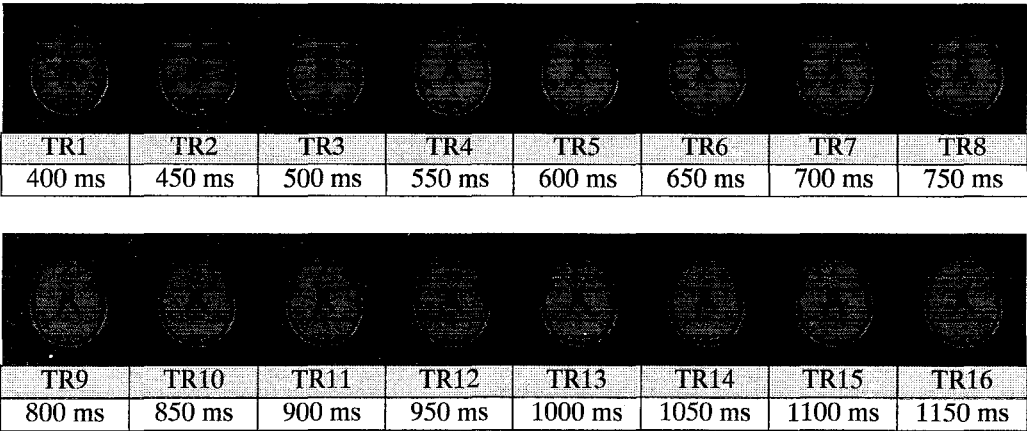


Figure 5.1 Spin echo images produced at successive TR times with short TE.

The images were taken from slices of 4 mm thickness and their sizes are 256 x 256. The pixel values in 16 images were plotted against TR values as shown in Figure

5.2.

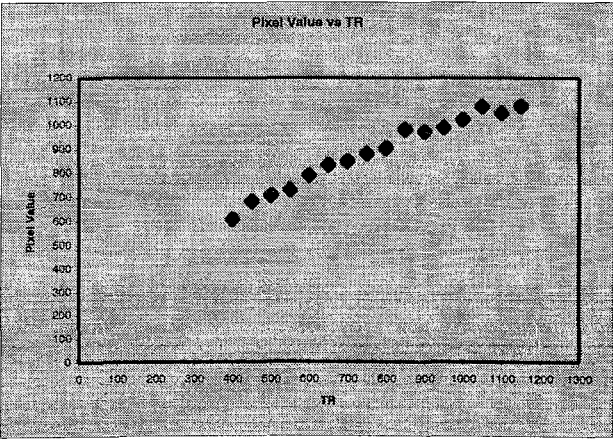


Figure 5.2 Pixel value versus TR.

At the next step, Levenberg-Marquardt Method is applied to the data and T_1 and PD values were estimated.

A sample data together with the fitted exponential curve and the estimated parameters, P1 and P2 (they stand for PD and T_1) is shown in Figure 5.3.

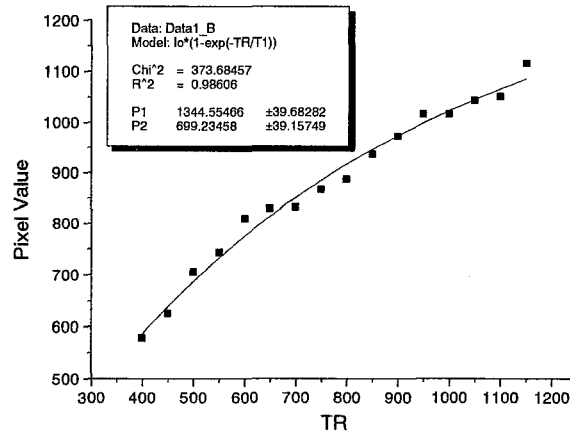


Figure 5.3 A sample dataset of estimated parameters and the best fit.

The fitting algorithm was applied to all pixels and (T_1 , PD) values were recorded for each coordinate. By using these estimated values the calculated T_1 (see Figure 5.4) and calculated PD image (see Figure 5.5) are created.

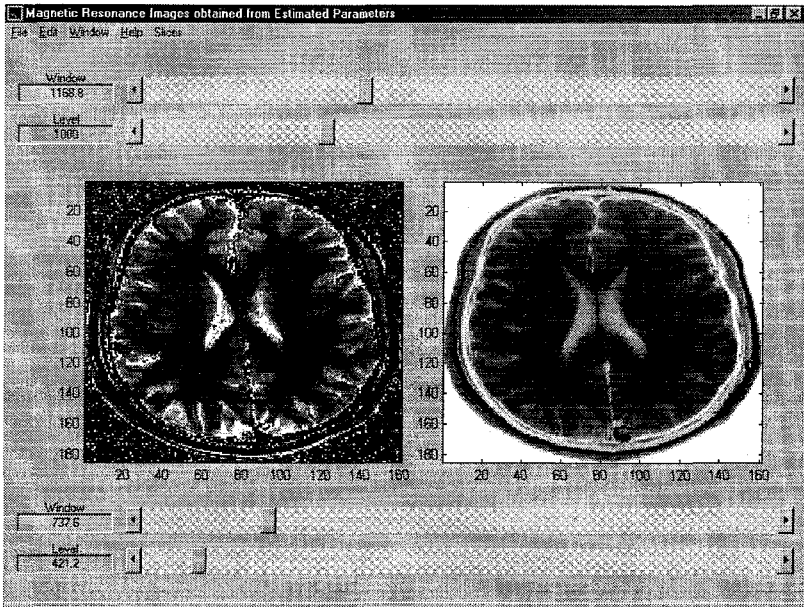


Figure 5.4 Calculated and weighted T_1 images.

The image on the left is the true T_1 image, next to it is the T_1 weighted image taken with spin echo imaging technique and with $TR=400$ ms.

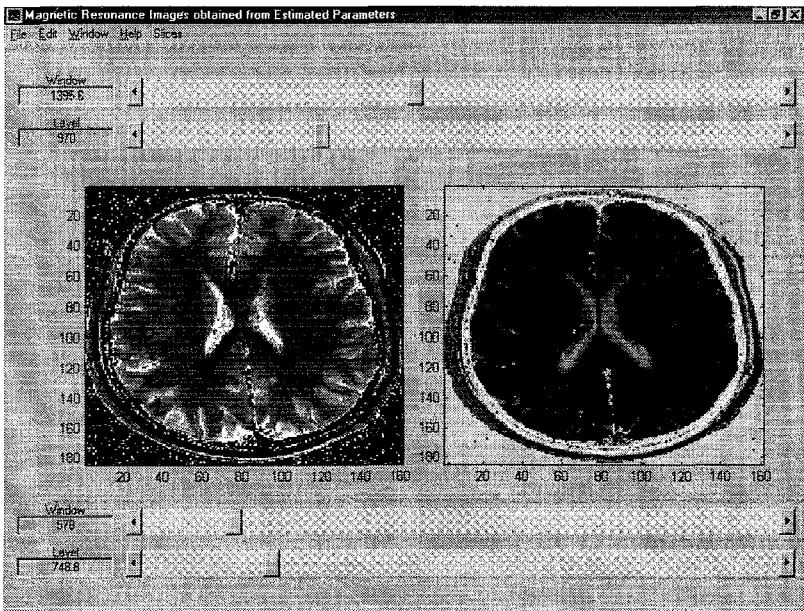


Figure 5.5 Calculated T_1 and calculated PD images.

The image on the left is the true T_1 image, next to it is the true PD image.

Both the calculated T_1 and calculated PD values are on a scale that can not be viewed without reducing them into 0-255 gray level window (Figure 5.6). To choose the window-levels, two sliders for each image are used. The sliders at the top of the user interface, control the gray level of the true T_1 image. The sliders at the bottom of the user interface, control the gray level of the true PD image.

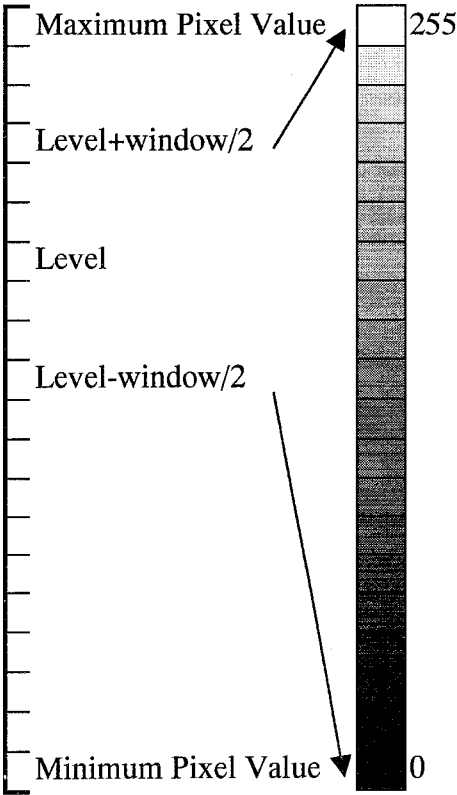


Figure 5.6 Window-level mechanism that is used to display the images in gray scale.

6. SEGMENTATION

6.1 Segmentation of Weighted and Calculated Images

In my research I used maximum likelihood method to choose the class that each pixel of our image belongs. First, two scatter diagrams are produced where each data point is a function of T_1 and PD. One of these scatter diagrams is achieved by using the T_1 -weighted (TR1=400 ms.) and PD-weighted (TR16=1150 ms.) images (Figure 6.1).

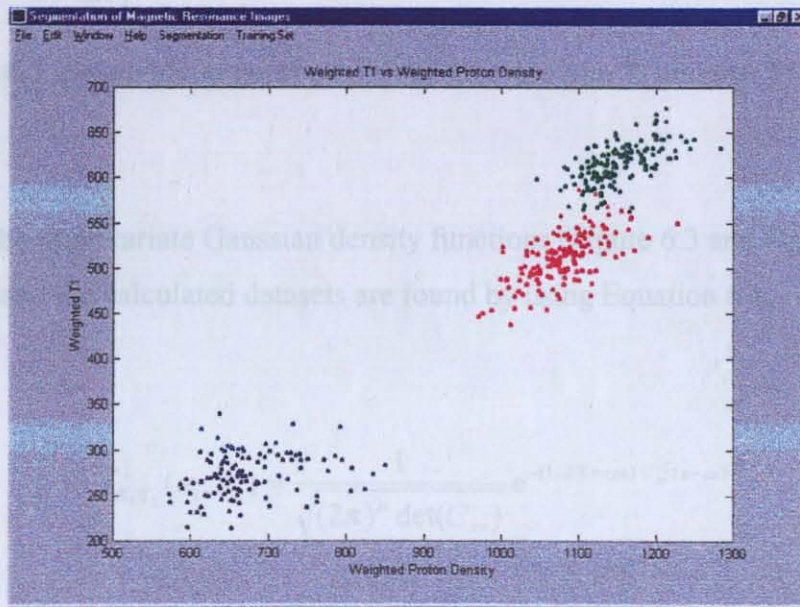


Figure 6.1 Scatter diagram of pixels chosen from T_1 -weighted and PD-weighted images.

Blue dataset is CSF, red dataset is GM, and green dataset is WM. The training dataset that will be used in weighted image segmentation is created by choosing some coordinates on the brain slices. The same coordinates are also necessary to set up the training dataset that will be used in calculated image segmentation. That training dataset is shown in Figure 6.2.

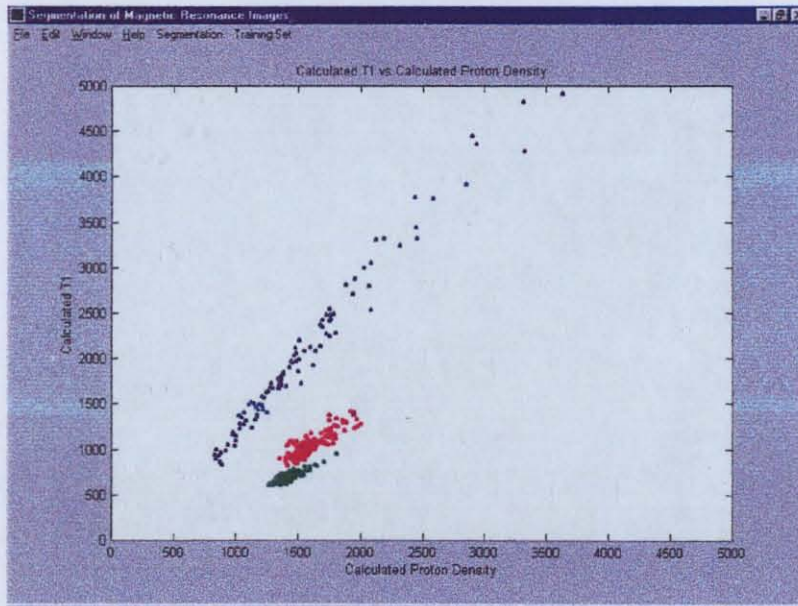


Figure 6.2 Scatter diagram of pixels chosen from true T_1 and true PD images.

Next the multivariate Gaussian density functions (Figure 6.3 and Figure 6.4) of both the weighted and the calculated datasets are found by using Equation 6.1.

$$f_{X_1, X_2}(x_1, x_2) = \frac{1}{\sqrt{(2\pi)^n \det(C_{xx})}} e^{-\frac{1}{2}(x - \mu_x)' C_{xx}^{-1} (x - \mu_x)} \quad (6.1)$$

where x and μ_x are each 2 by 1 vectors with components x_k and $E\{X_k\}$, respectively, for $1 \leq k \leq 2$ and C_{xx} is the associated 2 by 2 covariance matrix. The symbol $\det(C_{xx})$ denotes the determinant of the covariance matrix. [26]

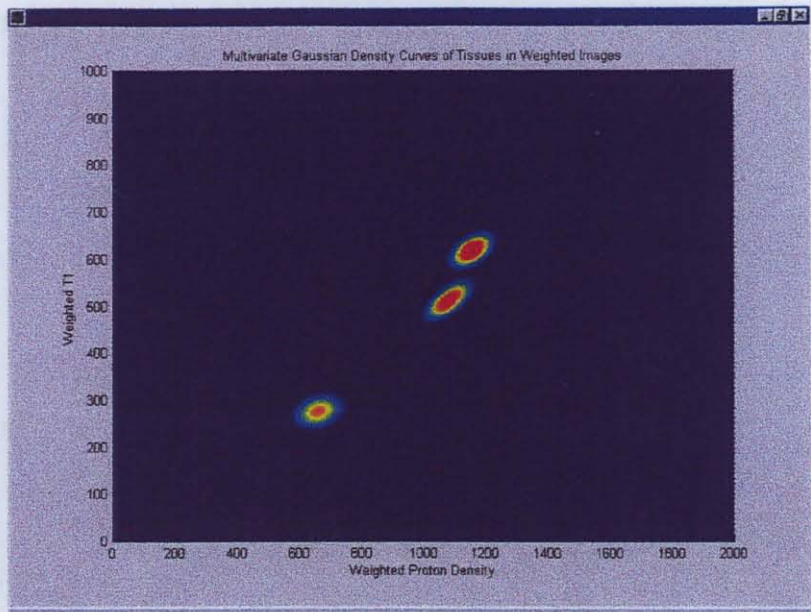


Figure 6.3 The multivariate Gaussian density function of the weighted dataset.

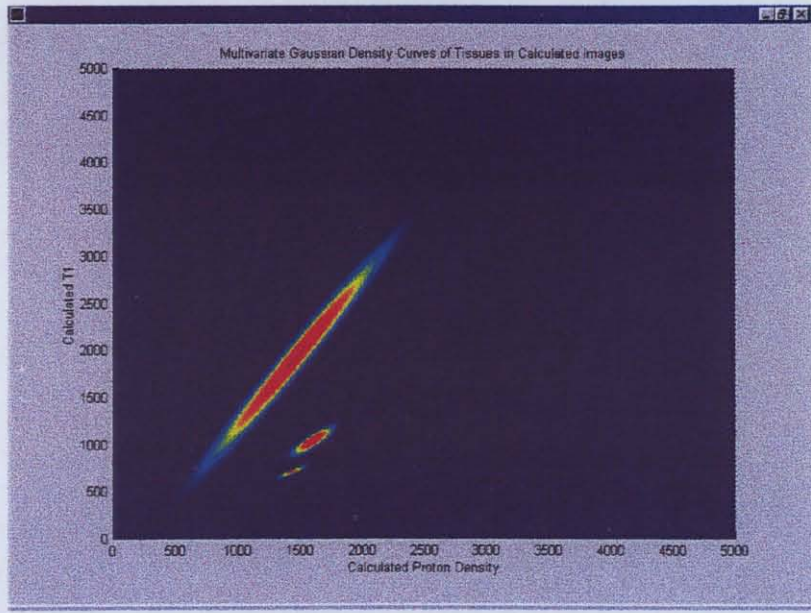


Figure 6.4 The multivariate Gaussian density function of the calculated dataset.

The segmentation algorithm based on the maximum likelihood classification method that utilizes the multivariate Gaussian density functions is applied to the 4 human head slices and the results are illustrated in Figures 6.5-6.12.

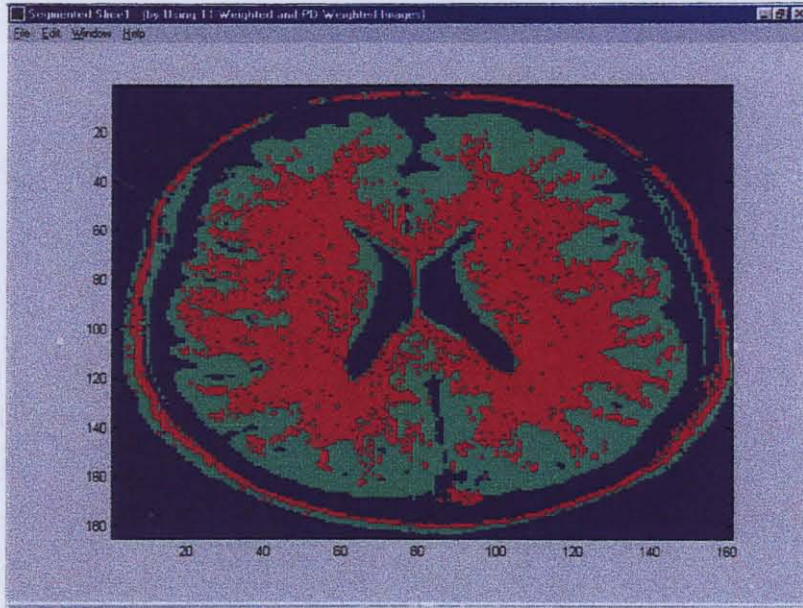


Figure 6.5 Segmented slice1 by using weighted dataset.

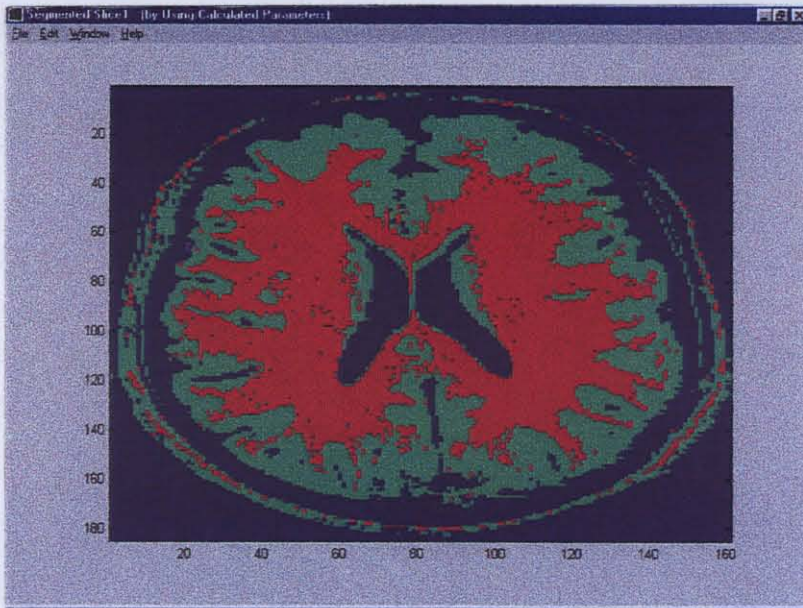


Figure 6.6 Segmented slice1 by using calculated dataset.

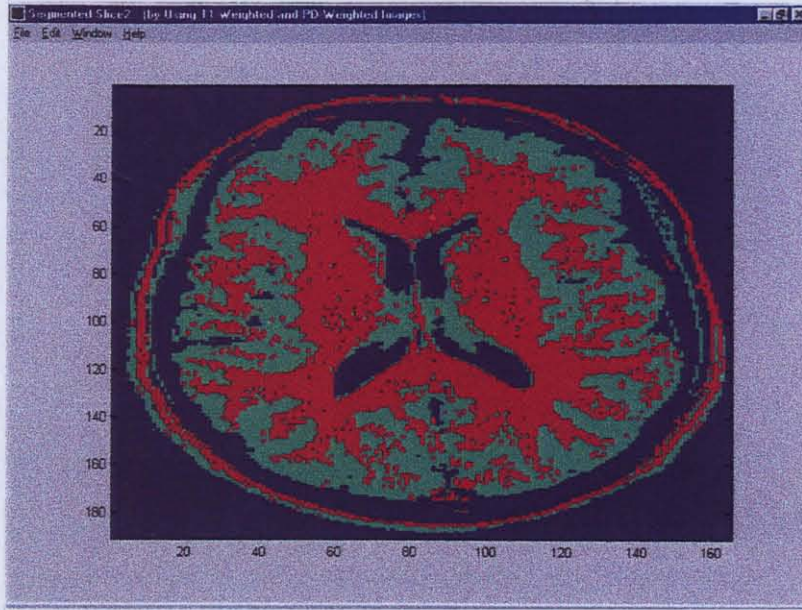


Figure 6.7 Segmented slice2 by using weighted dataset.

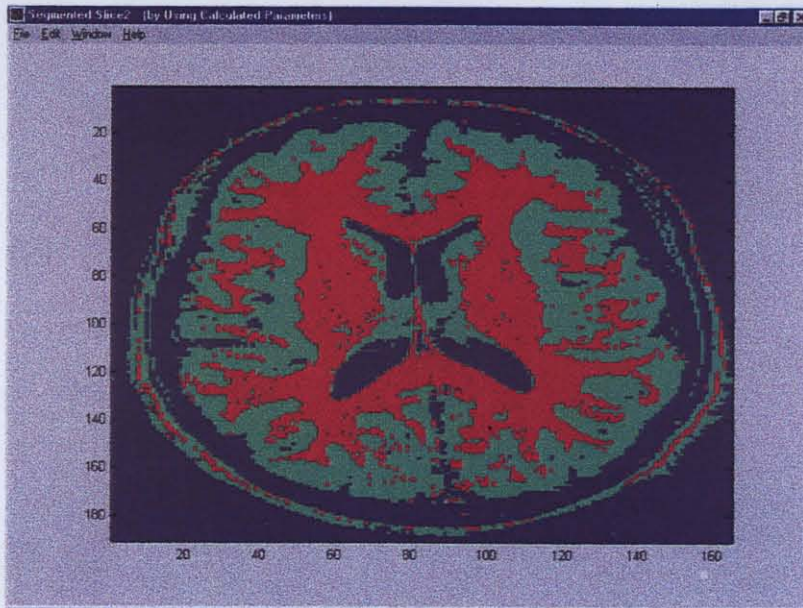


Figure 6.8 Segmented slice2 by using calculated dataset.

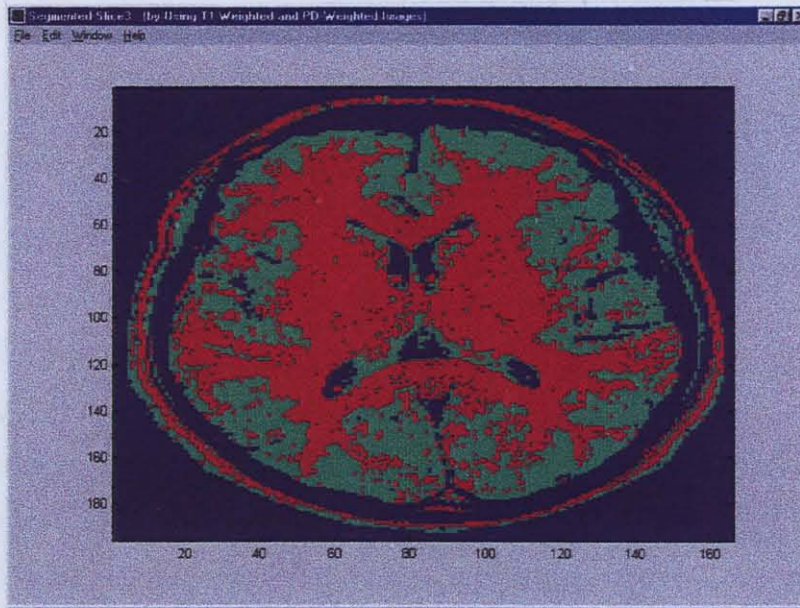


Figure 6.9 Segmented slice3 by using weighted dataset.

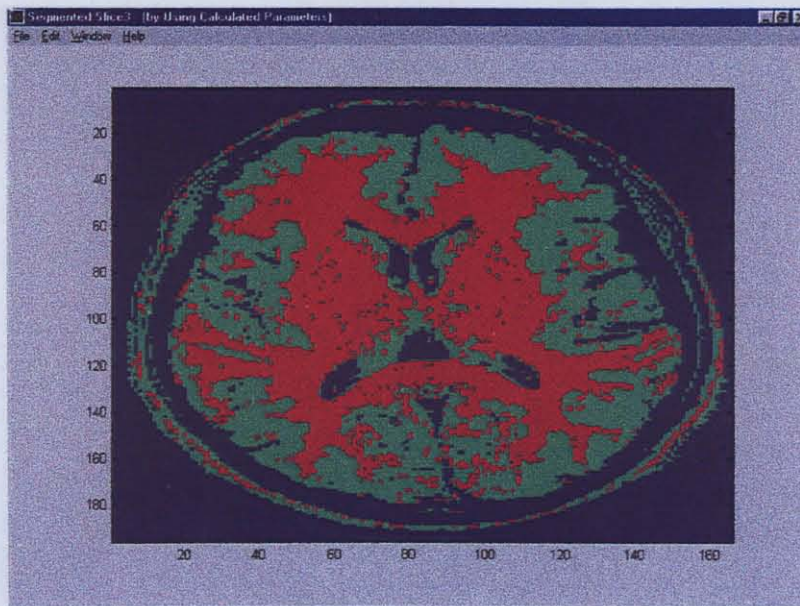


Figure 6.10 Segmented slice3 by using calculated dataset.

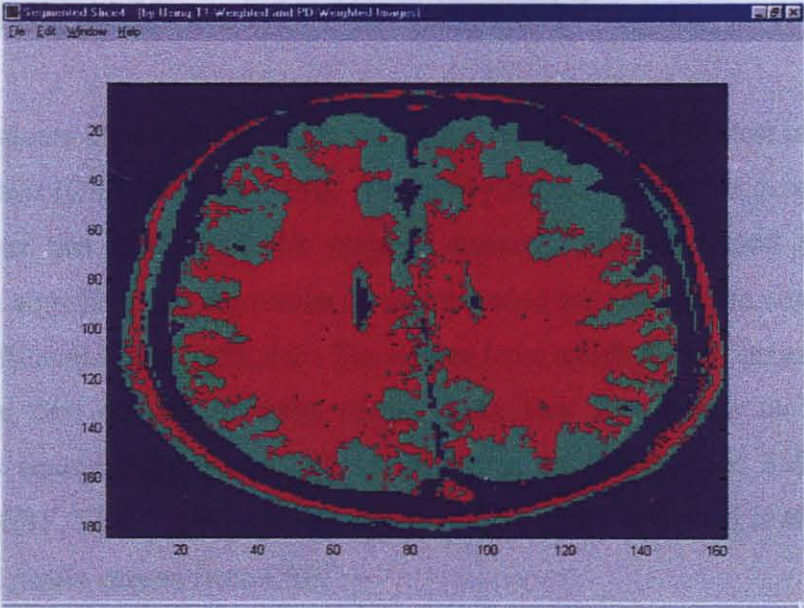


Figure 6.11 Segmented slice4 by using weighted dataset.

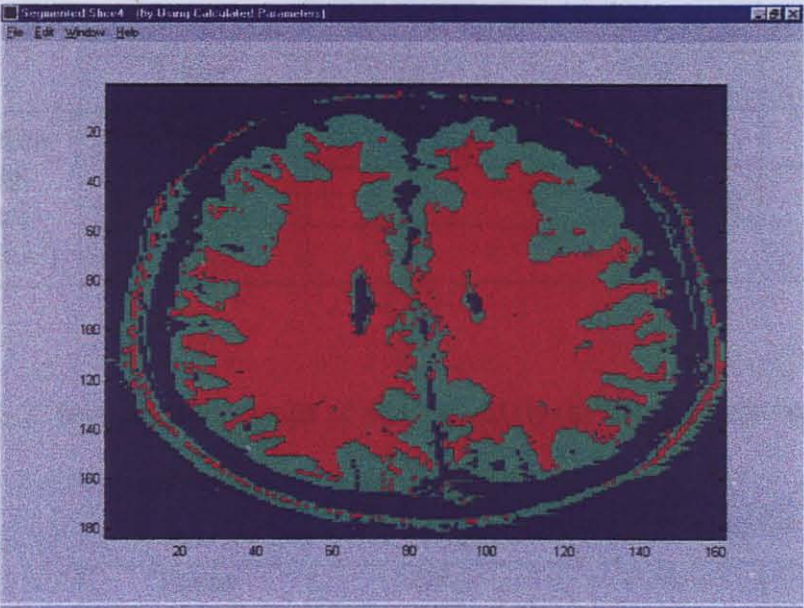


Figure 6.12 Segmented slice4 by using calculated dataset.

6.2 Test of Segmentation by Confusion Matrix

To evaluate the reliability of the segmentation algorithm, 400 test pixels are chosen from each tissue (CSF, WM, and GM). The newly chosen pixels (test pixels) are different from the ones that constituted the training dataset. They are used as input to the segmentation algorithm and the results of segmentation are arranged in confusion matrices as shown in Table 6.1. and Table 6.2. The tissues from which the test pixels are chosen are placed at the first column of these tables. The first row shows the distribution of segmentation among three tissues. For example, in the second row, 348 test pixels are classified as CSF, 52 test pixels are classified as GM, and 0 test pixel is classified as WM from 400 test pixels chosen from CSF.

Table 6.1 Segmentation results of 400 test pixels (by using weighted dataset).

		Result of segmentation		
Tissues from which the pixels are chosen	Tissue	CSF	GM	WM
	CSF	348	52	0
	GM	0	356	44
	WM	0	40	360

Table 6.2 Segmentation results of 400 test pixels (by using calculated dataset).

		Result of segmentation		
Tissues from which the pixels are chosen	Tissue	CSF	GM	WM
	CSF	360	40	0
	GM	0	388	12
	WM	0	8	392

7. DISCUSSION AND CONCLUSION

To evaluate the correctness of the segmentation algorithm the confusion matrix method is used. The test pixels are chosen from the coordinates that are not used when setting up the training datasets. To be certain at doing this, the training datasets are constituted from the tissues on the left lobe of the brain, and the test pixels are chosen from the tissues on the right lobe of the brain. The segmentation done with the calculated datasets gave more satisfactory results than the segmentation done with the weighted datasets.

In this study, development of a segmentation algorithm by using calculated T_1 and calculated PD images is aimed. For this purpose the true T_1 and true PD images are produced by finding the best exponential curves fitting to the images taken from an MR imager, and the segmentation is achieved by using the maximum likelihood method. We conclude that the segmentation done with the calculated datasets gave very satisfactory results and that it will be helpful in detecting the tumors in the brain.

7.1 Future Directions

For the segmentation algorithm to be used clinically for classifying tissues, the training datasets of other tissues should be constituted. Since calculated T_1 , T_2 , and ρ images do not show the variation in intensity caused by variations in imaging coil sensitivity, different MR imagers can be used to set up the training sets.

By changing TE and holding TR fixed, T_2 weighted images can be produced by using spin echo pulse sequence. True T_2 parameters of tissues can be calculated using these images. By adding true T_1 and true PD parameters calculated in this study, three dimensional scatter diagrams can be plotted. Finally, the reproducibility and reliability of classification must be studied by comparing results of segmentation of real (weighted) images and calculated (true) images by expert operators.

APPENDIX

LISTING OF SOFTWARE

Matlab Code of Calculated and Weighted Image User Interface: xxboth.m

```

h=figure('color',[.75 .75 .75], ...
    'numbertitle','off', ...
    'name','Magnetic Resonance Images obtained from Estimated Parameters',
    ...
    'position',[1 29 800 534]);

ws=1000;max_ws=3000;min_ws=200;
me=1000;max_me=3000;min_me=200;

ws2=1000;max_ws2=3000;min_ws2=200;
me2=1000;max_me2=3000;min_me2=200;

txt_w1=uicontrol('Style','text', ...
    'Pos',[10 380 72 15], ...
    'Units','points', ...
    'String','Window');
sl_w1=uicontrol('Style','slider', ...
    'Position',[92 371 502 20], ...
    'Units','points', ...
    'Value',ws, ...
    'Max',max_ws, ...
    'Min',min_ws, ...

'Callback','ws=get(sl_w1,'Value');set(ed_w1,'String',num2str(ws));win
_range');
ed_w1=uicontrol('Style','edit', ...
    'Pos',[10 371 72 15], ...
    'Units','points', ...
    'String',num2str(ws), ...
    'Callback','ws=str2num(get(ed_w1,'String'))');
set(sl_w1,'Value',ws);

txt_w2=uicontrol('Style','text', ...
    'Pos',[10 349 72 15], ...
    'Units','points', ...
    'String','Level');
sl_w2=uicontrol('Style','slider', ...
    'Position',[92 340 502 20], ...
    'Units','points', ...
    'Value',me, ...
    'Max',max_me, ...
    'Min',min_me, ...

'Callback','me=get(sl_w2,'Value');set(ed_w2,'String',num2str(me));win
_range');
ed_w2=uicontrol('Style','edit', ...
    'Pos',[10 340 72 15], ...
    'Units','points', ...
    'String',num2str(me), ...

```

```

        'Callback','me=str2num(get(ed_w2,'String')));
set(sl_w2,'Value',me));
txt_w3=uicontrol('Style','text', ...
    'Pos',[10 61 72 15], ...
    'Units','points', ...
    'String','Window');
sl_w3=uicontrol('Style','slider', ...
    'Position',[92 52 502 20], ...
    'Units','points', ...
    'Value',ws2, ...
    'Max',max_ws2, ...
    'Min',min_ws2, ...

'Callback','ws2=get(sl_w3,'Value');set(ed_w3,'String',num2str(ws2));c
trl_im');
ed_w3=uicontrol('Style','edit', ...
    'Pos',[10 52 72 15], ...
    'Units','points', ...
    'String',num2str(ws2), ...
    'Callback','ws2=str2num(get(ed_w3,'String')));
set(sl_w3,'Value',ws2));

txt_w4=uicontrol('Style','text', ...
    'Pos',[10 30 72 15], ...
    'Units','points', ...
    'String','Level');
sl_w4=uicontrol('Style','slider', ...
    'Position',[92 21 502 20], ...
    'Units','points', ...
    'Value',me2, ...
    'Max',max_me2, ...
    'Min',min_me2, ...

'Callback','me2=get(sl_w4,'Value');set(ed_w4,'String',num2str(me2));c
trl_im');
ed_w4=uicontrol('Style','edit', ...
    'Pos',[10 21 72 15], ...
    'Units','points', ...
    'String',num2str(me2), ...
    'Callback','me2=str2num(get(ed_w4,'String')));
set(sl_w4,'Value',me2));

menul=uimenu('parent', h, ...
    'Label','Slices');

menul_1=uimenu('parent', menul, ...
    'Label','Slice1');
menul_2=uimenu('parent', menul, ...
    'Label','Slice2');
menul_3=uimenu('parent', menul, ...
    'Label','Slice3');
menul_4=uimenu('parent', menul, ...
    'Label','Slice4');

```

Matlab Code of Calculated T_1 and Calculated PD Image User Interface: xxt1pd.m

```

h=figure('color',[.75 .75 .75], ...
    'numbertitle','off', ...
    'name','Magnetic Resonance Images obtained from Estimated Parameters',
    ...
    'position',[1    29    800    534]);

ws=1000;max_ws=3000;min_ws=200;
me=1000;max_me=3000;min_me=200;

ws2=1000;max_ws2=3000;min_ws2=200;
me2=1000;max_me2=3000;min_me2=200;

txt_w1=uicontrol('Style','text', ...
    'Pos',[10 380 72 15], ...
    'Units','points', ...
    'String','Window');
sl_w1=uicontrol('Style','slider', ...
    'Position',[92 371 502 20], ...
    'Units','points', ...
    'Value',ws, ...
    'Max',max_ws, ...
    'Min',min_ws, ...

'Callback','ws=get(sl_w1,'Value');set(ed_w1,'String',num2str(ws));win
_range');
ed_w1=uicontrol('Style','edit', ...
    'Pos',[10 371 72 15], ...
    'Units','points', ...
    'String',num2str(ws), ...
    'Callback','ws=str2num(get(ed_w1,'String'))');
set(sl_w1,'Value',ws);

txt_w2=uicontrol('Style','text', ...
    'Pos',[10 349 72 15], ...
    'Units','points', ...
    'String','Level');
sl_w2=uicontrol('Style','slider', ...
    'Position',[92 340 502 20], ...
    'Units','points', ...
    'Value',me, ...
    'Max',max_me, ...
    'Min',min_me, ...

'Callback','me=get(sl_w2,'Value');set(ed_w2,'String',num2str(me));win
_range');
ed_w2=uicontrol('Style','edit', ...
    'Pos',[10 340 72 15], ...
    'Units','points', ...
    'String',num2str(me), ...
    'Callback','me=str2num(get(ed_w2,'String'))');
set(sl_w2,'Value',me);

```

```

txt_w3=uicontrol('Style','text', ...
    'Pos',[10 61 72 15], ...
    'Units','points', ...
    'String','Window');
sl_w3=uicontrol('Style','slider', ...
    'Position',[92 52 502 20], ...
    'Units','points', ...
    'Value',ws2, ...
    'Max',max_ws2, ...
    'Min',min_ws2, ...

    'Callback','ws2=get(sl_w3,'Value');set(ed_w3,'String',num2str(ws2));w
in_rangedp');
ed_w3=uicontrol('Style','edit', ...
    'Pos',[10 52 72 15], ...
    'Units','points', ...
    'String',num2str(ws2), ...
    'CallBack','ws2=str2num(get(ed_w3,'String'))');
set(sl_w3,'Value',ws2));

txt_w4=uicontrol('Style','text', ...
    'Pos',[10 30 72 15], ...
    'Units','points', ...
    'String','Level');
sl_w4=uicontrol('Style','slider', ...
    'Position',[92 21 502 20], ...
    'Units','points', ...
    'Value',me2, ...
    'Max',max_me2, ...
    'Min',min_me2, ...

    'Callback','me2=get(sl_w4,'Value');set(ed_w4,'String',num2str(me2));w
in_rangedp');
ed_w4=uicontrol('Style','edit', ...
    'Pos',[10 21 72 15], ...
    'Units','points', ...
    'String',num2str(me2), ...
    'CallBack','me2=str2num(get(ed_w4,'String'))');
set(sl_w4,'Value',me2));

menul=uimenu('parent', h, ...
    'Label','Slices');

menul_1=uimenu('parent', menul, ...
    'Label','Slice1');
menul_2=uimenu('parent', menul, ...
    'Label','Slice2');
menul_3=uimenu('parent', menul, ...
    'Label','Slice3');
menul_4=uimenu('parent', menul, ...
    'Label','Slice4');

```

Matlab Code of Segmentation: segm.m

```

h=figure('color',[.75 .75 .75], ...
    'numbertitle','off', ...
    'name','Segmentation of Magnetic Resonance Images', ...
    'position',[1 29 800 534]);
% 'menubar','none');

menu1=uimenu('parent', h, ...
    'Label','Segmentation');

menu2=uimenu('parent', h, ...
    'Label','Training Set');

menu1_1=uimenu('parent', menu1, ...
    'Label','Weighted');

menu1_2=uimenu('parent', menu1, ...
    'Label','Calculated');

menu2_1=uimenu('parent', menu2, ...
    'Label','Choose', ...
    'Callback','trainingset');

menu2_2=uimenu('parent', menu2, ...
    'Label','Plot');

menu1_1_1=uimenu('parent', menu1_1, ...
    'Label','Slicel', ...
    'Callback','weight_segmoofslicelnew');

menu1_1_2=uimenu('parent', menu1_1, ...
    'Label','Slice2', ...
    'Callback','weight_segmoofslice2new');

menu1_1_3=uimenu('parent', menu1_1, ...
    'Label','Slice3', ...
    'Callback','weight_segmoofslice3new');

menu1_1_4=uimenu('parent', menu1_1, ...
    'Label','Slice4', ...
    'Callback','weight_segmoofslice4new');

menu1_2_1=uimenu('parent', menu1_2, ...
    'Label','Slicel', ...
    'Callback','calc_segmoofslicelnew');

menu1_2_2=uimenu('parent', menu1_2, ...
    'Label','Slice2', ...
    'Callback','calc_segmoofslice2new');

menu1_2_3=uimenu('parent', menu1_2, ...
    'Label','Slice3', ...
    'Callback','calc_segmoofslice3new');

```

```

menu1_2_4=uimenu('parent', menu1_2, ...
    'Label','Slice4', ...
    'Callback','calc_segmofofslice4new');

menu2_2_1=uimenu('parent', menu2_2, ...
    'Label','Weighted', ...
    'Callback','plot_weight');

menu2_2_2=uimenu('parent', menu2_2, ...
    'Label','Calculated', ...
    'Callback','plot_calc');

```

Matlab Code of Segmentation: weight_segmofofslice1new.m

```

load c:\old\bora\denemel\slicel_pd.mat;
load c:\old\bora\denemel\slicel_tl.mat;

for j=1:185
for k=1:161
    tempo_wm=([slicel_pd(j,k),slicel_Tl(j,k)]'-mean_wm_weight);
    mat_wm(j,k)=c5_wm_weight*exp(-
0.5*tempo_wm'*inv_cov_wm_weight*tempo_wm);

    tempo_gm=([slicel_pd(j,k),slicel_Tl(j,k)]'-mean_gm_weight);
    mat_gm(j,k)=c5_gm_weight*exp(-
0.5*tempo_gm'*inv_cov_gm_weight*tempo_gm);

    tempo_csf=([slicel_pd(j,k),slicel_Tl(j,k)]'-mean_csf_weight);
    mat_csf(j,k)=c5_csf_weight*exp(-
0.5*tempo_csf'*inv_cov_csf_weight*tempo_csf);

end
end

clear j k

mat_slicelw=zeros(185,161);

for j=1:185
for k=1:161

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

mat_slicelw(j,k)=temp;

end
end

clear j k;

```

```

clear temp*;

h=figure('name','Segmented Slice1 - (by Using T1-Weighted and PD-Weighted
Images)', ...
    'numbertitle','off',...
    'menubar','none');

imagesc(mat_slice1w)

```

Matlab Code of Segmentation: weight_segmo slice2new.m

```

load c:\old\bora\deneme1\slice2_pd.mat;
load c:\old\bora\deneme1\slice2_t1.mat;

for j=1:191
for k=1:165
    tempo_wm=([slice2_pd(j,k),slice2_T1(j,k)]'-mean_wm_weight);
    mat_wm(j,k)=c5_wm_weight*exp(-
0.5*tempo_wm'*inv_cov_wm_weight*tempo_wm);

    tempo_gm=([slice2_pd(j,k),slice2_T1(j,k)]'-mean_gm_weight);
    mat_gm(j,k)=c5_gm_weight*exp(-
0.5*tempo_gm'*inv_cov_gm_weight*tempo_gm);

    tempo_csf=([slice2_pd(j,k),slice2_T1(j,k)]'-mean_csf_weight);
    mat_csf(j,k)=c5_csf_weight*exp(-
0.5*tempo_csf'*inv_cov_csf_weight*tempo_csf);

end
end

clear j k

mat_slice2w=zeros(191,165);

for j=1:191
for k=1:165

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

mat_slice2w(j,k)=temp;

end
end

clear j k;
clear temp*;

```

```

h=figure('name','Segmented Slice2 - (by Using T1-Weighted and PD-Weighted
Images)', ...
    'numbertitle','off',...
    'menubar','none');

imagesc(mat_slice2w)

```

Matlab Code of Segmentation: weight_segmo slice3new.m

```

load c:\old\bora\deneme1\slice3_pd.mat;
load c:\old\bora\deneme1\slice3_t1.mat;

for j=1:196
for k=1:166
    tempo_wm=([slice3_PD(j,k),slice3_T1(j,k)]'-mean_wm_weight);
    mat_wm(j,k)=c5_wm_weight*exp(-
0.5*tempo_wm'*inv_cov_wm_weight*tempo_wm);

    tempo_gm=([slice3_PD(j,k),slice3_T1(j,k)]'-mean_gm_weight);
    mat_gm(j,k)=c5_gm_weight*exp(-
0.5*tempo_gm'*inv_cov_gm_weight*tempo_gm);

    tempo_csf=([slice3_PD(j,k),slice3_T1(j,k)]'-mean_csf_weight);
    mat_csf(j,k)=c5_csf_weight*exp(-
0.5*tempo_csf'*inv_cov_csf_weight*tempo_csf);

end
end

clear j k

mat_slice3w=zeros(196,166);

for j=1:196
for k=1:166

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

mat_slice3w(j,k)=temp;

end
end

clear j k;
clear temp*;

h=figure('name','Segmented Slice3 - (by Using T1-Weighted and PD-Weighted
Images)', ...
    'numbertitle','off',...

```

```

'menubar','none');

imagesc(mat_slice3w)

```

Matlab Code of Segmentation: weight_segmo4slice4new.m

```

load c:\old\bora\denemel\slice4_pd.mat;
load c:\old\bora\denemel\slice4_t1.mat;

for j=1:184
for k=1:162
    tempo_wm=([slice4_PD(j,k),slice4_T1(j,k)]'-mean_wm_weight);
    mat_wm(j,k)=c5_wm_weight*exp(-
0.5*tempo_wm'*inv_cov_wm_weight*tempo_wm);

    tempo_gm=([slice4_PD(j,k),slice4_T1(j,k)]'-mean_gm_weight);
    mat_gm(j,k)=c5_gm_weight*exp(-
0.5*tempo_gm'*inv_cov_gm_weight*tempo_gm);

    tempo_csf=([slice4_PD(j,k),slice4_T1(j,k)]'-mean_csf_weight);
    mat_csf(j,k)=c5_csf_weight*exp(-
0.5*tempo_csf'*inv_cov_csf_weight*tempo_csf);

end
end

clear j k

mat_slice4w=zeros(184,162);

for j=1:184
for k=1:162

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

mat_slice4w(j,k)=temp;

end
end

clear j k;
clear temp*;

h=figure('name','Segmented Slice4 - (by Using T1-Weighted and PD-Weighted
Images)', ...
    'numbertitle','off',...
    'menubar','none');

imagesc(mat_slice4w)

```

Matlab Code of Segmentation: calc_segmo slice1new.m

```

for j=1:185
for k=1:161
    tempo_wm=([pd_mat_slicel(j,k),Tl_mat_slicel(j,k)]'-mean_wm_calc);
    mat_wm(j,k)=c5_wm_calc*exp(-0.5*tempo_wm'*inv_cov_wm_calc*tempo_wm);

    tempo_gm=([pd_mat_slicel(j,k),Tl_mat_slicel(j,k)]'-mean_gm_calc);
    mat_gm(j,k)=c5_gm_calc*exp(-0.5*tempo_gm'*inv_cov_gm_calc*tempo_gm);

    tempo_csf=([pd_mat_slicel(j,k),Tl_mat_slicel(j,k)]'-mean_csf_calc);
    mat_csf(j,k)=c5_csf_calc*exp(-
0.5*tempo_csf'*inv_cov_csf_calc*tempo_csf);

end
end

clear j k

mat_slicelc=zeros(185,161);

for j=1:185
for k=1:161

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

mat_slicelc(j,k)=temp;

end
end

clear j k

h=figure('name','Segmented Slice1 - (by Using Calculated Parameters)',
...
    'numbertitle','off',...
    'menubar','none');

imagesc(mat_slicelc);

```

Matlab Code of Segmentation: calc_segmo slice2new.m

```

for j=1:191
for k=1:165
    tempo_wm=([pd_mat_slice2(j,k),Tl_mat_slice2(j,k)]'-mean_wm_calc);
    mat_wm(j,k)=c5_wm_calc*exp(-0.5*tempo_wm'*inv_cov_wm_calc*tempo_wm);

    tempo_gm=([pd_mat_slice2(j,k),Tl_mat_slice2(j,k)]'-mean_gm_calc);
    mat_gm(j,k)=c5_gm_calc*exp(-0.5*tempo_gm'*inv_cov_gm_calc*tempo_gm);

```

```

    tempo_csf=([pd_mat_slice2(j,k),T1_mat_slice2(j,k)]'-mean_csf_calc);
    mat_csf(j,k)=c5_csf_calc*exp(-
0.5*tempo_csf'*inv_cov_csf_calc*tempo_csf);

end
end

clear j k

mat_slice2c=zeros(191,165);

for j=1:191
for k=1:165

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

mat_slice2c(j,k)=temp;

end
end

clear j k

h=figure('name','Segmented Slice2 - (by Using Calculated Parameters)',
...
'numbertitle','off',...
'menubar','none');

imagesc(mat_slice2c)

```

Matlab Code of Segmentation: calc_segmo slice3new.m

```

for j=1:196
for k=1:166
    tempo_wm=([pd_mat_slice3(j,k),T1_mat_slice3(j,k)]'-mean_wm_calc);
    mat_wm(j,k)=c5_wm_calc*exp(-0.5*tempo_wm'*inv_cov_wm_calc*tempo_wm);

    tempo_gm=([pd_mat_slice3(j,k),T1_mat_slice3(j,k)]'-mean_gm_calc);
    mat_gm(j,k)=c5_gm_calc*exp(-0.5*tempo_gm'*inv_cov_gm_calc*tempo_gm);

    tempo_csf=([pd_mat_slice3(j,k),T1_mat_slice3(j,k)]'-mean_csf_calc);
    mat_csf(j,k)=c5_csf_calc*exp(-
0.5*tempo_csf'*inv_cov_csf_calc*tempo_csf);

end
end

clear j k

```

```

mat_slice3c=zeros(196,166);

for j=1:196
for k=1:166

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

mat_slice3c(j,k)=temp;

end
end

clear j k

h=figure('name','Segmented Slice3 - (by Using Calculated Parameters)',
...
    'numbertitle','off',...
    'menubar','none');

imagesc(mat_slice3c)

```

Matlab Code of Segmentation: calc_segmoofslice4new.m

```

for j=1:184
for k=1:162
    tempo_wm=([pd_mat_slice4(j,k),T1_mat_slice4(j,k)]'-mean_wm_calc);
    mat_wm(j,k)=c5_wm_calc*exp(-0.5*tempo_wm'*inv_cov_wm_calc*tempo_wm);

    tempo_gm=([pd_mat_slice4(j,k),T1_mat_slice4(j,k)]'-mean_gm_calc);
    mat_gm(j,k)=c5_gm_calc*exp(-0.5*tempo_gm'*inv_cov_gm_calc*tempo_gm);

    tempo_csf=([pd_mat_slice4(j,k),T1_mat_slice4(j,k)]'-mean_csf_calc);
    mat_csf(j,k)=c5_csf_calc*exp(-
0.5*tempo_csf'*inv_cov_csf_calc*tempo_csf);

end
end

clear j k

mat_slice4c=zeros(184,162);

for j=1:184
for k=1:162

    if mat_wm(j,k)>mat_gm(j,k) & mat_wm(j,k)>mat_csf(j,k) temp=255; end
    if mat_csf(j,k)>mat_gm(j,k) & mat_csf(j,k)>mat_wm(j,k) temp=0; end
    if mat_gm(j,k)>mat_wm(j,k) & mat_gm(j,k)>mat_csf(j,k) temp=128; end

```

```

mat_slice4c(j,k)=temp;

end
end

clear j k

h=figure('name','Segmented Slice4 - (by Using Calculated Parameters)',
...
    'numbertitle','off',...
    'menubar','none');

imagesc(mat_slice4c)

```

Matlab Code of Confusion Matrix: conf_set_w.m

```

confusion_csf;
confusion_wm;
confusion_gm;

b1_csf=size(coord_csf_1);b1_csf=b1_csf(1);
b2_csf=size(coord_csf_2);b2_csf=b2_csf(1);
b3_csf=size(coord_csf_3);b3_csf=b3_csf(1);
b4_csf=size(coord_csf_4);b4_csf=b4_csf(1);
bt_csf=b1_csf+b2_csf+b3_csf+b4_csf;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b1_wm=size(coord_wm_1);b1_wm=b1_wm(1);
b2_wm=size(coord_wm_2);b2_wm=b2_wm(1);
b3_wm=size(coord_wm_3);b3_wm=b3_wm(1);
b4_wm=size(coord_wm_4);b4_wm=b4_wm(1);
bt_wm=b1_wm+b2_wm+b3_wm+b4_wm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b1_gm=size(coord_gm_1);b1_gm=b1_gm(1);
b2_gm=size(coord_gm_2);b2_gm=b2_gm(1);
b3_gm=size(coord_gm_3);b3_gm=b3_gm(1);
b4_gm=size(coord_gm_4);b4_gm=b4_gm(1);
bt_gm=b1_gm+b2_gm+b3_gm+b4_gm;

```

Matlab Code of Confusion Matrix: confusion_csf.m

```

imagesc(readmat1s(:,:,1)); colormap(gray);
zT1=readmat1s(:,:,1);
zPD=readmat1s(:,:,16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);

```

```

%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_csf_1=zeros(4*s,2);
w=zeros(4,2,s);
coord_csf_1=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_csf_1(i*4-3:4*i,:)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_csf_1(j,1);
    yaxis=coord_csf_1(j,2);
    wvalue_csf_1(j,:)= [zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imagesc(readmat2s(:, :, 1));colormap(gray);
zT1=readmat2s(:, :, 1);
zPD=readmat2s(:, :, 16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_csf_2=zeros(4*s,2);
w=zeros(4,2,s);
coord_csf_2=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;

```

```

end

for i=1:s coord_csf_2(i*4-3:4*i,:)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_csf_2(j,1);
    yaxis=coord_csf_2(j,2);
    wvalue_csf_2(j,:)= [zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imagesc(readmat3s(:, :, 1));colormap(gray);
zT1=readmat3s(:, :, 1);
zPD=readmat3s(:, :, 16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_csf_3=zeros(4*s,2);
w=zeros(4,2,s);
coord_csf_3=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_csf_3(i*4-3:4*i,:)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_csf_3(j,1);
    yaxis=coord_csf_3(j,2);
    wvalue_csf_3(j,:)= [zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imagesc(readmat4s(:, :, 1));colormap(gray);
zT1=readmat4s(:, :, 1);

```

```

zPD=readmat4s(:, :, 16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_csf_4=zeros(4*s,2);
w=zeros(4,2,s);
coord_csf_4=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_csf_4(i*4-3:4*i, :)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_csf_4(j,1);
    yaxis=coord_csf_4(j,2);
    wvalue_csf_4(j, :)= [zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

wvalue_csf=[wvalue_csf_1;wvalue_csf_2;wvalue_csf_3;wvalue_csf_4];

coord_csf=[coord_csf_1;coord_csf_2;coord_csf_3;coord_csf_4];

```

Matlab Code of Confusion Matrix: confusion_wm.m

```

imagesc(readmat1s(:, :, 1));colormap(gray);
zT1=readmat1s(:, :, 1);
zPD=readmat1s(:, :, 16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

```

```

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_wm_1=zeros(4*s,2);
w=zeros(4,2,s);
coord_wm_1=zeros(4*s,2);
v=ones(2,1);

```

```

for repeat=1:s;

```

```

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

```

```

    w(:, :, repeat)=window;
end

```

```

for i=1:s coord_wm_1(i*4-3:4*i,:)=w(:, :, i); end

```

```

for j=1:4*s
    xaxis=coord_wm_1(j,1);
    yaxis=coord_wm_1(j,2);
    wvalue_wm_1(j,:)= [zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

```

```

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

imagesc(readmat2s(:, :, 1));colormap(gray);
zT1=readmat2s(:, :, 1);
zPD=readmat2s(:, :, 16);

```

```

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

```

```

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_wm_2=zeros(4*s,2);
w=zeros(4,2,s);
coord_wm_2=zeros(4*s,2);
v=ones(2,1);

```

```

for repeat=1:s;

```

```

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

```

```

    w(:, :, repeat)=window;
end

```

```

for i=1:s coord_wm_2(i*4-3:4*i,:)=w(:, :, i); end

```

```

for j=1:4*s
    xaxis=coord_wm_2(j,1);
    yaxis=coord_wm_2(j,2);
    wvalue_wm_2(j,:)=[zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imagesc(readmat3s(:,:,1));colormap(gray);
zT1=readmat3s(:,:,1);
zPD=readmat3s(:,:,16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_wm_3=zeros(4*s,2);
w=zeros(4,2,s);
coord_wm_3=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_wm_3(i*4-3:4*i,:)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_wm_3(j,1);
    yaxis=coord_wm_3(j,2);
    wvalue_wm_3(j,:)=[zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imagesc(readmat4s(:,:,1));colormap(gray);
zT1=readmat4s(:,:,1);
zPD=readmat4s(:,:,16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);

```

```

sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_wm_4=zeros(4*s,2);
w=zeros(4,2,s);
coord_wm_4=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_wm_4(i*4-3:4*i, :)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_wm_4(j,1);
    yaxis=coord_wm_4(j,2);
    wvalue_wm_4(j, :)= [zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

wvalue_wm=[wvalue_wm_1;wvalue_wm_2;wvalue_wm_3;wvalue_wm_4];

coord_wm=[coord_wm_1;coord_wm_2;coord_wm_3;coord_wm_4];

```

Matlab Code of Confusion Matrix: confusion_gm.m

```

imagesc(readmatls(:, :, 1));colormap(gray);
zT1=readmatls(:, :, 1);
zPD=readmatls(:, :, 16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_gm_1=zeros(4*s,2);
w=zeros(4,2,s);
coord_gm_1=zeros(4*s,2);

```

```

v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_gm_1(i*4-3:4*i,:)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_gm_1(j,1);
    yaxis=coord_gm_1(j,2);
    wvalue_gm_1(j,:)= [zPD(xaxis,yaxis) zTl(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zTl j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imagesc(readmat2s(:, :, 1));colormap(gray);
zTl=readmat2s(:, :, 1);
zPD=readmat2s(:, :, 16);

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_gm_2=zeros(4*s,2);
w=zeros(4,2,s);
coord_gm_2=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_gm_2(i*4-3:4*i,:)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_gm_2(j,1);

```

```

    yaxis=coord_gm_2(j,2);
    wvalue_gm_2(j,:)=[zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

```

```

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

imagesc(readmat3s(:,:,1));colormap(gray);
zT1=readmat3s(:,:,1);
zPD=readmat3s(:,:,16);

```

```

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen
s=sayz(1);

```

```

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_gm_3=zeros(4*s,2);
w=zeros(4,2,s);
coord_gm_3=zeros(4*s,2);
v=ones(2,1);

```

```

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

```

```

    w(:, :, repeat)=window;
end

```

```

for i=1:s coord_gm_3(i*4-3:4*i,:)=w(:, :, i); end

```

```

for j=1:4*s
    xaxis=coord_gm_3(j,1);
    yaxis=coord_gm_3(j,2);
    wvalue_gm_3(j,:)=[zPD(xaxis,yaxis) zT1(xaxis,yaxis)];
end

```

```

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zT1 j;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

imagesc(readmat4s(:,:,1));colormap(gray);
zT1=readmat4s(:,:,1);
zPD=readmat4s(:,:,16);

```

```

oku=ginput;
%top left coordinate of the window is determined
oku=round(oku);
sayz=size(oku);
%finds how many windows are chosen

```

```

s=sayz(1);

window=zeros(4,2); %allocates 4 by 2 matrix for window coordinates
wvalue_gm_4=zeros(4*s,2);
w=zeros(4,2,s);
coord_gm_4=zeros(4*s,2);
v=ones(2,1);

for repeat=1:s;

    window(1:2,1)=oku(repeat,2)*v;
    window(3:4,1)=(oku(repeat,2)+1)*v;

    window(1:2:3,2)=oku(repeat,1)*v;
    window(2:2:4,2)=(oku(repeat,1)+1)*v;

    w(:, :, repeat)=window;
end

for i=1:s coord_gm_4(i*4-3:4*i,:)=w(:, :, i); end

for j=1:4*s
    xaxis=coord_gm_4(j,1);
    yaxis=coord_gm_4(j,2);
    wvalue_gm_4(j,:)=[zPD(xaxis,yaxis) zTl(xaxis,yaxis)];
end

clear repeat s sayz v w window oku i;
clear xaxis yaxis zPD zTl j;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

wvalue_gm=[wvalue_gm_1;wvalue_gm_2;wvalue_gm_3;wvalue_gm_4];

coord_gm=[coord_gm_1;coord_gm_2;coord_gm_3;coord_gm_4];

```

Matlab Code of Confusion Matrix: confusion_mat1.m

```

conf_csf_calc_slicel=zeros(1,b1_csf);
for tt=1:b1_csf
    xq=coord_csf_1(tt,1);
    yq=coord_csf_1(tt,2);
    conf_csf_calc_slicel(tt)=mat_slicelc(xq,yq);
end

conf_wm_calc_slicel=zeros(1,b1_wm);
for tt=1:b1_wm
    xq=coord_wm_1(tt,1);
    yq=coord_wm_1(tt,2);
    conf_wm_calc_slicel(tt)=mat_slicelc(xq,yq);
end

conf_gm_calc_slicel=zeros(1,b1_gm);
for tt=1:b1_gm
    xq=coord_gm_1(tt,1);

```

```

    yq=coord_gm_1(tt,2);
    conf_gm_calc_slicel(tt)=mat_slicelc(xq,yq);
end

```

```

%% % % % % % % % % % % % % % % % %

```

```

conf_csf_weight_slicel=zeros(1,b1_csf);
for tt=1:b1_csf
    xq=coord_csf_1(tt,1);
    yq=coord_csf_1(tt,2);
    conf_csf_weight_slicel(tt)=mat_slicelw(xq,yq);
end

```

```

conf_wm_weight_slicel=zeros(1,b1_wm);
for tt=1:b1_wm
    xq=coord_wm_1(tt,1);
    yq=coord_wm_1(tt,2);
    conf_wm_weight_slicel(tt)=mat_slicelw(xq,yq);
end

```

```

conf_gm_weight_slicel=zeros(1,b1_gm);
for tt=1:b1_gm
    xq=coord_gm_1(tt,1);
    yq=coord_gm_1(tt,2);
    conf_gm_weight_slicel(tt)=mat_slicelw(xq,yq);
end

```

```

clear tt xq yq;
csf_csf_calc=0; csf_gm_calc=0; csf_wm_calc=0;
gm_csf_calc=0; gm_gm_calc=0; gm_wm_calc=0;
wm_csf_calc=0; wm_gm_calc=0; wm_wm_calc=0;
csf_csf_weight=0; csf_gm_weight=0; csf_wm_weight=0;
gm_csf_weight=0; gm_gm_weight=0; gm_wm_weight=0;
wm_csf_weight=0; wm_gm_weight=0; wm_wm_weight=0;

```

```

for tt=1:b1_csf
    if conf_csf_calc_slicel(tt)==0 csf_csf_calc=csf_csf_calc+1; end
    if conf_csf_calc_slicel(tt)==128 csf_gm_calc=csf_gm_calc+1; end
    if conf_csf_calc_slicel(tt)==255 csf_wm_calc=csf_wm_calc+1; end
    if conf_csf_weight_slicel(tt)==0 csf_csf_weight=csf_csf_weight+1; end
    if conf_csf_weight_slicel(tt)==128 csf_gm_weight=csf_gm_weight+1; end
    if conf_csf_weight_slicel(tt)==255 csf_wm_weight=csf_wm_weight+1; end

```

```

end

```

```

for tt=1:b1_gm
    if conf_gm_calc_slicel(tt)==0, gm_csf_calc=gm_csf_calc+1; end
    if conf_gm_calc_slicel(tt)==128, gm_gm_calc=gm_gm_calc+1; end
    if conf_gm_calc_slicel(tt)==255, gm_wm_calc=gm_wm_calc+1; end
    if conf_gm_weight_slicel(tt)==0, gm_csf_weight=gm_csf_weight+1; end
    if conf_gm_weight_slicel(tt)==128, gm_gm_weight=gm_gm_weight+1; end
    if conf_gm_weight_slicel(tt)==255, gm_wm_weight=gm_wm_weight+1; end

```

```

end

```

```

for tt=1:b1_wm
    if conf_wm_calc_slicel(tt)==0, wm_csf_calc=wm_csf_calc+1; end
    if conf_wm_calc_slicel(tt)==128, wm_gm_calc=wm_gm_calc+1; end
    if conf_wm_calc_slicel(tt)==255, wm_wm_calc=wm_wm_calc+1; end

```

```

    if conf_wm_weight_slicel(tt)==0, wm_csf_weight=wm_csf_weight+1; end
    if conf_wm_weight_slicel(tt)==128, wm_gm_weight=wm_gm_weight+1; end
    if conf_wm_weight_slicel(tt)==255, wm_wm_weight=wm_wm_weight+1; end

end

clear tt;

conf_mat_calc_slicel=zeros(3,3);
conf_mat_weight_slicel=zeros(3,3);

conf_mat_calc_slicel(1,1)=csf_csf_calc;
conf_mat_calc_slicel(1,2)=csf_gm_calc;
conf_mat_calc_slicel(1,3)=csf_wm_calc;
conf_mat_calc_slicel(2,1)=gm_csf_calc;
conf_mat_calc_slicel(2,2)=gm_gm_calc;
conf_mat_calc_slicel(2,3)=gm_wm_calc;
conf_mat_calc_slicel(3,1)=wm_csf_calc;
conf_mat_calc_slicel(3,2)=wm_gm_calc;
conf_mat_calc_slicel(3,3)=wm_wm_calc;

conf_mat_weight_slicel(1,1)=csf_csf_weight;
conf_mat_weight_slicel(1,2)=csf_gm_weight;
conf_mat_weight_slicel(1,3)=csf_wm_weight;
conf_mat_weight_slicel(2,1)=gm_csf_weight;
conf_mat_weight_slicel(2,2)=gm_gm_weight;
conf_mat_weight_slicel(2,3)=gm_wm_weight;
conf_mat_weight_slicel(3,1)=wm_csf_weight;
conf_mat_weight_slicel(3,2)=wm_gm_weight;
conf_mat_weight_slicel(3,3)=wm_wm_weight;

```

Matlab Code of Confusion Matrix: confusion_mat2.m

```

conf_csf_calc_slice2=zeros(1,b2_csf);
for tt=1:b2_csf
    xq=coord_csf_2(tt,1);
    yq=coord_csf_2(tt,2);
    conf_csf_calc_slice2(tt)=mat_slice2c(xq,yq);
end

conf_wm_calc_slice2=zeros(1,b2_wm);
for tt=1:b2_wm
    xq=coord_wm_2(tt,1);
    yq=coord_wm_2(tt,2);
    conf_wm_calc_slice2(tt)=mat_slice2c(xq,yq);
end

conf_gm_calc_slice2=zeros(1,b2_gm);
for tt=1:b2_gm
    xq=coord_gm_2(tt,1);
    yq=coord_gm_2(tt,2);
    conf_gm_calc_slice2(tt)=mat_slice2c(xq,yq);
end

```

```

%% %% %% %% %% %% %% %% %% %%

```

```

conf_csf_weight_slice2=zeros(1,b2_csf);
for tt=1:b2_csf
    xq=coord_csf_2(tt,1);
    yq=coord_csf_2(tt,2);
    conf_csf_weight_slice2(tt)=mat_slice2w(xq,yq);
end

conf_wm_weight_slice2=zeros(1,b2_wm);
for tt=1:b2_wm
    xq=coord_wm_2(tt,1);
    yq=coord_wm_2(tt,2);
    conf_wm_weight_slice2(tt)=mat_slice2w(xq,yq);
end

conf_gm_weight_slice2=zeros(1,b2_gm);
for tt=1:b2_gm
    xq=coord_gm_2(tt,1);
    yq=coord_gm_2(tt,2);
    conf_gm_weight_slice2(tt)=mat_slice2w(xq,yq);
end

clear tt xq yq;
csf_csf_calc=0; csf_gm_calc=0; csf_wm_calc=0;
gm_csf_calc=0; gm_gm_calc=0; gm_wm_calc=0;
wm_csf_calc=0; wm_gm_calc=0; wm_wm_calc=0;
csf_csf_weight=0; csf_gm_weight=0; csf_wm_weight=0;
gm_csf_weight=0; gm_gm_weight=0; gm_wm_weight=0;
wm_csf_weight=0; wm_gm_weight=0; wm_wm_weight=0;

for tt=1:b2_csf
    if conf_csf_calc_slice2(tt)==0 csf_csf_calc=csf_csf_calc+1; end
    if conf_csf_calc_slice2(tt)==128 csf_gm_calc=csf_gm_calc+1; end
    if conf_csf_calc_slice2(tt)==255 csf_wm_calc=csf_wm_calc+1; end
    if conf_csf_weight_slice2(tt)==0 csf_csf_weight=csf_csf_weight+1; end
    if conf_csf_weight_slice2(tt)==128 csf_gm_weight=csf_gm_weight+1; end
    if conf_csf_weight_slice2(tt)==255 csf_wm_weight=csf_wm_weight+1; end
end

for tt=1:b2_gm
    if conf_gm_calc_slice2(tt)==0, gm_csf_calc=gm_csf_calc+1; end
    if conf_gm_calc_slice2(tt)==128, gm_gm_calc=gm_gm_calc+1; end
    if conf_gm_calc_slice2(tt)==255, gm_wm_calc=gm_wm_calc+1; end
    if conf_gm_weight_slice2(tt)==0, gm_csf_weight=gm_csf_weight+1; end
    if conf_gm_weight_slice2(tt)==128, gm_gm_weight=gm_gm_weight+1; end
    if conf_gm_weight_slice2(tt)==255, gm_wm_weight=gm_wm_weight+1; end
end

for tt=1:b2_wm
    if conf_wm_calc_slice2(tt)==0, wm_csf_calc=wm_csf_calc+1; end
    if conf_wm_calc_slice2(tt)==128, wm_gm_calc=wm_gm_calc+1; end
    if conf_wm_calc_slice2(tt)==255, wm_wm_calc=wm_wm_calc+1; end
    if conf_wm_weight_slice2(tt)==0, wm_csf_weight=wm_csf_weight+1; end
    if conf_wm_weight_slice2(tt)==128, wm_gm_weight=wm_gm_weight+1; end
    if conf_wm_weight_slice2(tt)==255, wm_wm_weight=wm_wm_weight+1; end
end

```

```
clear tt;

conf_mat_calc_slice2=zeros(3,3);
conf_mat_weight_slice2=zeros(3,3);

conf_mat_calc_slice2(1,1)=csf_csf_calc;
conf_mat_calc_slice2(1,2)=csf_gm_calc;
conf_mat_calc_slice2(1,3)=csf_wm_calc;
conf_mat_calc_slice2(2,1)=gm_csf_calc;
conf_mat_calc_slice2(2,2)=gm_gm_calc;
conf_mat_calc_slice2(2,3)=gm_wm_calc;
conf_mat_calc_slice2(3,1)=wm_csf_calc;
conf_mat_calc_slice2(3,2)=wm_gm_calc;
conf_mat_calc_slice2(3,3)=wm_wm_calc;

conf_mat_weight_slice2(1,1)=csf_csf_weight;
conf_mat_weight_slice2(1,2)=csf_gm_weight;
conf_mat_weight_slice2(1,3)=csf_wm_weight;
conf_mat_weight_slice2(2,1)=gm_csf_weight;
conf_mat_weight_slice2(2,2)=gm_gm_weight;
conf_mat_weight_slice2(2,3)=gm_wm_weight;
conf_mat_weight_slice2(3,1)=wm_csf_weight;
conf_mat_weight_slice2(3,2)=wm_gm_weight;
conf_mat_weight_slice2(3,3)=wm_wm_weight;
```

Matlab Code of Confusion Matrix: confusion_mat3.m

```

conf_csf_calc_slice3=zeros(1,b3_csf);
for tt=1:b3_csf
    xq=coord_csf_3(tt,1);
    yq=coord_csf_3(tt,2);
    conf_csf_calc_slice3(tt)=mat_slice3c(xq,yq);
end

conf_wm_calc_slice3=zeros(1,b3_wm);
for tt=1:b3_wm
    xq=coord_wm_3(tt,1);
    yq=coord_wm_3(tt,2);
    conf_wm_calc_slice3(tt)=mat_slice3c(xq,yq);
end

conf_gm_calc_slice3=zeros(1,b3_gm);
for tt=1:b3_gm
    xq=coord_gm_3(tt,1);
    yq=coord_gm_3(tt,2);
    conf_gm_calc_slice3(tt)=mat_slice3c(xq,yq);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

conf_csf_weight_slice3=zeros(1,b3_csf);
for tt=1:b3_csf
    xq=coord_csf_3(tt,1);
    yq=coord_csf_3(tt,2);
    conf_csf_weight_slice3(tt)=mat_slice3w(xq,yq);
end

```

```

conf_wm_weight_slice3=zeros(1,b3_wm);
for tt=1:b3_wm
    xq=coord_wm_3(tt,1);
    yq=coord_wm_3(tt,2);
    conf_wm_weight_slice3(tt)=mat_slice3w(xq,yq);
end

conf_gm_weight_slice3=zeros(1,b3_gm);
for tt=1:b3_gm
    xq=coord_gm_3(tt,1);
    yq=coord_gm_3(tt,2);
    conf_gm_weight_slice3(tt)=mat_slice3w(xq,yq);
end

clear tt xq yq;
csf_csf_calc=0; csf_gm_calc=0; csf_wm_calc=0;
gm_csf_calc=0; gm_gm_calc=0; gm_wm_calc=0;
wm_csf_calc=0; wm_gm_calc=0; wm_wm_calc=0;
csf_csf_weight=0; csf_gm_weight=0; csf_wm_weight=0;
gm_csf_weight=0; gm_gm_weight=0; gm_wm_weight=0;
wm_csf_weight=0; wm_gm_weight=0; wm_wm_weight=0;

for tt=1:b3_csf
    if conf_csf_calc_slice3(tt)==0 csf_csf_calc=csf_csf_calc+1; end
    if conf_csf_calc_slice3(tt)==128 csf_gm_calc=csf_gm_calc+1; end
    if conf_csf_calc_slice3(tt)==255 csf_wm_calc=csf_wm_calc+1; end
    if conf_csf_weight_slice3(tt)==0 csf_csf_weight=csf_csf_weight+1; end
    if conf_csf_weight_slice3(tt)==128 csf_gm_weight=csf_gm_weight+1; end
    if conf_csf_weight_slice3(tt)==255 csf_wm_weight=csf_wm_weight+1; end
end

for tt=1:b3_gm
    if conf_gm_calc_slice3(tt)==0, gm_csf_calc=gm_csf_calc+1; end
    if conf_gm_calc_slice3(tt)==128, gm_gm_calc=gm_gm_calc+1; end
    if conf_gm_calc_slice3(tt)==255, gm_wm_calc=gm_wm_calc+1; end
    if conf_gm_weight_slice3(tt)==0, gm_csf_weight=gm_csf_weight+1; end
    if conf_gm_weight_slice3(tt)==128, gm_gm_weight=gm_gm_weight+1; end
    if conf_gm_weight_slice3(tt)==255, gm_wm_weight=gm_wm_weight+1; end
end

for tt=1:b3_wm
    if conf_wm_calc_slice3(tt)==0, wm_csf_calc=wm_csf_calc+1; end
    if conf_wm_calc_slice3(tt)==128, wm_gm_calc=wm_gm_calc+1; end
    if conf_wm_calc_slice3(tt)==255, wm_wm_calc=wm_wm_calc+1; end
    if conf_wm_weight_slice3(tt)==0, wm_csf_weight=wm_csf_weight+1; end
    if conf_wm_weight_slice3(tt)==128, wm_gm_weight=wm_gm_weight+1; end
    if conf_wm_weight_slice3(tt)==255, wm_wm_weight=wm_wm_weight+1; end
end

clear tt;

conf_mat_calc_slice3=zeros(3,3);
conf_mat_weight_slice3=zeros(3,3);

conf_mat_calc_slice3(1,1)=csf_csf_calc;
conf_mat_calc_slice3(1,2)=csf_gm_calc;

```

```

conf_mat_calc_slice3(1,3)=csf_wm_calc;
conf_mat_calc_slice3(2,1)=gm_csf_calc;
conf_mat_calc_slice3(2,2)=gm_gm_calc;
conf_mat_calc_slice3(2,3)=gm_wm_calc;
conf_mat_calc_slice3(3,1)=wm_csf_calc;
conf_mat_calc_slice3(3,2)=wm_gm_calc;
conf_mat_calc_slice3(3,3)=wm_wm_calc;

conf_mat_weight_slice3(1,1)=csf_csf_weight;
conf_mat_weight_slice3(1,2)=csf_gm_weight;
conf_mat_weight_slice3(1,3)=csf_wm_weight;
conf_mat_weight_slice3(2,1)=gm_csf_weight;
conf_mat_weight_slice3(2,2)=gm_gm_weight;
conf_mat_weight_slice3(2,3)=gm_wm_weight;
conf_mat_weight_slice3(3,1)=wm_csf_weight;
conf_mat_weight_slice3(3,2)=wm_gm_weight;
conf_mat_weight_slice3(3,3)=wm_wm_weight;

```

Matlab Code of Confusion Matrix: confusion_mat4.m

```

conf_csf_calc_slice4=zeros(1,b4_csf);
for tt=1:b4_csf
    xq=coord_csf_4(tt,1);
    yq=coord_csf_4(tt,2);
    conf_csf_calc_slice4(tt)=mat_slice4c(xq,yq);
end

conf_wm_calc_slice4=zeros(1,b4_wm);
for tt=1:b4_wm
    xq=coord_wm_4(tt,1);
    yq=coord_wm_4(tt,2);
    conf_wm_calc_slice4(tt)=mat_slice4c(xq,yq);
end

conf_gm_calc_slice4=zeros(1,b4_gm);
for tt=1:b4_gm
    xq=coord_gm_4(tt,1);
    yq=coord_gm_4(tt,2);
    conf_gm_calc_slice4(tt)=mat_slice4c(xq,yq);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

conf_csf_weight_slice4=zeros(1,b4_csf);
for tt=1:b4_csf
    xq=coord_csf_4(tt,1);
    yq=coord_csf_4(tt,2);
    conf_csf_weight_slice4(tt)=mat_slice4w(xq,yq);
end

conf_wm_weight_slice4=zeros(1,b4_wm);
for tt=1:b4_wm
    xq=coord_wm_4(tt,1);
    yq=coord_wm_4(tt,2);
    conf_wm_weight_slice4(tt)=mat_slice4w(xq,yq);
end

```

```

conf_gm_weight_slice4=zeros(1,b4_gm);
for tt=1:b4_gm
    xq=coord_gm_4(tt,1);
    yq=coord_gm_4(tt,2);
    conf_gm_weight_slice4(tt)=mat_slice4w(xq,yq);
end

clear tt xq yq;
csf_csf_calc=0; csf_gm_calc=0; csf_wm_calc=0;
gm_csf_calc=0; gm_gm_calc=0; gm_wm_calc=0;
wm_csf_calc=0; wm_gm_calc=0; wm_wm_calc=0;
csf_csf_weight=0; csf_gm_weight=0; csf_wm_weight=0;
gm_csf_weight=0; gm_gm_weight=0; gm_wm_weight=0;
wm_csf_weight=0; wm_gm_weight=0; wm_wm_weight=0;

for tt=1:b4_csf
    if conf_csf_calc_slice4(tt)==0 csf_csf_calc=csf_csf_calc+1; end
    if conf_csf_calc_slice4(tt)==128 csf_gm_calc=csf_gm_calc+1; end
    if conf_csf_calc_slice4(tt)==255 csf_wm_calc=csf_wm_calc+1; end
    if conf_csf_weight_slice4(tt)==0 csf_csf_weight=csf_csf_weight+1; end
    if conf_csf_weight_slice4(tt)==128 csf_gm_weight=csf_gm_weight+1; end
    if conf_csf_weight_slice4(tt)==255 csf_wm_weight=csf_wm_weight+1; end
end

for tt=1:b4_gm
    if conf_gm_calc_slice4(tt)==0, gm_csf_calc=gm_csf_calc+1; end
    if conf_gm_calc_slice4(tt)==128, gm_gm_calc=gm_gm_calc+1; end
    if conf_gm_calc_slice4(tt)==255, gm_wm_calc=gm_wm_calc+1; end
    if conf_gm_weight_slice4(tt)==0, gm_csf_weight=gm_csf_weight+1; end
    if conf_gm_weight_slice4(tt)==128, gm_gm_weight=gm_gm_weight+1; end
    if conf_gm_weight_slice4(tt)==255, gm_wm_weight=gm_wm_weight+1; end
end

for tt=1:b4_wm
    if conf_wm_calc_slice4(tt)==0, wm_csf_calc=wm_csf_calc+1; end
    if conf_wm_calc_slice4(tt)==128, wm_gm_calc=wm_gm_calc+1; end
    if conf_wm_calc_slice4(tt)==255, wm_wm_calc=wm_wm_calc+1; end
    if conf_wm_weight_slice4(tt)==0, wm_csf_weight=wm_csf_weight+1; end
    if conf_wm_weight_slice4(tt)==128, wm_gm_weight=wm_gm_weight+1; end
    if conf_wm_weight_slice4(tt)==255, wm_wm_weight=wm_wm_weight+1; end
end

clear tt;

conf_mat_calc_slice4=zeros(3,3);
conf_mat_weight_slice4=zeros(3,3);

conf_mat_calc_slice4(1,1)=csf_csf_calc;
conf_mat_calc_slice4(1,2)=csf_gm_calc;
conf_mat_calc_slice4(1,3)=csf_wm_calc;
conf_mat_calc_slice4(2,1)=gm_csf_calc;
conf_mat_calc_slice4(2,2)=gm_gm_calc;
conf_mat_calc_slice4(2,3)=gm_wm_calc;
conf_mat_calc_slice4(3,1)=wm_csf_calc;
conf_mat_calc_slice4(3,2)=wm_gm_calc;
conf_mat_calc_slice4(3,3)=wm_wm_calc;

```

```

conf_mat_weight_slice4(1,1)=csf_csf_weight;
conf_mat_weight_slice4(1,2)=csf_gm_weight;
conf_mat_weight_slice4(1,3)=csf_wm_weight;
conf_mat_weight_slice4(2,1)=gm_csf_weight;
conf_mat_weight_slice4(2,2)=gm_gm_weight;
conf_mat_weight_slice4(2,3)=gm_wm_weight;
conf_mat_weight_slice4(3,1)=wm_csf_weight;
conf_mat_weight_slice4(3,2)=wm_gm_weight;
conf_mat_weight_slice4(3,3)=wm_wm_weight;

```

Matlab Code of Multivariate Gaussian Density Curves: bellcurve_weight.m

```

axis1_csf_weight=[1:10:4991];
axis2_csf_weight=[1:10:4991];
total_csf_weight=zeros(500,500);

for j=1:500
for k=1:500
    tempo=([axis1_csf_weight(j),axis2_csf_weight(k)]'-mean_csf_weight);
    total_csf_weight(k,j)=c5_csf_weight*exp(-
0.5*tempo'*inv_cov_csf_weight*tempo);
end
end

axis1_wm_weight=[1:10:4991];
axis2_wm_weight=[1:10:4991];
total_wm_weight=zeros(500,500);

for j=1:500
for k=1:500
    tempo=([axis1_wm_weight(j),axis2_wm_weight(k)]'-mean_wm_weight);
    total_wm_weight(k,j)=c5_wm_weight*exp(-
0.5*tempo'*inv_cov_wm_weight*tempo);
end
end

axis1_gm_weight=[1:10:4991];
axis2_gm_weight=[1:10:4991];
total_gm_weight=zeros(500,500);

for j=1:500
for k=1:500
    tempo=([axis1_gm_weight(j),axis2_gm_weight(k)]'-mean_gm_weight);
    total_gm_weight(k,j)=c5_gm_weight*exp(-
0.5*tempo'*inv_cov_gm_weight*tempo);
end
end

h=figure('numbertitle','off',...
    'menubar','none');

mesh(axis1_csf_weight,axis2_csf_weight,total_csf_weight);view(2);hold on;
mesh(axis1_wm_weight,axis2_wm_weight,total_wm_weight);view(2);hold on;
mesh(axis1_gm_weight,axis2_gm_weight,total_gm_weight);view(2)

```

```
title('Multivariate Gaussian Density Curves of Tissues in Weighted
Images')
```

Matlab Code of Multivariate Gaussian Density Curves: bellcurve_calc.m

```
axis1_csf_calc=[1:10:4991];
axis2_csf_calc=[1:10:4991];
total_csf_calc=zeros(500,500);

for j=1:500
for k=1:500
    tempo=([axis1_csf_calc(j),axis2_csf_calc(k)]'-mean_csf_calc);
    total_csf_calc(k,j)=c5_csf_calc*exp(-
0.5*tempo'*inv_cov_csf_calc*tempo);
end
end

axis1_wm_calc=[1:10:4991];
axis2_wm_calc=[1:10:4991];
total_wm_calc=zeros(500,500);

for j=1:500
for k=1:500
    tempo=([axis1_wm_calc(j),axis2_wm_calc(k)]'-mean_wm_calc);
    total_wm_calc(k,j)=c5_wm_calc*exp(-0.5*tempo'*inv_cov_wm_calc*tempo);
end
end

axis1_gm_calc=[1:10:4991];
axis2_gm_calc=[1:10:4991];
total_gm_calc=zeros(500,500);

for j=1:500
for k=1:500
    tempo=([axis1_gm_calc(j),axis2_gm_calc(k)]'-mean_gm_calc);
    total_gm_calc(k,j)=c5_gm_calc*exp(-0.5*tempo'*inv_cov_gm_calc*tempo);
end
end

h=figure('numbertitle','off',...
    'menubar','none');

mesh(axis1_csf_calc,axis2_csf_calc,40*total_csf_calc);view(2);hold on;
mesh(axis1_wm_calc,axis2_wm_calc,total_wm_calc);view(2);hold on;
mesh(axis1_gm_calc,axis2_gm_calc,4*total_gm_calc);view(2)

title('Multivariate Gaussian Density Curves of Tissues in Calculated
Images')
xlabel('Calculated Proton Density')
ylabel('Calculated T1')
```

Matlab Code of Training Datasets: trainingset.m

```

load c:\old\bora\readslc\readmat1s;
load c:\old\bora\readslc\readmat2s;
load c:\old\bora\readslc\readmat3s;
load c:\old\bora\readslc\readmat4s;
load c:\old\bora\prm_images\prm_mat.mat;

tlw_slice1=readmat1s(:,:,1);
tlw_slice2=readmat2s(:,:,1);
tlw_slice3=readmat3s(:,:,1);
tlw_slice4=readmat4s(:,:,1);
pdw_slice1=readmat1s(:,:,16);
pdw_slice2=readmat2s(:,:,16);
pdw_slice3=readmat3s(:,:,16);
pdw_slice4=readmat4s(:,:,16);

training2csf

training2wm

training2gm

s1_csf=size(w_mat_csf_1);s1_csf=s1_csf(1);
s2_csf=size(w_mat_csf_2);s2_csf=s2_csf(1);
s3_csf=size(w_mat_csf_3);s3_csf=s3_csf(1);
s4_csf=size(w_mat_csf_4);s4_csf=s4_csf(1);
st_csf=s1_csf+s2_csf+s3_csf+s4_csf;

est_csf_weight=zeros(st_csf,2);

for i=1:s1_csf;
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);
    est_csf_weight(i,:)=[pdw_slice1(row,col),tlw_slice1(row,col)];
end
for i=(s1_csf+1):(s1_csf+s2_csf);
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);
    est_csf_weight(i,:)=[pdw_slice2(row,col),tlw_slice2(row,col)];
end
for i=(s1_csf+s2_csf+1):(s1_csf+s2_csf+s3_csf);
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);
    est_csf_weight(i,:)=[pdw_slice3(row,col),tlw_slice3(row,col)];
end
for i=(s1_csf+s2_csf+s3_csf+1):(s1_csf+s2_csf+s3_csf+s4_csf);
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);
    est_csf_weight(i,:)=[pdw_slice4(row,col),tlw_slice4(row,col)];
end
est_csf_weight=PD_T1_csf;
%%
for k=1:st_csf
    if est_csf_weight(k,1)<5000 & est_csf_weight(k,2)<5000,
e_csf_weight(k,:)=est_csf_weight(k,:); end;
end

clear est_csf_weight;

```

```

[i,j,s] = find(e_csf_weight);

h=(size(s)/2);

est_csf_weight=[s(1:h) s((h+1):(h+h))];
%%

cov_csf=cov(est_csf_weight);
mean_csf_weight=mean(est_csf_weight);

mean_csf_weight=mean_csf_weight';

det_cov_csf=det(cov_csf);

inv_cov_csf_weight=inv(cov_csf);

c1=2*pi;
c2=c1^2;
c3=c2*det_cov_csf;
c4=sqrt(c3);
c5_csf_weight=1/c4;

clear i j k s h e_csf_weight c1 c2 c3 c4;
clear cov_csf det_cov_csf;
clear col row;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s1_wm=size(w_mat_wm_1);s1_wm=s1_wm(1);
s2_wm=size(w_mat_wm_2);s2_wm=s2_wm(1);
s3_wm=size(w_mat_wm_3);s3_wm=s3_wm(1);
s4_wm=size(w_mat_wm_4);s4_wm=s4_wm(1);
st_wm=s1_wm+s2_wm+s3_wm+s4_wm;

est_wm_weight=zeros(st_wm,2);

for i=1:s1_wm;
    row=w_mat_wm(i,1);
    col=w_mat_wm(i,2);
    est_wm_weight(i,:)=[pdw_slice1(row,col),tlw_slice1(row,col)];
end
for i=(s1_wm+1):(s1_wm+s2_wm);
    row=w_mat_wm(i,1);
    col=w_mat_wm(i,2);
    est_wm_weight(i,:)=[pdw_slice2(row,col),tlw_slice2(row,col)];
end
for i=(s1_wm+s2_wm+1):(s1_wm+s2_wm+s3_wm);
    row=w_mat_wm(i,1);
    col=w_mat_wm(i,2);
    est_wm_weight(i,:)=[pdw_slice3(row,col),tlw_slice3(row,col)];
end
for i=(s1_wm+s2_wm+s3_wm+1):(s1_wm+s2_wm+s3_wm+s4_wm);
    row=w_mat_wm(i,1);
    col=w_mat_wm(i,2);
    est_wm_weight(i,:)=[pdw_slice4(row,col),tlw_slice4(row,col)];
end

est_wm_weight=PD_T1_wm;
%%
for k=1:st_wm

```

```

    if est_wm_weight(k,1)<5000 & est_wm_weight(k,2)<5000,
    e_wm_weight(k,:)=est_wm_weight(k,:); end;
end

clear est_wm_weight;

[i,j,s] = find(e_wm_weight);

h=(size(s)/2);

est_wm_weight=[s(1:h) s((h+1):(h+h))];
%%

cov_wm=cov(est_wm_weight);
mean_wm_weight=mean(est_wm_weight);

mean_wm_weight=mean_wm_weight';

det_cov_wm=det(cov_wm);

inv_cov_wm_weight=inv(cov_wm);

c1=2*pi;
c2=c1^2;
c3=c2*det_cov_wm;
c4=sqrt(c3);
c5_wm_weight=1/c4;

clear i j k s h e_wm_weight c1 c2 c3 c4;
clear cov_wm det_cov_wm;
clear col row;

%%
s1_gm=size(w_mat_gm_1);s1_gm=s1_gm(1);
s2_gm=size(w_mat_gm_2);s2_gm=s2_gm(1);
s3_gm=size(w_mat_gm_3);s3_gm=s3_gm(1);
s4_gm=size(w_mat_gm_4);s4_gm=s4_gm(1);
st_gm=s1_gm+s2_gm+s3_gm+s4_gm;

est_gm_weight=zeros(st_gm,2);

for i=1:s1_gm;
    row=w_mat_gm(i,1);
    col=w_mat_gm(i,2);
    est_gm_weight(i,:)=[pdw_slicel(row,col),tlw_slicel(row,col)];
end
for i=(s1_gm+1):(s1_gm+s2_gm);
    row=w_mat_gm(i,1);
    col=w_mat_gm(i,2);
    est_gm_weight(i,:)=[pdw_slice2(row,col),tlw_slice2(row,col)];
end
for i=(s1_gm+s2_gm+1):(s1_gm+s2_gm+s3_gm);
    row=w_mat_gm(i,1);
    col=w_mat_gm(i,2);
    est_gm_weight(i,:)=[pdw_slice3(row,col),tlw_slice3(row,col)];
end
for i=(s1_gm+s2_gm+s3_gm+1):(s1_gm+s2_gm+s3_gm+s4_gm);
    row=w_mat_gm(i,1);
    col=w_mat_gm(i,2);

```

```

    est_gm_weight(i,:)=[pdw_slice4(row,col),tlw_slice4(row,col)];
end

est_gm_weight=PD_Tl_gm;
%%%
for k=1:st_gm
    if est_gm_weight(k,1)<5000 & est_gm_weight(k,2)<5000,
e_gm_weight(k,:)=est_gm_weight(k,:); end;
end

clear est_gm_weight;

[i,j,s] = find(e_gm_weight);

h=(size(s)/2);

est_gm_weight=[s(1:h) s((h+1):(h+h))];
%%%

cov_gm=cov(est_gm_weight);
mean_gm_weight=mean(est_gm_weight);

mean_gm_weight=mean_gm_weight';

det_cov_gm=det(cov_gm);

inv_cov_gm_weight=inv(cov_gm);

c1=2*pi;
c2=c1^2;
c3=c2*det_cov_gm;
c4=sqrt(c3);
c5_gm_weight=1/c4;

clear i j k s h e_gm_weight c1 c2 c3 c4;
clear cov_gm det_cov_gm;
clear col row;

%%%
%%%Maximum Likelihood of Calculated Parameters

load c:\old\bora\prm_images\prm_mat.mat;

est_csf_calc=zeros(st_csf,2);

for i=1:s1_csf;
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);
    est_csf_calc(i,:)=[pd_mat_slicel(row,col),Tl_mat_slicel(row,col)];
end
for i=(s1_csf+1):(s1_csf+s2_csf);
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);
    est_csf_calc(i,:)=[pd_mat_slice2(row,col),Tl_mat_slice2(row,col)];
end
for i=(s1_csf+s2_csf+1):(s1_csf+s2_csf+s3_csf);
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);

```

```

    est_csf_calc(i,:)=[pd_mat_slice3(row,col),T1_mat_slice3(row,col)];
end
for i=(s1_csf+s2_csf+s3_csf+1):(s1_csf+s2_csf+s3_csf+s4_csf);
    row=w_mat_csf(i,1);
    col=w_mat_csf(i,2);
    est_csf_calc(i,:)=[pd_mat_slice4(row,col),T1_mat_slice4(row,col)];
end

%%%
for k=1:st_csf
    if est_csf_calc(k,1)<5000 & est_csf_calc(k,2)<5000,
e_csf_calc(k,:)=est_csf_calc(k,:); end;
end

clear est_csf_calc;

[i,j,s] = find(e_csf_calc);

h=(size(s)/2);

est_csf_calc=[s(1:h) s((h+1):(h+h))];
%%%

cov_csf=cov(est_csf_calc);

mean_csf_calc=mean(est_csf_calc);

mean_csf_calc=mean_csf_calc';

det_cov_csf=det(cov_csf);

inv_cov_csf_calc=inv(cov_csf);

c1=2*pi;
c2=c1^2;
c3=c2*det_cov_csf;
c4=sqrt(c3);
c5_csf_calc=1/c4;

clear i j k s h e_csf_calc c1 c2 c3 c4;
clear cov_csf det_cov_csf;
clear col row;

load c:\old\bora\prm_images\prm_mat.mat;

est_wm_calc=zeros(st_wm,2);

for i=1:s1_wm;
    row=w_mat_wm(i,1);
    col=w_mat_wm(i,2);
    est_wm_calc(i,:)=[pd_mat_slicel(row,col),T1_mat_slicel(row,col)];
end
for i=(s1_wm+1):(s1_wm+s2_wm);
    row=w_mat_wm(i,1);

```

```

        col=w_mat_wm(i,2);
        est_wm_calc(i,:)=[pd_mat_slice2(row,col),Tl_mat_slice2(row,col)];
    end
    for i=(s1_wm+s2_wm+1):(s1_wm+s2_wm+s3_wm);
        row=w_mat_wm(i,1);
        col=w_mat_wm(i,2);
        est_wm_calc(i,:)=[pd_mat_slice3(row,col),Tl_mat_slice3(row,col)];
    end
    for i=(s1_wm+s2_wm+s3_wm+1):(s1_wm+s2_wm+s3_wm+s4_wm);
        row=w_mat_wm(i,1);
        col=w_mat_wm(i,2);
        est_wm_calc(i,:)=[pd_mat_slice4(row,col),Tl_mat_slice4(row,col)];
    end

    %%%
    for k=1:st_wm
        if est_wm_calc(k,1)<5000 & est_wm_calc(k,2)<5000,
            e_wm_calc(k,:)=est_wm_calc(k,:); end;
    end

    clear est_wm_calc;

    [i,j,s] = find(e_wm_calc);

    h=(size(s)/2);

    est_wm_calc=[s(1:h) s((h+1):(h+h))];
    %%%

    cov_wm=cov(est_wm_calc);
    mean_wm_calc=mean(est_wm_calc);

    mean_wm_calc=mean_wm_calc';

    det_cov_wm=det(cov_wm);

    inv_cov_wm_calc=inv(cov_wm);

    c1=2*pi;
    c2=c1^2;
    c3=c2*det_cov_wm;
    c4=sqrt(c3);
    c5_wm_calc=1/c4;

    clear i j k s h e_wm_calc c1 c2 c3 c4;
    clear cov_wm det_cov_wm;
    clear col row;

    load c:\old\bora\prm_images\prm_mat.mat;

    est_gm_calc=zeros(st_gm,2);

    for i=1:s1_gm;
        row=w_mat_gm(i,1);
        col=w_mat_gm(i,2);
    end

```

```

    est_gm_calc(i,:)=[pd_mat_slice1(row,col),T1_mat_slice1(row,col)];
end
for i=(s1_gm+1):(s1_gm+s2_gm);
    row=w_mat_gm(i,1);
    col=w_mat_gm(i,2);
    est_gm_calc(i,:)=[pd_mat_slice2(row,col),T1_mat_slice2(row,col)];
end
for i=(s1_gm+s2_gm+1):(s1_gm+s2_gm+s3_gm);
    row=w_mat_gm(i,1);
    col=w_mat_gm(i,2);
    est_gm_calc(i,:)=[pd_mat_slice3(row,col),T1_mat_slice3(row,col)];
end
for i=(s1_gm+s2_gm+s3_gm+1):(s1_gm+s2_gm+s3_gm+s4_gm);
    row=w_mat_gm(i,1);
    col=w_mat_gm(i,2);
    est_gm_calc(i,:)=[pd_mat_slice4(row,col),T1_mat_slice4(row,col)];
end

%%
for k=1:st_gm
    if est_gm_calc(k,1)<5000 & est_gm_calc(k,2)<5000,
e_gm_calc(k,:)=est_gm_calc(k,:); end;
end

clear est_gm_calc;

[i,j,s] = find(e_gm_calc);

h=(size(s)/2);

est_gm_calc=[s(1:h) s((h+1):(h+h))];
%%

cov_gm=cov(est_gm_calc);
mean_gm_calc=mean(est_gm_calc);

mean_gm_calc=mean_gm_calc';

det_cov_gm=det(cov_gm);

inv_cov_gm_calc=inv(cov_gm);

c1=2*pi;
c2=c1^2;
c3=c2*det_cov_gm;
c4=sqrt(c3);
c5_gm_calc=1/c4;

clear i j k s h e_gm_calc c1 c2 c3 c4;
clear cov_gm det_cov_gm;
clear col row;

```

REFERENCES

1. Paty, D. W., D. K. B. Li, "Interferon beta-1b is effective in relapsing-remitting multiple sclerosis. ii. MRI analysis results of a multicenter, randomized, double-blind, placebo-controlled trial", *Neurol.*, Vol. 43, pp. 662-667, April 1993.
2. Paty, D. W., "Interferon beta-1b in the treatment of multiple sclerosis: Final outcome of the randomized controlled trial", *Neurol.*, Vol. 45, pp. 1277-1285, July 1995.
3. Johnston, B., M. S. Atkins, "Segmentation of multiple sclerosis lesions in intensity corrected multispectral MRI", *IEEE Trans. Med. Imag.*, Vol. 15, pp. 154-169, April 1996.
4. Zijdenbos, A. P., B. M. Dawant, "Morphometric analysis of white matter lesions in MR images: Method and validation", *IEEE Trans. Med. Imag.*, Vol. 13, pp. 716-724, Dec. 1994.
5. Dean D., P. Buckley, "Three dimensional MR-based morphometric comparison of schizophrenic and normal cerebral ventricles", *Proc. Visualization in Biomedical Computing*, Vol. 1131, pp. 363-372, Sept. 1996.
6. Freeborough P. A., N. C. Fox, "Assessing patterns and rates of brain atrophy by serial MRI: A segmentation, registration, display and quantification procedure", *Proc. Visualization in Biomedical Computing*, Vol. 1131, pp. 419-428, Sept. 1996.
7. Özkan M., B. M. Dawant, R. J. Maciunas, "Neural-network-based segmentation of Multi-modal medical images: A comparative and prospective study.", *IEEE Trans. Med. Imag.*, Vol. 12, pp. 534-544, Sept. 1993.
8. Vannier M. W., R. L. Butterfield, "Multispectral analysis of magnetic resonance images", *Radiology*, Vol. 154, no. 1 pp. 221-224, 1985.

9. Herskovits, E., "A hybrid classifier for automated radiological diagnosis: Preliminary results and clinical applications", *Computer Methods and Programs in Biomedicine*, Vol. 32, no. 1, pp. 45-52, May 1990.
10. Kohn M. I., N. K. Tanna, G. T. Herman, *et al.*, "Analysis of brain and cerebrospinal fluid volumes with MR imaging: Part I-Methods, reliability, and validation", *Radiology*, Vol. 178, pp. 115-122, 1991.
11. Vannier M. W., R. L. Butterfield, *et al.* "Multispectral analysis of magnetic resonance images," *Radiology*. vol.154, no.1, pp. 221-224, 1985.
12. Schenz L. D., M. V. Vannier, *et al.*, "Statistical assessment of MR image classification techniques," *Proc. Seventh Annual IEEE/EMBS Conf.* pp. 585-592, 1985.
13. Gohagan J. K., E. L. Spitznagel, *et al.*, "Multispectral analysis of MR images of the breast," *Radiology*, vol.163. pp. 703-707, 1987.
14. Ortendahl D. A., J. W. Carlson, "Segmentation of magnetic resonance images using fuzzy clustering," *Information Processing in Medical Imaging* pp. 91-106 C.N. de Gruf and M.A. Viergever. eds. New York: Plenum Press, 1988.
15. Dudewicz E. J., G. C. Levy, *et al.*, "Advanced statistical methods for tissue characteristics," pp. 1793-1803 in *Magnetic Resonance Imaging*, C. L. Partain , R. R. Price, J. A., Patton, M. V. Kulkarni, A. E. James, eds. Philadelphia, PA: W.B. Saunders. 1988.
16. Vannier M. W., R. L. Butterfield, *et al.*, "Multispectral magnetic resonance image analysis." *Critical Reviews in Biomedical Engineering*. vol.15, no.2, pp. 117-141, 1987.
17. Hyman T. J., R. J. Kurland, *et al.*, "Characterization of normal brain tissue using seven calculated MR parameters and a statistical analysis system," *Magn. Reson.*

Med., vol.11. no.1, pp.22-34, July 1989.

18. Alaux A. and P. A. Rick, "Multispectral analysis of magnetic resonance images: A comparison between supervised and unsupervised classification techniques", *Proc. Int. Symp. on Tissue Characterization in MR imaging*, Wiesbaden, April 19, 1989.
19. Gerig G., J. Martin, *et al.* "Automatic segmentation of dual-echo MR head data," *Information Processing in Medical imaging*. A.C.F. Colchester and D.J. Hawkes, eds. (Lecture Notes in Computer Science), vol. 511, pp.175-185, July 1991.
20. Özkan M., H. G. Sprenkels, B. M. Dawant, "Multi-spectral resonance image segmentation using neural networks," in *Proc. IJCNN 90*, vol.1, pp. 429-437, San Diego, June 1990.
21. Dawant B. M., M. Özkan, *et al.*, "A neural network approach to magnetic resonance imaging tissue characterization," in *Proc. 1990 Bilkent Int. Conf on New Trends in Communication, Control and Signal Processing*, vol, 2, pp. 1803-1809, Ankara, July 1990.
22. Hornak, J. P., 1997, The Basics of MRI, A Hypertext Book on Magnetic Resonance Imaging, <http://www.cis.rit.edu/htbooks/mri/>
23. Fletcher, L. M., J. B. Barsotti, J. P. Hornak, "A Multispectral Analysis of Brain Tissues", *Magnetic Resonance in Medicine*, Vol. 29, pp. 623-630, 1993.
24. Bushong, S. C., *Magnetic Resonance Imaging, Physical and Biological Principles*, Mosby, USA, 1996.
25. Liang, Z. and P. C. Lauterbur, *Principles of Magnetic Resonance Imaging*, IEEE Press, New York, 1999.
26. Cadzow, J. A., *Foundations of Digital Signal Processing and Data Analysis*, Macmillan, USA, 1987.