

**A Comparison of Actual and Synthesized Magnetic
Resonance Images**

Sangyun Moon

Advisor: Dr. Joseph P. Hornak

Center for Imaging Science

Rochester Institute of Technology

May 25, 2006

Copyright © 2006

Chester F. Carlson Center for Imaging Science

Rochester Institute of Technology

Rochester, NY 14623-5604

This work is copyrighted and may not be reproduced in whole or part without permission of the Center for Imaging Science at the Rochester Institute of Technology.

A Comparison of Actual and Synthesized Magnetic Resonance Images

Sangyun Moon

Chester F. Carlson Center for Imaging Science
Rochester Institute of Technology
Rochester, NY 14623-5604

May 2005

Abstract:

Magnetic resonance imaging is a medical imaging technique used for taking tomographic pictures of the inside of the human body. It has greater diagnostic utility than other imaging techniques such as ultrasound, computed tomography (CT), positron emission tomography (PET), or X-ray. For example, unlike other imaging techniques, magnetic resonance images can be taken in any plane; coronal, sagittal, axial, and oblique. Therefore, the human body can be viewed in any direction. In order to locate pathology in the human body, several magnetic resonance images having different image contrast are necessary. The contrast of the image is varied based on the pulse sequences used in the MRI instrument. The most commonly used pulse sequences are spin-echo, inversion recovery, and gradient echo pulse sequence. Those pulse sequences have their own signal equations, so if the equation is used, the synthetic image can be generated. A magnetic resonance imaging synthetic image generator (MRISIG) was written to synthesize magnetic resonance sequences from these pulse sequences. If there is little difference between generated synthetic image and real magnetic resonance image, the MRISIG will be useful software to predict optimal sequence for diagnosing disease.

Acknowledgement

I would like to thank my advisor, Dr. Joseph P.Hornak for giving me this great research experience.

Table of Contents

| | |
|--|-----|
| Copyright Release | i |
| Abstract | ii |
| Acknowledgement | iii |
| Table of Contents | iv |
| Table of Symbols and Abbreviation..... | v |
| Introduction | 1 |
| Background..... | 2 |
| Experimental Methods | 8 |
| Results | 10 |
| Conclusions | 13 |
| References..... | 14 |
| Appendix A – Results | 15 |
| Appendix B - IDL code | 28 |

Table of Symbols and Abbreviation

| Symbols and abbreviations | Definition |
|---------------------------|--|
| MR | Magnetic resonance |
| NMR | Nuclear magnetic resonance |
| MRI | Magnetic resonance imaging |
| MRISIG | Magnetic resonance imaging synthetic image generator |
| IDL | Interactive data language |
| T_1 | Spin-lattice relaxation time |
| T_2 | Spin-spin relaxation time |
| FOV | Field of view |
| ρ | Spin density |
| TR | Repetition time |
| TE | Echo time |
| TI | Inversion time |
| Θ | Rotation time |
| G_ϕ | Phase encoding gradient |
| G_s | Slice selection gradient |
| G_f | Frequency encoding gradient |
| RF | Radio frequency |

Introduction

The purpose of this project is to develop a magnetic resonance imaging synthetic image generator (MRISIG) and find the differences between real magnetic resonance images and the synthetic images. The theory behind the MRISIG is relatively simple. The MRI machine creates a static magnetic field inside of the machine. A perturbation is sent into the human body in the form of a radio frequency (RF) pulse. [1] A Special coil receives the signal which is radiated back from the human body. The perturbations are prescribed by a pulse sequence. The MRISIG uses the equation of this pulse sequence. To use the equation in MRISIG, three tissue dependent images are necessary; a spin-lattice relaxation time image (T_1), a spin-spin relaxation time image (T_2), and a spin density (ρ) image. Every tissue in the human body has its own T_1 , T_2 , and ρ values. These images are provided by my advisor. [2] In addition to these images, acquisition parameters for the pulse sequence must be defined.

One of the utilities of a MRISIG is to predict image contrast for a given set of acquisition parameters. Since images which have different image contrast can be generated from a MRISIG without taking magnetic resonance images, imaging time could be saved. Another utility is to help determining optimal acquisition parameters to locate pathology.

MRISIG was developed using Interactive Data Language (IDL) and has a simple graphic user interface so the user can enter imaging acquisition parameters such as pulse repetition time (TR), echo time (TE), inversion time (TI), and rotation angle (Θ).

In order to test the success, the average RMS value of difference between a synthetic and actual image is used. RMS is a statistical measure of the dispersion or spread of data. In this project, RMS represents the differences in pixel values between two images. The average RMS value shows the average difference for each of the pixel. In addition to that, the image registration technique is used. The image registration technique shows a specific area which contains the differences. Also, it shows how much difference exists in that area.

Background

The magnetic resonance imaging (MRI) process can be divided into two parts for this project. The first is receiving a radiated signal from the human body after scanning the body in the magnetic field, and the second is processing the signal to form the image. In order to let the human body radiate the signal, MRI uses spin characteristic of the nuclei in the body.

1. Tomographic imaging

Magnetic resonance started out as a tomographic imaging modality for producing NMR images of a slice through the human body. A tomographic image is the image of slice. Each slice had a thickness (thk) and composed of several volume elements called voxels. A voxel is defined as a three dimension rectangular section of the subject being imaged. The signal from a voxel contributes to the intensity of a pixel in the image. For this project, a voxel is assumed to be composed of one kind of tissue, meaning only one type of signal is radiating from voxel. In reality, a voxel can include one or more type of tissues so a voxel radiates more than one type of signals.

The resolution of a tomographic image is determined by slice thickness and number of pixels across the field of view (FOV). As slice thickness increases, signal intensity and signal-to-noise ratio (SNR) increase, but the spatial resolution decreases perpendicular to the slice or in the slice thickness direction. The field of view is the width or height of a square image that contains the object of interest to be measured. For the constant number of pixels, the smaller the FOV, the higher the resolution. SNR decreases with FOV. [2]

2. Relaxation Process

The nuclear spin has not only a vertical component but also a horizontal component. When the spinning particles are in the magnetic field, the magnetic force affects the particles. The magnetic field influences both directions when the particles are in the magnetic field. The net magnetization of the particle can be separated into vertical and horizontal components.

2.1. Longitudinal Relaxation Time (T_1)

When the particles are in the magnetic field, the spins have a direction and the net amplitude. Let the direction of the net spin be the same as direction of the static magnetic field (B_0) along the z-axis. This can be represented by a magnetization vector M_0 called equilibrium magnetization. After the magnetic field B_0 is applied to the particle, the magnetic field creates a magnetization. This net magnetization can be perturbed from its equilibrium value by the application of a radio frequency (RF) pulse. The value of the z component of magnetization at any time is referred to as M_z .

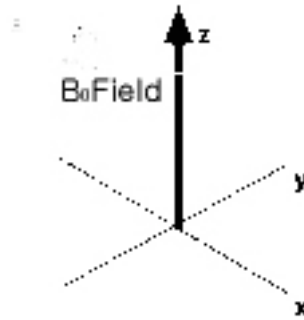


Figure 1. The magnetization vector, M_0 .

After a perturbation from equilibrium the net magnetization returns to its equilibrium value. The time duration between the net magnetization and the equilibrium value of the net magnetization is governed the longitudinal relaxation time or spin-lattice relaxation time (T_1).

2.2. T_2 process (transverse relaxation time)

When the net magnetization is placed in the x-y plane, the particle will rotate about z-axis at the Larmor frequency. [1] The net magnetization starts to dephase. The time constant which describes the return to equilibrium of the transverse magnetization, M_{xy} is called the spin-spin relaxation time, T_2 is described as

$$M_{XY} = M_{XY0} e^{-t/T} \quad (1)$$

where M_{xy0} is the initial perturbed value of the transverse magnetization and t is time after the perturbation. T_1 is always greater than or equal to T_2 . Therefore, when the net magnetization returns to the equilibrium, x,y magnetization decays to zero first, followed

by the z magnetization at a slower rate. The human body's tissues have different T_1 and T_2 values. For example, tissues found in a magnetic resonance image of the human head have the unique values shown in Table 1. [1]

Table 1. T_1 , T_2 , and spin density of human brain [1]

| Tissue | T_1 (s) | T_2 (ms) | ρ^* |
|----------|-------------|------------|----------|
| CSF | 0.8 - 20 | 110 - 2000 | 70-230 |
| White | 0.76 - 1.08 | 61-100 | 70-90 |
| Gray | 1.09 - 2.15 | 61 - 109 | 85 - 125 |
| Meninges | 0.5 - 2.2 | 50 - 165 | 5 - 44 |
| Muscle | 0.95 - 1.82 | 20 - 67 | 45 - 90 |
| Adipose | 0.2 - 0.75 | 53 - 94 | 50 - 100 |

* Based on $\rho = 111$ for 12mM aqueous NiCl_2

The MRI signal from the human body depends on T_1 , T_2 , and the spin density of the tissue. The exact signal depends on the pulse sequence or set of perturbations applied. Each pulse sequence is unique and has a set of acquisition parameters. The value of these parameters determines the signal.

3. Pulse Sequence

For MRI, image contrast can be explained as the difference of signal intensity. The signal intensity is determined base on the pulse sequence which is prescribed by the operator of the MRI machine. The pulse sequence has a unique equation which describes the signal. The equations are a function of the intrinsic variables; T_1 , T_2 and ρ , and instrumental variables; repetition time (TR), Echo time (TE), inversion time (TI), and rotation angle (Θ). There are signal equations for each specific sequence. The four common pulse sequences are the 90-FID, spin-echo, inversion recovery, and gradient echo. The signals are as follows.

90-FID

$$S = k \rho (1 - \exp(-TR/T_1)) \quad (2)$$

Spin-Echo

$$S = k \rho (1 - \exp(-TR/T_1)) \exp(-TE/T_2) \quad (3)$$

Inversion Recovery (180-90)

$$S = k \rho (1 - 2\exp(-TI/T_1) + \exp(-TR/T_1)) \quad (4)$$

Inversion Recovery (180-90-180)

$$S = k \rho (1 - 2\exp(-TI/T_1) + \exp(-TR/T_1)) \exp(-TE/T_2) \quad (5)$$

Gradient Recalled Echo

$$S = k \rho (1 - \exp(-TR/T_1)) \sin\theta \exp(-TE/T_2^*) / (1 - \cos\theta \exp(-TR/T_1)) \quad (6)$$

Each of these equations, S represents the amplitude of the signal in the frequency domain spectrum. The quantity k is a proportionality constant which depends on the sensitivity of the detection circuitry on the imager.

3.1. 90-FID sequence

In the 90-FID sequence pulse, the net magnetization is rotated down to the $x'y'$ plane where $x'y'$ is the rotating frame of reference. [1] Then the net magnetization vector begins to rotate about z -axis, and it decay with time. When this sequence is repeated, for example when signal-to-noise improvement is needed, and amplitude of the signal after being Fourier transformed will depend on T_1 and repetition time. The equation for a 90-FID sequence was presented as Equation 2.

3.2. The Spin-Echo Sequence

In the spin-echo sequence, a 90° pulse is applied with a slice selection gradient. A period of time equal to $TE/2$ elapses and an 180° slice selective 180° pulse is applied in conjunction with the slice selection gradient. A phase encoding gradient is applied between the 90° and 180° pulse. The phase encoding gradient could be applied after the 180° pulse. Figure 6. shows the time diagram for a spin-echo imaging sequence.

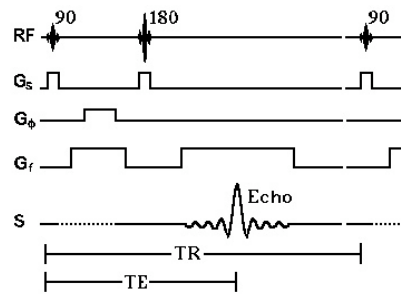


Figure 2. Time diagram of the spin-echo pulse sequence. [1]

The signal from Spin-Echo sequence can be expressed as Equation 3, where TE is echo time.

3.3. The Inversion Recovery Sequence

The inversion recovery sequence can come in two forms: a 180-90 and 180-90-180. In the 180-90-180 inversion recovery sequence, a selective 180° RF pulse is applied in conjunction with a slice selection gradient. A period of time equal T_1 elapses and a spin-echo sequence is applied. This spin-echo part recorded the magnetization present at a time TI after the first 180° pulse. A phase encoding gradient is applied between 90° and 180° pulse. Figure 3. shows the time diagram for the inversion recovery imaging sequence.

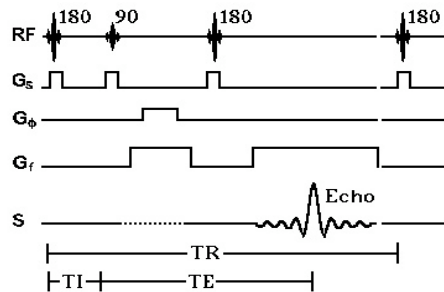


Figure 3. Time diagram of the inversion recovery pulse sequence. [1]

When an inversion recovery sequence is repeated every TR seconds, the signal equation becomes Equation 5. The signal equation for a 180-90 version is represented by Equation 4.

3.4. The Gradient Recalled Echo

The purpose of the gradient recalled echo is to shorten the imaging sequence time. The spin-echo sequence and the inversion recovery sequence require returning the transverse and the longitude magnetization to the equilibrium. Therefore if T_1 is long, the total recovery time is lengthened. However, the recovery time can be less than it would be if the magnetization is rotated by an angle Θ less than 90 degrees. However, since the signal will be proportional to the $\text{Sin } \Theta$, there will be less signal. So the gradient recalled echo sequence trades off signal for imaging time. In the gradient recalled echo imaging sequence a slice selective RF pulse is applied to the imaged object. This RF pulse typically produces a rotation angle of between 10° and 90° . A Slice selection gradient is applied with the RF pulse. A phase encoding gradient is applied next. Figure 4. shows the time diagram for the gradient recalled echo imaging sequence.

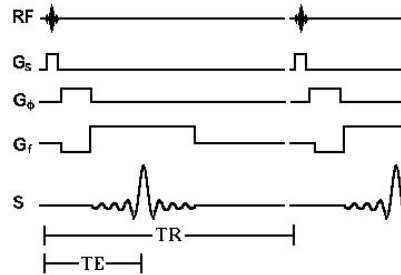


Figure 4. Time diagram of the gradient recalled echo pulse sequence. [1]

Equation for the gradient recalled echo pulse sequence is Equation 6.

Experimental Methods

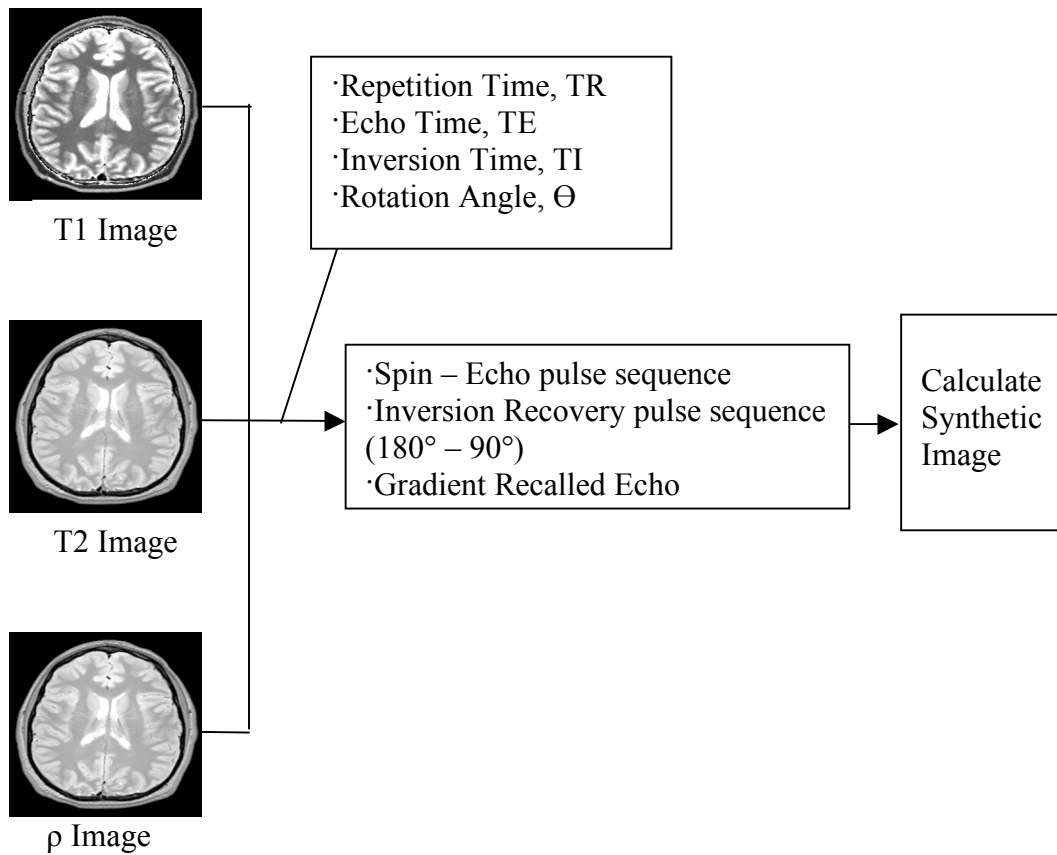


Figure 5. Flow of information in the synthetic image generator.

Figure.5 depicts the flow of information in MRISIG. The input of the program is three MR images: T₁, T₂, and spin density image (ρ image). The MR images are of the brain of a 25- year normal male volunteer. The image is acquired with a 1.5 T General Electric (Milwaukee, WI) Signa imager with shielded gradient coils and a standard quadrate bird cage head coil. The volunteer was imaged in the supine position with a single-echo sequence. Imaging parameters for the image sequence were: TR/TE = 4000, 3000, 2000, 1500, 1000, 750, 500, 250/15 and 1000/25, 50, 75, 100, 150, 200 ms, 256 X 192 matrix, 5-mm slice thickness (Thk), one excitation (NEX), and 24 cm field of view (FOV). [2] After the input image is entered, user will choose one pulse sequence. If the spin–echo pulse sequence is chosen, the program will allow user to choose TR and TE as an imaging parameter. The gradient recalled echo and the inversion recovery echo pulse sequence have imaging parameters: TR, Θ and TE; and TR, TE, and TI respectively.

Then using signal equation (2) ~ (6), signal intensity will be computed, and the synthetic image is created.

After the image is calculated, the actual MR image and the calculated image are compared by root mean square (RMS) difference. RMS is used to measure the differences of two images statistically. If the images are exactly identical, this value is zero. Let $f'(x,y)$ be the calculated image and $f(x,y)$ be the actual image. First calculate square of the difference between two images.

$$\frac{1}{MN} \sum_{x=1}^{m-1} \sum_{y=1}^{n-1} [f'(x,y) - f(x,y)]^2 \quad (7)$$

In order to make $f'(x,y)$ close to $f(x,y)$, scaling factor, k , is used.

$$\frac{1}{MN} \sum_{x=1}^{m-1} \sum_{y=1}^{n-1} [K * f'(x,y) - f(x,y)]^2 \quad (8)$$

The RMS error (e_{RMS}) is defined as

$$e_{\text{RMS}} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [k * f'(x,y) - f(x,y)]^2} . \quad (9)$$

Results

The Synthetic image generator was developed using IDL. In order to compare the synthetic and real images, fourteen synthetic images were generated with spin-echo pulse sequence and TR/TE values of 4000, 3000, 2000, 1500, 1000, 750, 500, 250/15 and 1000/25, 50, 75, 100, 150, 200ms. These synthetic images were compared with real MR images.



Figure 6. The real and synthetic spin-echo images with a TR=250 and TE=15ms.

Figure 6. shows the real image and the synthetic image using spin-echo pulse sequence, TR = 250 ms, and TE = 15ms. The white circles at the back left and right of the head are intensity standards. There are not significant differences between the synthetic image and the real image except that the real image is brighter than the synthetic image. Also some noise is observed in the real image.

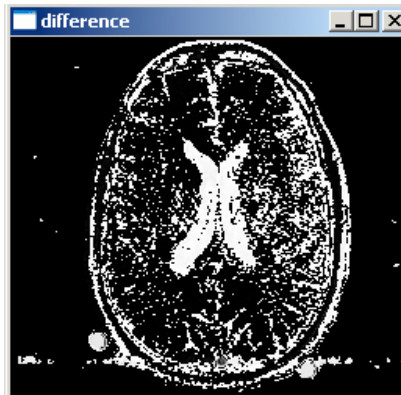


Figure 7. The Intensity difference between images in Figure 6.

Figure 7. represents the intensity difference. The white area was where two images have the same intensity and the black area was where the intensity differences existed. Since the real MR image had some noise, there were some white spots in the background.

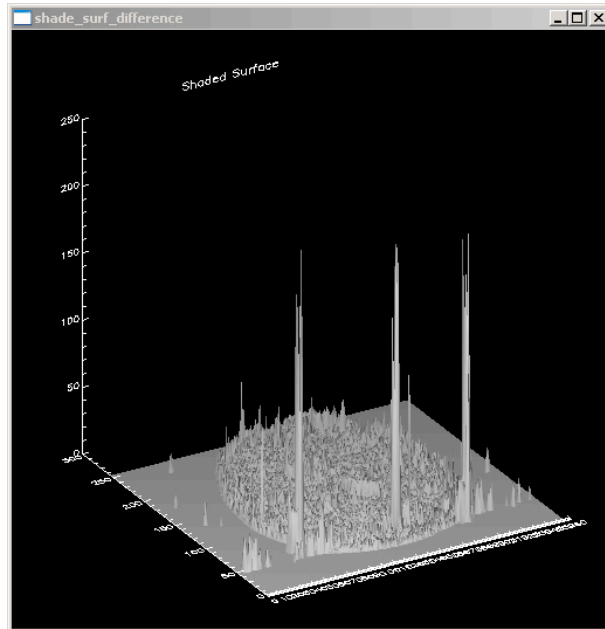


Figure 8. Plot of the intensity differences between images in Figure 6.

Figure 8. is the plot of the intensity difference. The plot shows the specific area that has difference. As shown in Figure 8. there are three spikes on the plot. Those areas are artificial spots which are created while taking MRI. The purpose of the three spots is to mark standardized area. If the difference in these areas is discounted, the RMS value could decrease. The remaining TR/TE combinations are presented in Appendix A.

Table 2. Shows the RMS values and the average RMS values as defined by Equation 9 for these images. As imaging time (TR) goes up, the RMS value increases. For example, when TR is 250 milliseconds and TE is 15 milliseconds, the RMS was the lowest, 115575. When TR is 1 second and TE is 200 milliseconds, RMS value was the highest, 1000003.

Table 2. RMS and average RMS differences between the synthetic and real image.

| RESULT | TR (ms) | TE (ms) | RMS | Average RMS |
|--------|------------|------------|---------|----------------|
| 1 | 4000 | 15 | 265999 | 4 |
| 2 | 3000 | | 245143 | 3.7 |
| 3 | 2000 | | 237611 | 3.6 |
| 4 | 1500 | | 200771 | 3 |
| 5 | 1000 | | 171595 | 2.6 |
| 6 | 750 | | 150750 | 2.3 |
| 7 | 500 | | 126853 | 1.9 |
| 8 | 250 | | 115575 | 1.7 |
| 9 | 1000 | 25 | 226869 | 3.4 |
| 10 | | 50 | 322577 | 4.9 |
| 11 | | 75 | 405886 | 6.2 |
| 12 | | 100 | 412452 | 6.2 |
| 13 | | 150 | 632761 | 9.6 |
| 14 | | 200 | 1000003 | 15.2 |

Although MRISIG can calculate inversion recovery and gradient echo image, none are presented because T_1 , T_2 and ρ images were not available for the corresponding real inversion recovery and gradient echo images.

Conclusions

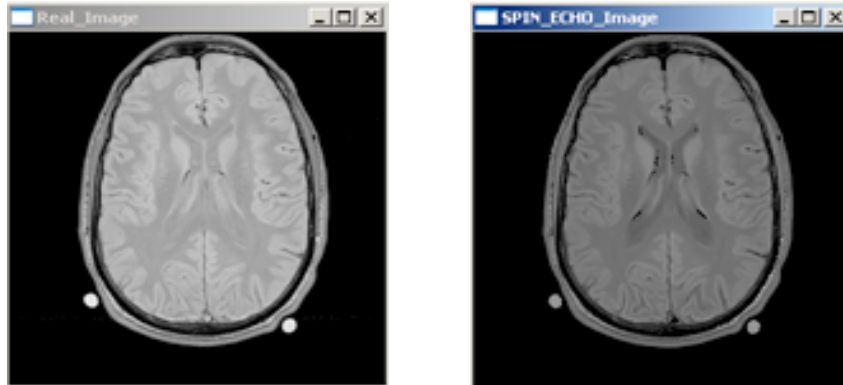
A magnetic resonance imaging synthetic image generator computed magnetic resonance images. MRISIG should be most accurate when voxel size was small. The synthesized spin-echo images with a repetition time of 250 ms and echo time 15 ms had the smallest RMS value. In general, the synthetic image was darker than the real image, even after scaling by k and differed the most from the actual image for cerebral spinal fluid. MRISGI did not synthesize flow artifacts nor signal variations due to inhomogeneous RF magnetic field. Future work will be incorporate inhomogeneities in RF magnetic field into MRISIG.

Reference

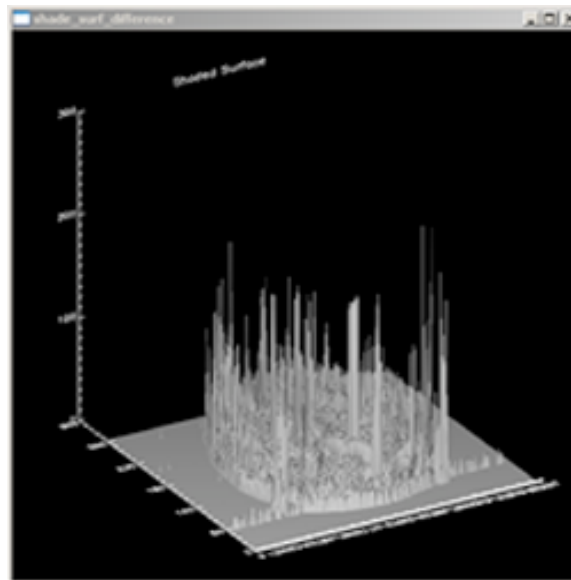
- [1] J.P.Hornak, The Basic of MRI, Interactive Learning Software,
<http://www.cis.rit.edu/htbooks/mri>
- [2] L.M. Fletcher, J.B. Barsotti, J.P. Hornak, "A Multispectral Analysis of Brain Tissues." *Magn. Reson. Med.* 29:623-630 (1993).

Appendix A

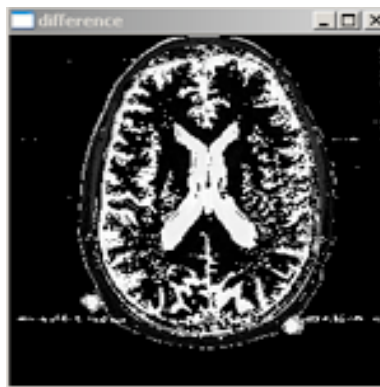
1. Result of using Spin-Echo pulse sequence with TR = 4000ms, TE = 15ms



Result1.a. The real image and the synthetic image.

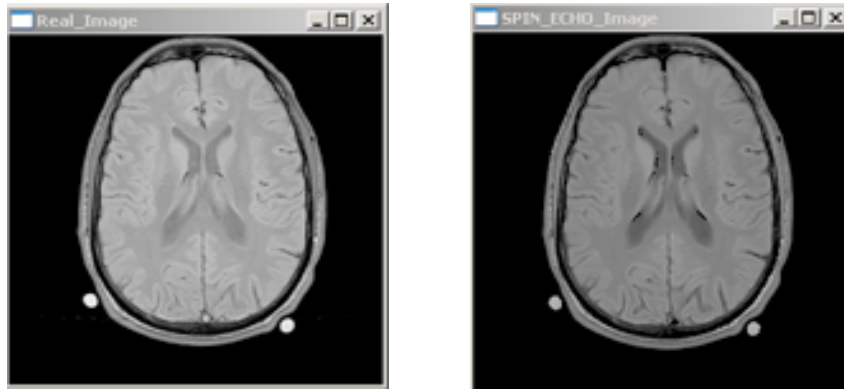


Result1.b. Plot of the intensity differences

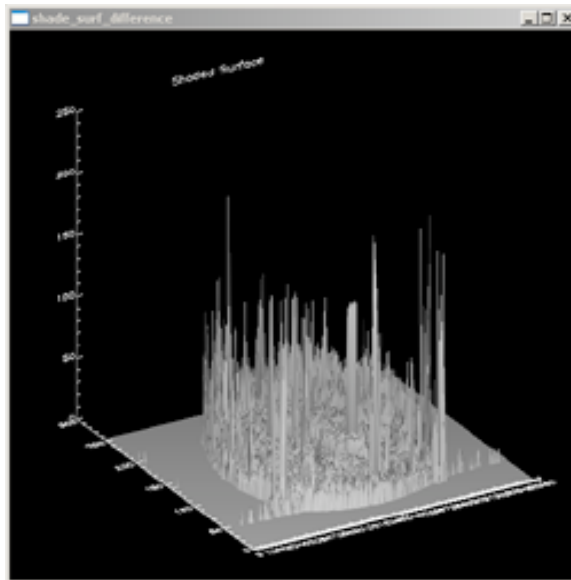


Result1.c. The Intensity difference between images

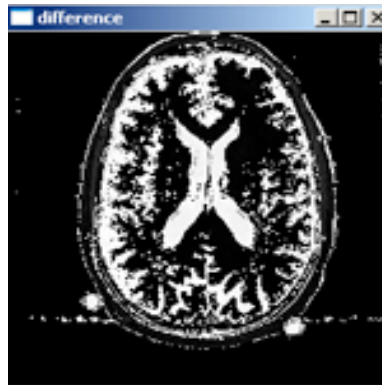
2. Result of using Spin-Echo pulse sequence with TR = 3000ms, TE = 15ms



Result 2.a. The real image and the synthetic image.

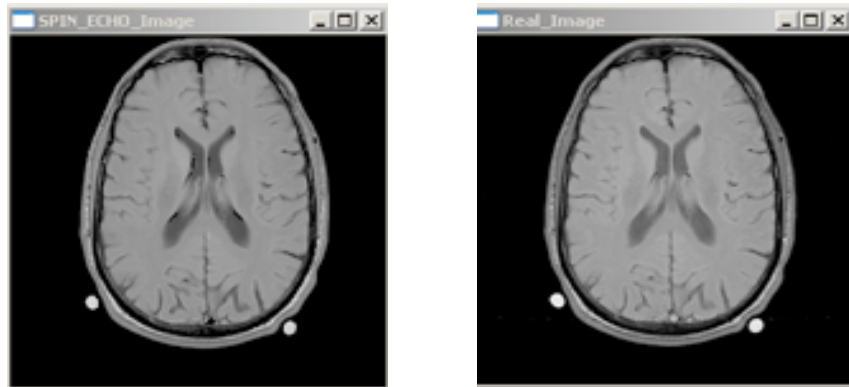


Result 2.b. Plot of the intensity differences

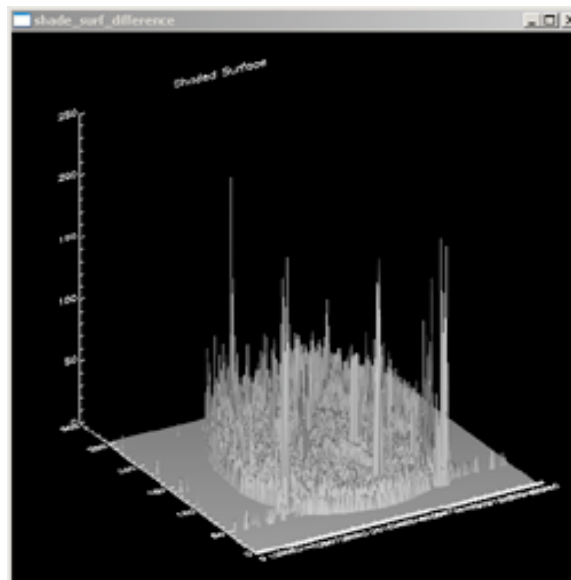


Result2.c. The Intensity difference between images

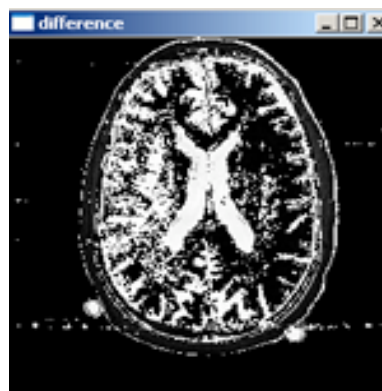
3. Result of using Spin-Echo pulse sequence with TR = 2000ms, TE = 15ms



Result 3.a. The real image and the synthetic image.

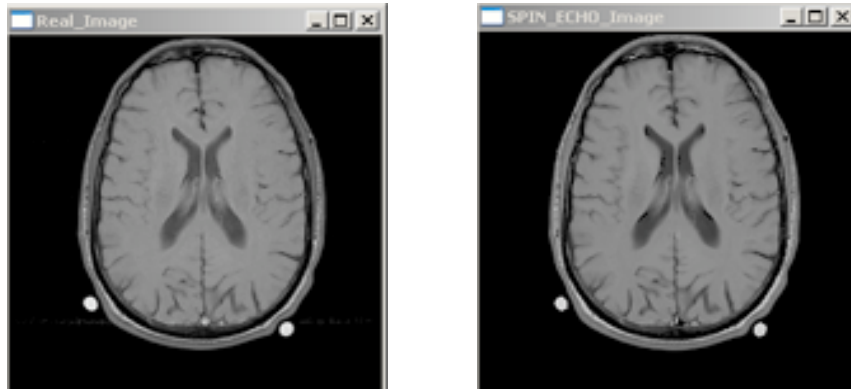


Result 3.b. Plot of the intensity differences

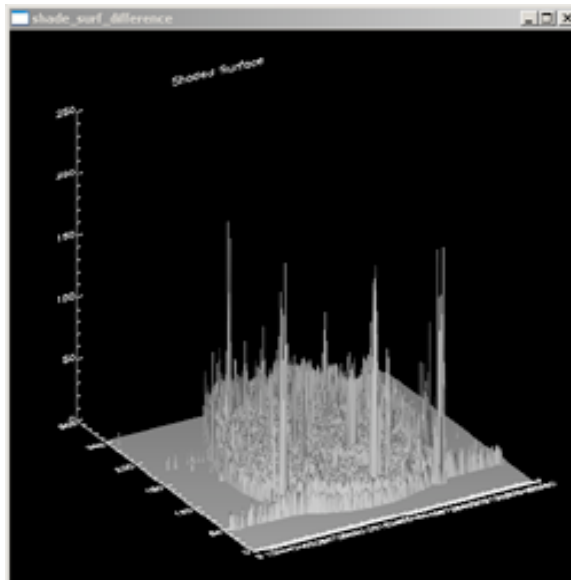


Result 3.c. The Intensity difference between images

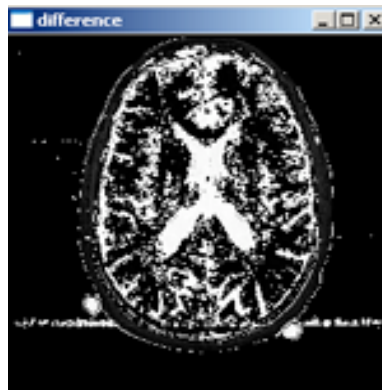
4. Result of using Spin-Echo pulse sequence with TR = 1500ms, TE = 15ms



Result 4.a. The real image and the synthetic image.

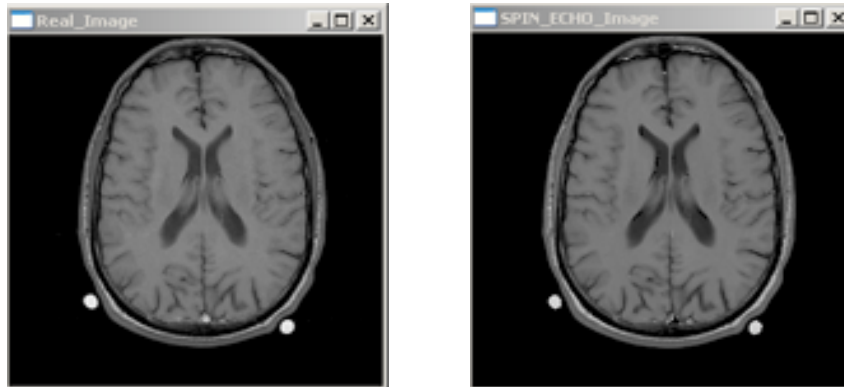


Result 4.b. Plot of the intensity differences

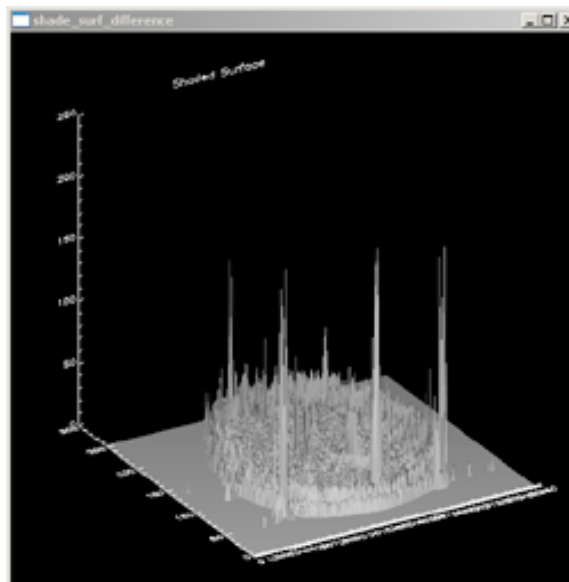


Result 4.c. The Intensity difference between images

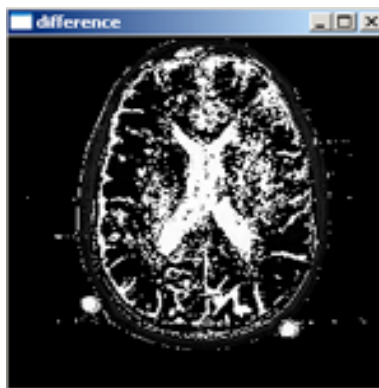
5. Result of using Spin-Echo pulse sequence with $TR = 1000\text{ms}$, $TE = 15\text{ms}$



Result 5.a. The real image and the synthetic image.

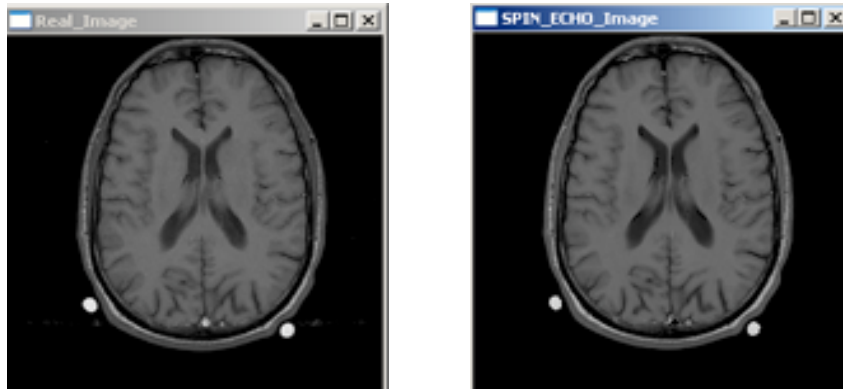


Result 5.b. Plot of the intensity differences

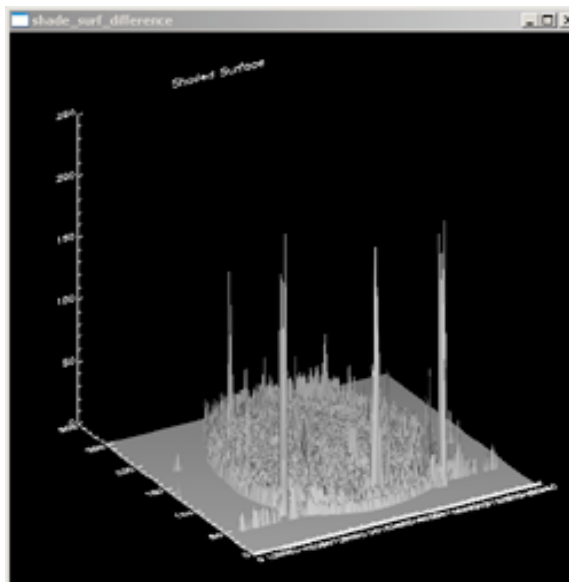


Result 5.c. The Intensity difference between images

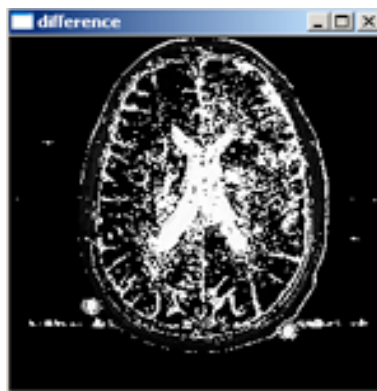
6. Result of using Spin-Echo pulse sequence with $TR = 750\text{ms}$, $TE = 15\text{ms}$



Result 6.a. The real image and the synthetic image.



Result 6.b. Plot of the intensity differences



Result 6.c. The Intensity difference between images

7. Result of using Spin-Echo pulse sequence with TR = 500ms, TE = 15ms

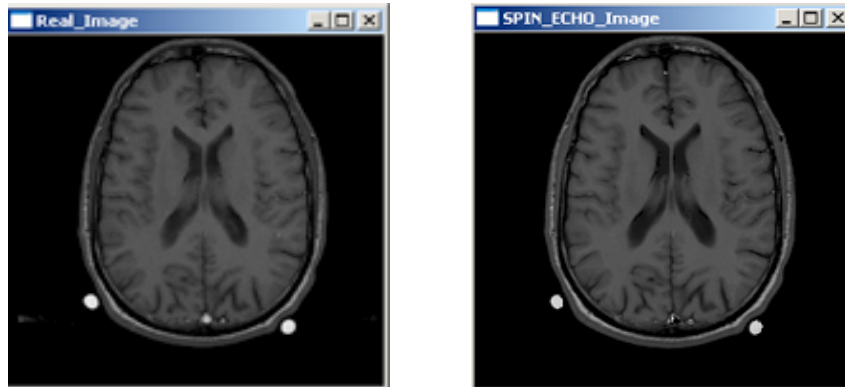


Figure 7.a. The real image and the synthetic image.

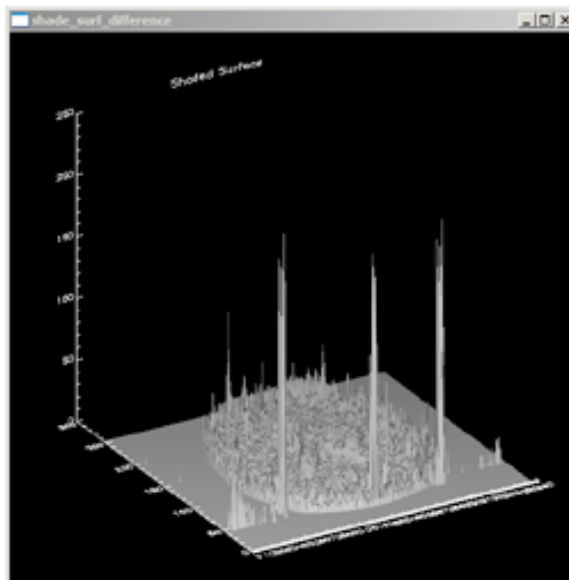


Figure 7.c. Plot of the intensity differences

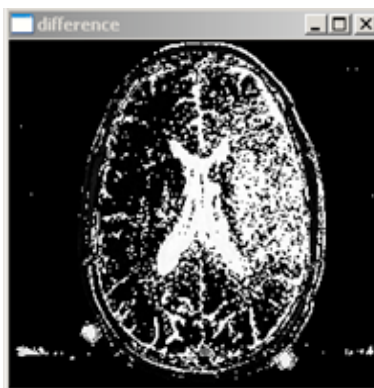


Figure 7.b. The Intensity difference between images

8. Result of using Spin-Echo pulse sequence with TR = 250ms, TE = 15ms

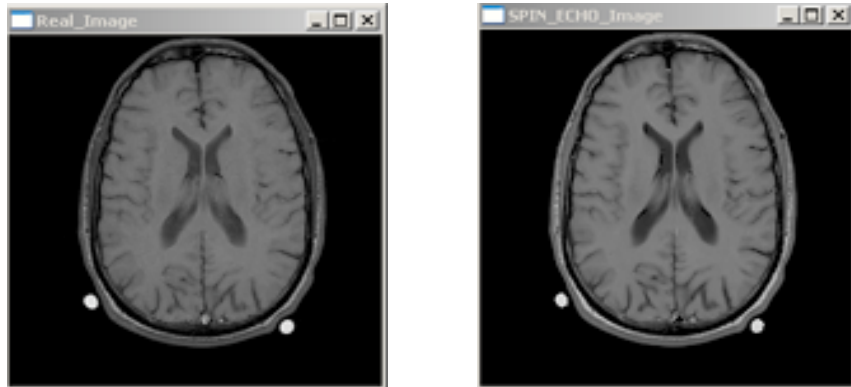


Figure 8.a. The real image and the synthetic image.

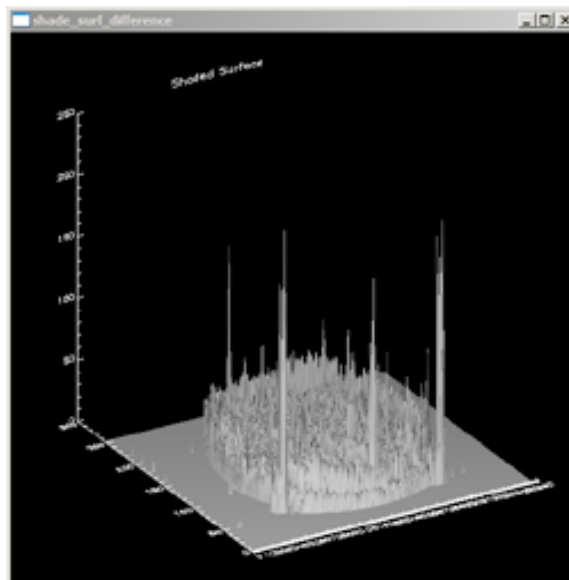


Figure 8.b. Plot of the intensity differences

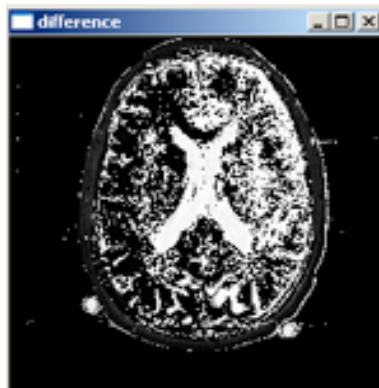


Figure 8.c. Plot of the intensity differences

9. Result of using Spin-Echo pulse sequence with TR = 1000ms, TE = 50ms

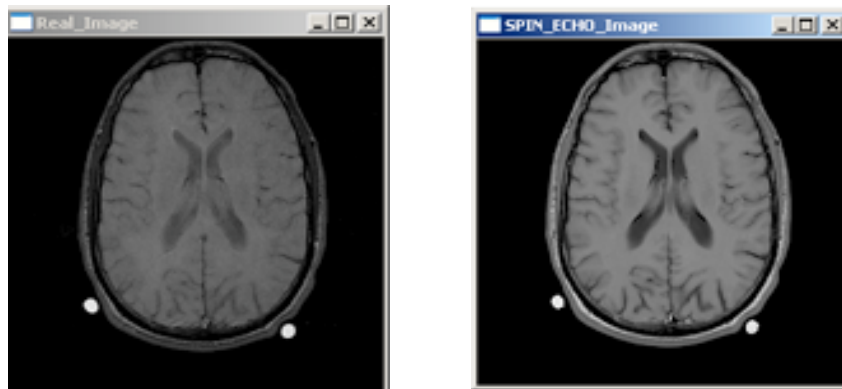


Figure 9.a. The real image and the synthetic image.

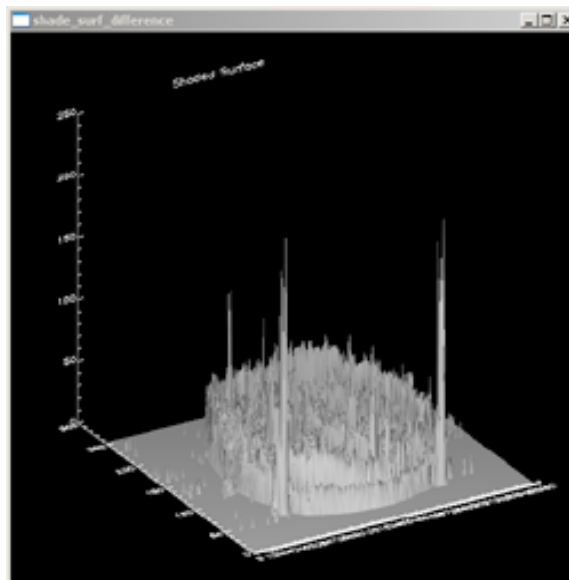


Figure 9.b. Plot of the intensity differences

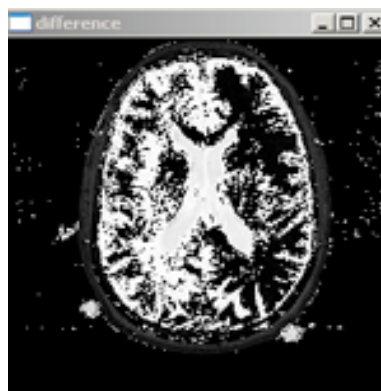


Figure 9.c. Plot of the intensity differences

10. Result of using Spin-Echo pulse sequence with TR = 1000ms, TE = 75ms

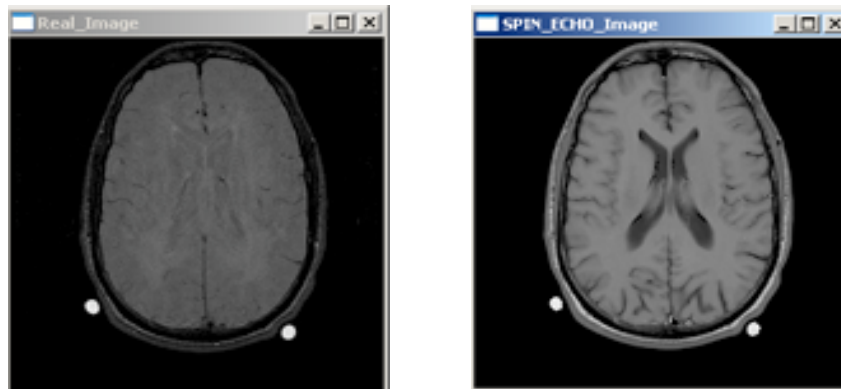


Figure 10.a. The real image and the synthetic image.

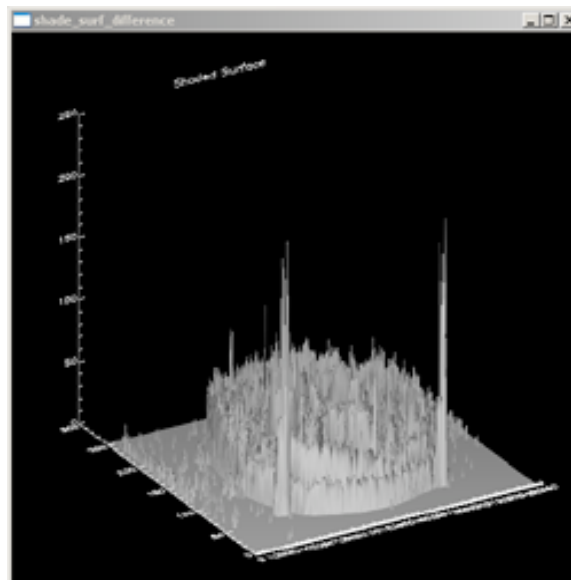


Figure 10.b. Plot of the intensity differences

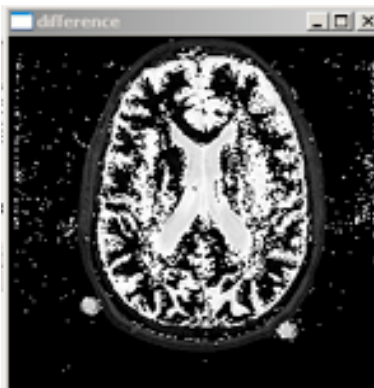


Figure 10.c. Plot of the intensity differences

11. Result of using Spin-Echo pulse sequence with TR = 1000ms, TE = 100ms

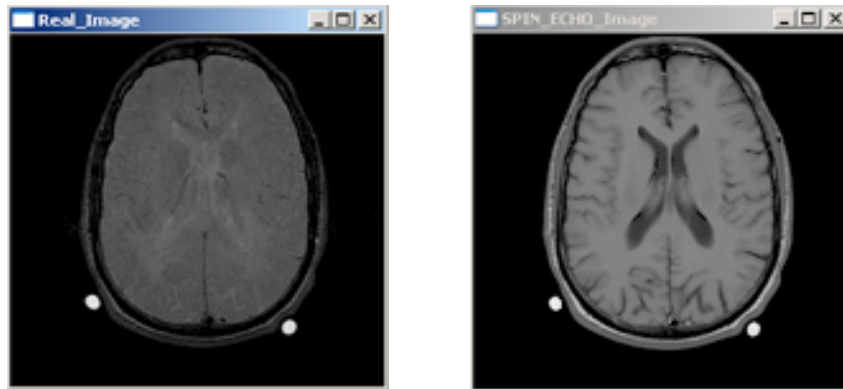


Figure 11.a. The real image and the synthetic image.

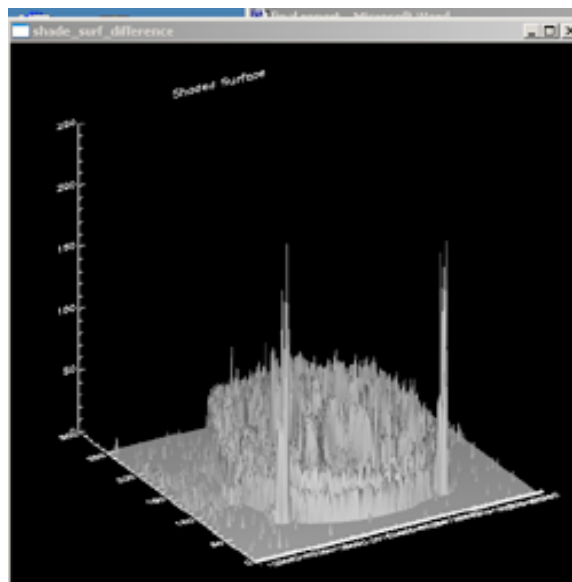


Figure 11.b. Plot of the intensity differences



Figure 11.c. Plot of the intensity differences

12. Result of using Spin-Echo pulse sequence with TR = 1000ms, TE = 150ms

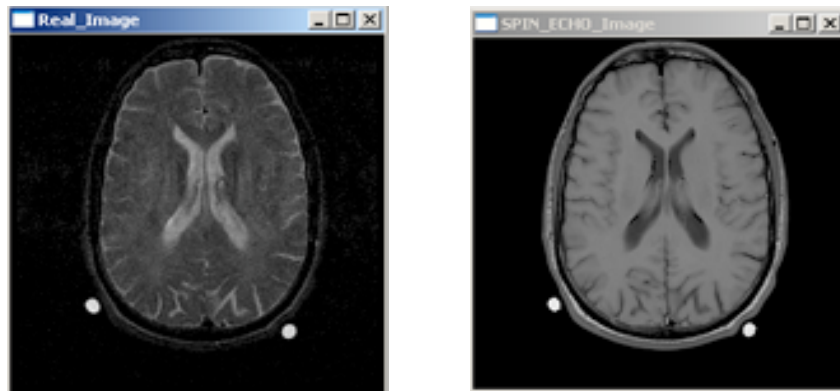


Figure 12.a. The real image and the synthetic image.

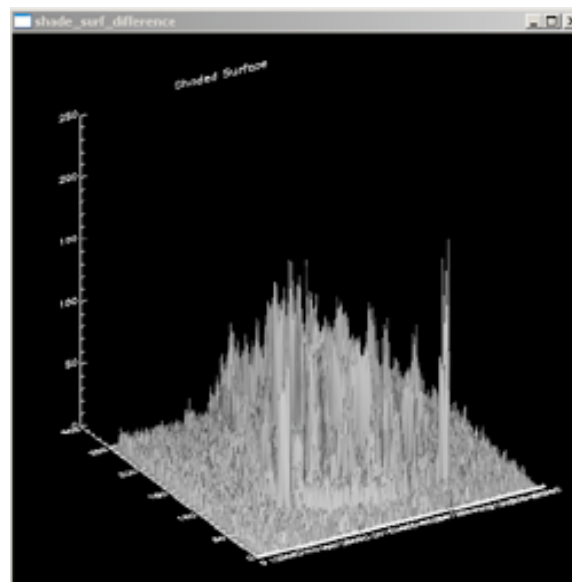


Figure 12.b. Plot of the intensity differences

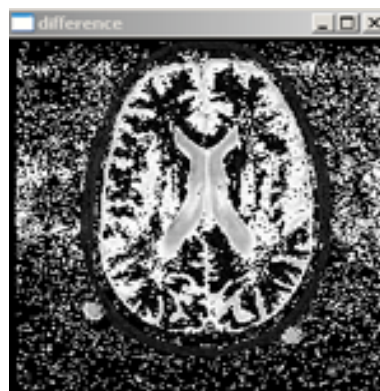


Figure 12.c. Plot of the intensity differences

13. Result of using Spin-Echo pulse sequence with TR = 1000ms, TE = 200ms

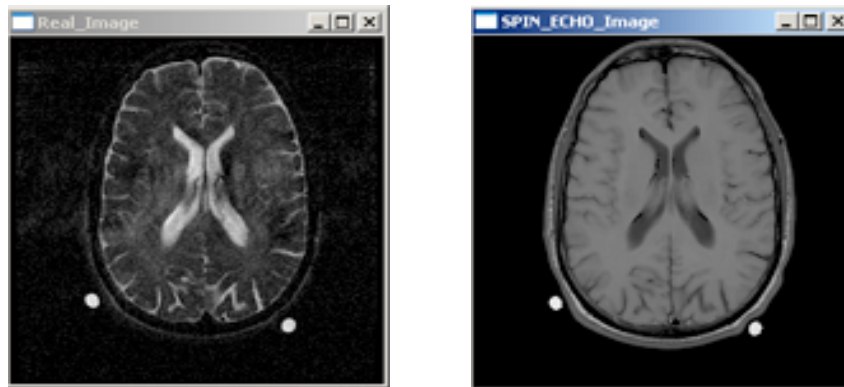


Figure 13.a. The real image and the synthetic image.

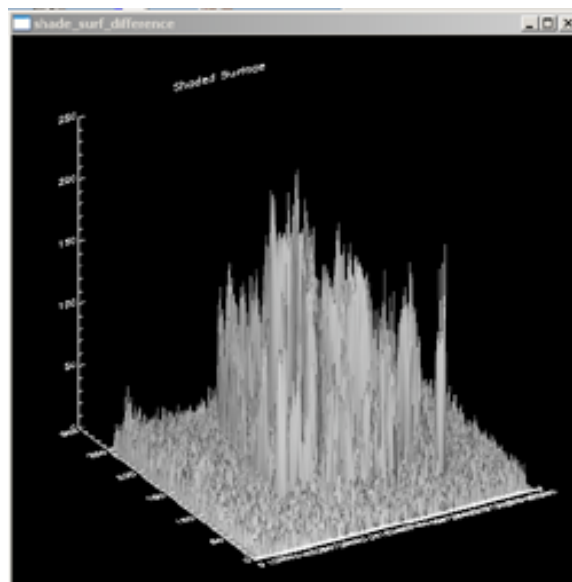


Figure 13.b. Plot of the intensity differences

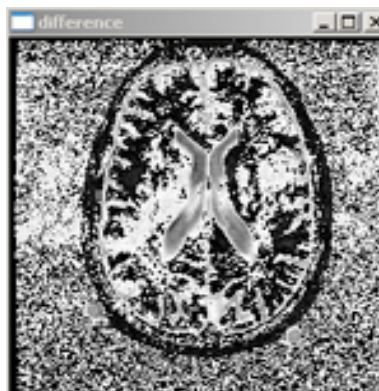


Figure 13.c. Plot of the intensity differences

Appendix B

IDL code

1. MRISIG.pro

```
PRO DIP_EVENT, event

WIDGET_CONTROL, event.TOP, GET_UVALUE=state, /NO_COPY

state.mainEvent = event.TOP

WIDGET_CONTROL, event.ID, GET_UVALUE=widget

CASE widget OF

'File': $
  BEGIN

    CASE event.VALUE OF

      'Load Image...': $
        BEGIN
          LOAD_IMAGE, state
        END

      'Save Image...': $
        BEGIN
          imageFilename = $
            DIALOG_PICKFILE( TITLE='Save Image As', $
              PATH=state.previousDirectory, $
              FILTER='*.tif' )
          print, imageFilename
          IF ( imageFilename EQ " ) THEN BREAK

          WRITE_TIFF, imageFilename, BYTE( ABS( *(state.currentImage) ) )

        END

      'Quit': $
        BEGIN
          HEAP_FREE, state
          HEAP_GC
          WHILE ( !D.WINDOW GE 0 ) DO WDELETE, !D.WINDOW
          CLOSE, /ALL
          WIDGET_CONTROL, Event.TOP, /DESTROY
          RETURN
        END

    ENDCASE

  END

'PulseSequece': $
  BEGIN

    CASE event.VALUE OF

      '90-FID Sequence': $
        BEGIN
```

```

        FID,state
    END

'Spin-Echo': $
    BEGIN
        SPIN_ECHO, state
    END

'Inversion Recovery': $
    BEGIN
        INVERSION_RECOVERY, state
    END

'Gradient recalled Echo': $
    BEGIN
        GRADIENT_RECALLED_ECHO, state
    END

ENDCASE

END

'Image': $
    BEGIN

    CASE event.VALUE OF

        'Real Image': $
            BEGIN

                imageFilename = $
                    DIALOG_PICKFILE( TITLE='Select Image File', $
                        PATH=state.previousDirectory )

                IF ( imageFilename EQ " ) THEN BREAK

                state.ImageLoad = 4

                Title = 'Real MR Image'

                LOAD_IMAGE_process, imageFilename,state

                state.previousDirectory = FILE_DIRNAME( imageFilename )

            END

        ENDCASE

    END

'Process': $
    BEGIN

    CASE event.VALUE OF

        'RMS(Root Mean Square)': $
            BEGIN
                RMS, state
            END

        ENDCASE

    ENDCASE

```

```

END

ENDCASE

WIDGET_CONTROL, event.TOP, SET_UVALUE=state, /NO_COPY

RETURN

END

PRO DIP_GUI

state = { mainEvent:-1L, $
  previousDirectory:", $
  ImageLoad:0L, $
  SystemType:0D, $
  lastImageLoaded:PTR_NEW( /ALLOCATE_HEAP ), $
  t1Image:PTR_NEW( /ALLOCATE_HEAP ), $
  t2Image:PTR_NEW( /ALLOCATE_HEAP ), $
  rImage:PTR_NEW( /ALLOCATE_HEAP ), $
  currentImage:PTR_NEW( /ALLOCATE_HEAP ), $
  ReallImage:PTR_NEW( /ALLOCATE_HEAP ) }

base = WIDGET_BASE( TITLE='MRISIG - Magnetic Resonance Image Synthetic Image Generator', $
  TLB_FRAME_ATTR=25, $
  XOFFSET=0, $
  YOFFSET=0, $
  XSIZE=800, $
  MBAR=menuBar, $
  /COLUMN )

WIDGET_CONTROL, base, SET_UVALUE=state

fileMenu = WIDGET_BUTTON( menuBar, $
  VALUE='File', $
  /MENU )

fileMenuEntries = $
[ '0\Load Image...', $
  '0\Save Image...', $
  '0\Quit' ]

file = $
CW_PDMENU( fileMenu, $
  fileMenuEntries, $
  /MBAR, $
  /RETURN_FULL_NAME, $
  UVALUE='File' )

PulseSequeceMenu = WIDGET_BUTTON( menuBar, $
  VALUE='Pulse Sequece', $
  /MENU )

PulseSequeceMenuEntries = $
[ '0\90-FID Sequence', $
  '0\Spin-Echo', $
  '0\Inversion Recovery', $
  '0\Gradient recalled Echo' ]

```

```

PulseSequece = $
  CW_PDMENU( PulseSequeceMenu, $
    PulseSequeceMenuEntries, $
    /MBAR, $
    /RETURN_FULL_NAME, $
    UVALUE='PulseSequece' )

ImageMenu = WIDGET_BUTTON( menuBar, $
  VALUE='Image', $
  /MENU )
ImageMenuEntries = $
[ '0\Real Image' ]

Image = $
  CW_PDMENU( ImageMenu, $
    ImageMenuEntries, $
    /MBAR, $
    /RETURN_FULL_NAME, $
    UVALUE='Image' )

ProcessMenu = WIDGET_BUTTON( menuBar, $
  VALUE='Process', $
  /MENU )
ProcessMenuEntries = $
[ '0\RMS(Root Mean Square)' ]

Process = $
  CW_PDMENU( ProcessMenu, $
    ProcessMenuEntries, $
    /MBAR, $
    /RETURN_FULL_NAME, $
    UVALUE='Process' )

WIDGET_CONTROL, base, /REALIZE

XMANAGER, 'DIP_GUI', base, EVENT_HANDLER='DIP_EVENT'

END

PRO MRISIG

  DIP_GUI

END

```

2. Load_Image.pro

```
PRO LOAD_IMAGE_event, event
```

```
WIDGET_CONTROL, event.TOP, GET_UVALUE=state
```

```
WIDGET_CONTROL, event.ID, GET_UVALUE=widget
```

```
CASE widget OF
```

```
'T1ImageButton' : $
```

```
  BEGIN
```

```
    imageFilename = $
```

```
    DIALOG_PICKFILE( TITLE='Select Image File' )
```

```
    T1_Textbox = $
```

```
      WIDGET_INFO( EVENT.top, $
```

```
        FIND_BY_UNAME='T1_Textbox' )
```

```
    WIDGET_CONTROL, T1_Textbox, SET_VALUE = imageFilename
```

```
    WIDGET_CONTROL, T1_Textbox, GET_VALUE=T1
```

```
    state.ImageLoad = 1
```

```
    draw_t1 = WIDGET_INFO( EVENT.top , $
```

```
      FIND_BY_UNAME='draw_T1' )
```

```
    WIDGET_CONTROL, draw_t1, GET_VALUE = draw_t11
```

```
    WSET, draw_t11
```

```
    LOAD_IMAGE_process, T1, state
```

```
WIDGET_CONTROL, (*state.callingState).mainEvent, $
```

```
  SET_UVALUE=(*state.callingState)
```

```
END
```

```
'T2ImageButton' : $
```

```
  BEGIN
```

```
    imageFilename = $
```

```
    DIALOG_PICKFILE( TITLE='Select Image File' )
```

```
    T2_Textbox = $
```

```
      WIDGET_INFO( EVENT.top,$
```

```
        FIND_BY_UNAME='T2_Textbox' )
```

```
    WIDGET_CONTROL, T2_Textbox, SET_VALUE = imageFilename
```

```
    WIDGET_CONTROL, T2_Textbox, GET_VALUE=T2
```

```
    state.ImageLoad = 2
```

```
    draw_t2 = WIDGET_INFO( EVENT.top , $
```

```
      FIND_BY_UNAME='draw_T2' )
```

```
    WIDGET_CONTROL, draw_t2, GET_VALUE = draw_t22
```

```
    WSET, draw_t22
```

```
    LOAD_IMAGE_process, T2, state
```

```
WIDGET_CONTROL, (*state.callingState).mainEvent, $
```

```
  SET_UVALUE=(*state.callingState)
```

```

        END

'rImageButton': $
    BEGIN
        imageFilename = $
            DIALOG_PICKFILE( TITLE='Select Image File' )
        r_Textbox = WIDGET_INFO( EVENT.top, $
            FIND_BY_UNAME='r_Textbox' )

        WIDGET_CONTROL, r_Textbox, SET_VALUE = imageFilename
        WIDGET_CONTROL, r_Textbox, GET_VALUE=r

        state.ImageLoad = 3

        draw_r = WIDGET_INFO( EVENT.top, $
            FIND_BY_UNAME='draw_r' )

        WIDGET_CONTROL, draw_r, GET_VALUE = draw_rr
        WSET, draw_rr

        LOAD_IMAGE_process, r, state

        WIDGET_CONTROL, (*state.callingState).mainEvent, $
            SET_UVALUE>(*state.callingState)

        END

'cancelButton': $
    BEGIN
        WIDGET_CONTROL, event.TOP, /DESTROY
        RETURN
    END

'checkbox1': $
    BEGIN
        state.SystemType = 0L
        WIDGET_CONTROL, event.TOP, SET_UVALUE=state
    END

'checkbox2': $
    BEGIN
        state.SystemType = 1L
        WIDGET_CONTROL, event.TOP, SET_UVALUE=state
    END

ENDCASE

END

PRO LOAD_IMAGE_gui,callingState

state = {TR:0.0D, $
    TE:0.0D, $
    ImageLoad:0.0D, $
    SystemType:0.0D, $
    callingState:PTR_NEW( callingState ) }

functionDialog = $
    WIDGET_BASE( COLUMN=1, $
        TLB_FRAME_ATTR=1, $
        TITLE='Load T1,T2 and r Image' )

```

WIDGET_CONTROL, functionDialog, SET_UVALUE=state

```
drawBase = $
  WIDGET_BASE( functionDialog, $
    COLUMN = 3, $
    /ALIGN_CENTER, $
    MAP=1, $
    FRAME=0 )

draw_T1 = $
  WIDGET_DRAW( drawBase, $
    UNAME='draw_T1', $
    UVALUE='draw_T1', $
    XSize=100, $
    YSize=100 )

draw_T1_Label = $
  WIDGET_Label( drawBase, $
    VALUE='T1')

draw_T2 = $
  WIDGET_DRAW( drawBase, $
    UNAME='draw_T2', $
    UVALUE='draw_T2', $
    XSize=100, $
    YSize=100 )

draw_T2_Label = $
  WIDGET_Label( drawBase, $
    VALUE='T2')

draw_r = $
  WIDGET_DRAW( drawBase,$
    UNAME='draw_r', $
    UVALUE='draw_r', $
    XSize=100, $
    YSize=100 )

draw_r_Label = $
  WIDGET_Label( drawBase, $
    VALUE='r')

fileSection = $
  WIDGET_BASE( functionDialog, $
    ROW=3, $
    /ALIGN_CENTER, $
    MAP=1, $
    FRAME=0 )

T1_Textbox = $
  WIDGET_TEXT( fileSection, $
    UNAME='T1_Textbox', $
    UVALUE='T1_Textbox', $
    XSIZE=30, $
    /EDITABLE )

T1_Label = $
  WIDGET_LABEL( fileSection, $
    VALUE='T1 Image' )

T1_button = $
  WIDGET_BUTTON( fileSection, $
    UNAME='T1ImageButton', $
```

```

        UVALUE='T1ImageButton', $
        Value ='Browse' )

T2_Textbox = $
    WIDGET_TEXT( fileSection, $
        UNAME='T2_Textbox', $
        UVALUE='T2_Textbox', $
        XSIZE=30, $
        /EDITABLE )
T2_Label = $
    WIDGET_LABEL( fileSection, $
        VALUE='T2 Image' )

T2_button = $
    WIDGET_BUTTON( fileSection, $
        UNAME='T2ImageButton', $
        UVALUE='T2ImageButton', $
        Value ='Browse' )

r_Textbox = $
    WIDGET_TEXT( fileSection, $
        UNAME='r_Textbox', $
        UVALUE='r_Textbox', $
        XSIZE=30, $
        /EDITABLE )
r_Label = $
    WIDGET_LABEL( fileSection, $
        VALUE='r Image' )
r_button = $
    WIDGET_BUTTON( fileSection, $
        UNAME='rImageButton', $
        UVALUE='rImageButton', $
        Value ='Browse' )

checkBoxBase = $
    WIDGET_BASE( functionDialog, ROW = 1,$
        /EXCLUSIVE, /ALIGN_CENTER, MAP=1, FRAME=0)

checkbox1 = WIDGET_BUTTON( checkBoxBase, $
    UNAME='checkbox1', $
    UVALUE='checkbox1', $
    VALUE='PC' )

checkbox2 = WIDGET_BUTTON( checkBoxBase, $
    UNAME='checkbox2', $
    UVALUE='checkbox2', $
    VALUE='UNIX/Mac' )

actionSection = $
    WIDGET_BASE( functionDialog, $
        ROW=1, $
        /ALIGN_CENTER, $
        MAP=1, $
        FRAME=0 )

cancelButton = $
    WIDGET_BUTTON( actionSection, $
        UNAME='cancelButton', $
        UVALUE='cancelButton', $
        VALUE='Done' )

```

```
WIDGET_CONTROL, /REALIZE, functionDialog
```

```
XMANAGER, 'load_image_gui', $  
    functionDialog, $  
    EVENT_HANDLER='load_image_event'
```

```
END
```

```
PRO LOAD_IMAGE, callingState
```

```
    LOAD_IMAGE_gui, callingState
```

```
END
```

3. Load_Image_process.pro

```
PRO LOAD_IMAGE_process, imageFilename, state

image = INTARR(256,256)

OPENR, LUN, imageFilename, /GET_LUN

READU, LUN, image

CLOSE, LUN

SizeInfo = SIZE(image)

systemtype = state.SystemType

IF ( systemtype EQ 1 ) THEN BEGIN

    image = SWAP_ENDIAN ( image )

ENDIF

IF ( state.ImageLoad EQ 1 ) THEN BEGIN

    (*state.callingState).t1Image = PTR_NEW( image )

    EXPAND, image, 100, 100, image

    TVSCL, ROTATE(image, 2)

ENDIF

IF ( state.ImageLoad EQ 2 ) THEN BEGIN

    (*state.callingState).t2Image = PTR_NEW( image )

    EXPAND, image, 100, 100, image

    TVSCL, ROTATE(image, 2), XSIZE = 100, YSIZE = 100

ENDIF

IF ( state.ImageLoad EQ 3 ) THEN BEGIN

    (*state.callingState).rImage = PTR_NEW( image )

    EXPAND, image, 100, 100, image

    TVSCL, ROTATE(image, 2), XSIZE = 100, YSIZE = 100

ENDIF

IF ( state.ImageLoad EQ 4 ) THEN BEGIN

    State.RealImage = PTR_NEW( image )

    SizeInfo = SIZE (image)

    WINDOW, !D.WINDOW+1,$
        /FREE,$
        XSIZE=SizeInfo[1],$
        YSIZE=SizeInfo[2],$
```

```
TITLE='Real Image'  
TVSCL, ROTATE(image, 2)  
ENDIF  
END
```

4. FID.pro

PRO FID_gui, callingState

```
state = {TR:0.0D, $
  callingState:PTR_NEW( callingState ) }
```

```
functionDialog = $
  WIDGET_BASE( COLUMN=1, $
    TLB_FRAME_ATTR=1, $
    TITLE='90-FID sequence' )
```

```
WIDGET_CONTROL, functionDialog, SET_UVALUE=state
```

```
parameterSection = $
  WIDGET_BASE( functionDialog, $
    ROW=2, $
    /ALIGN_CENTER, $
    MAP=1, $
    FRAME=0 )
TR_Textbox = $
  WIDGET_TEXT( parameterSection, $
    UNAME='TR_Textbox', $
    UVALUE='TR_Textbox', $
    XSIZE=10, $
    /EDITABLE )
TR_Label = $
  WIDGET_LABEL( parameterSection, $
    VALUE='TR (Repetition Time)' )
```

```
actionSection = $
  WIDGET_BASE( functionDialog, $
    ROW=1, $
    /ALIGN_CENTER, $
    MAP=1, $
    FRAME=0 )
```

```
applyButton = $
  WIDGET_BUTTON( actionSection, $
    UNAME='applyButton', $
    UVALUE='applyButton', $
    VALUE='Apply' )
```

```
cancelButton = $
  WIDGET_BUTTON( actionSection, $
    UNAME='cancelButton', $
    UVALUE='cancelButton', $
    VALUE='Cancel' )
```

```
WIDGET_CONTROL, /REALIZE, functionDialog
```

```
XMANAGER, 'FID_gui', $
  functionDialog, $
  EVENT_HANDLER='FID_event'
```

END

PRO FID, callingState

```

FID_gui, callingState
END

PRO FID_event, event

WIDGET_CONTROL, event.TOP, GET_UVALUE=state

WIDGET_CONTROL, event.ID, GET_UVALUE=widget

CASE widget OF

'applyButton': $
  BEGIN

    TR_Textbox = $
      WIDGET_INFO( event.TOP, $
        FIND_BY_UNAME='TR_Textbox' )
      WIDGET_CONTROL, TR_Textbox, GET_VALUE=TR

    state.TR = DOUBLE( TR[0] )

    FID_process, state

    WIDGET_CONTROL, event.TOP, SET_UVALUE=state, /DESTROY
    WIDGET_CONTROL, (*state.callingState).mainEvent, $
      SET_UVALUE=(*state.callingState)

  END

'cancelButton': $
  BEGIN
    WIDGET_CONTROL, event.TOP, /DESTROY
    RETURN
  END

ENDCASE

END

```

5. FID_process.pro

```
PRO FID_process, state

t1 =>(*state.callingState).t1image
t2 =>(*state.callingState).t2image
r =>(*state.callingState).rimage

systemtype = (*state.callingState).SystemType

TR = state.TR

FID_Image = FLOAT(r[*,*]) * (1 - exp(-TR/FLOAT(T1[*,*])))

SizeInfo = SIZE(FID_Image)

WINDOW, !D.WINDOW+1,$
  /FREE,$
  XSIZE=SizeInfo[1],$
  YSIZE=SizeInfo[2],$
  TITLE='90-FID pulse Image'

IF ( systemtype EQ 1 ) THEN BEGIN

  FID_Image = SWAP_ENDIAN ( FID_Image )
  TVSCL, ROTATE(FID_Image, 2)

ENDIF ELSE BEGIN

  TVSCL, ROTATE(FIX(FID_Image), 2)

ENDELSE

print, 'total = ', total (FIX(FID_Image))
PTR_FREE, (*state.callingState).lastImageLoaded
(*state.callingState).lastImageLoaded = $
  PTR_NEW (FIX(FID_image))

END
```

6. Gradient_Recall_Echo_process.pro

```
PRO GRADIENT_RECALLED_ECHO_process, state
```

```
t1 =>(*state.callingState).t1image  
t2 =>(*state.callingState).t2image  
r =>(*state.callingState).rimage
```

```
TR = state.TR  
DEGREE = state.DEGREE  
TE = state.TE
```

```
GRADIENT_RECALLED_ECHO_Image = FLOAT(r[*,*]) * (1 - EXP(-TR/FLOAT(T1[*,*]))) * $  
    sin(DEGREE * !pi/180) * EXP(-TE/FLOAT(T2[*,*])) / $  
    (1 - cos(DEGREE * !pi/180) * EXP(-TR/T1[*,*]))
```

```
SizeInfo = SIZE(GRADIENT_RECALLED_ECHO_Image)
```

```
WINDOW, !D.WINDOW+1,$  
    XSIZE=SizeInfo[1],$  
    YSIZE=SizeInfo[2],$  
    TITLE='GRADIENT_RECALLED_ECHO_Image (' + STRTRIM( STRING( !D.WINDOW+1 ), 2 ) + ')'
```

```
TV, GRADIENT_RECALLED_ECHO_Image
```

```
PTR_FREE, (*state.callingState).lastImageLoaded  
(*state.callingState).lastImageLoaded = PTR_NEW (GRADIENT_RECALLED_ECHO_Image)
```

```
END
```

7. Inversion_Recovery_process.pro

```
PRO INVERSION_RECOVERY_process, state
```

```
t1 =>(*state.callingState).t1image  
t2 =>(*state.callingState).t2image  
r =>(*state.callingState).rimage
```

```
TR = state.TR  
TI = state.TI
```

```
INVERSION_RECOVERY_Image = FLOAT(r[*,*]) * (1 - 2 * EXP(-TI/FLOAT(T1[*,*])) $  
+ EXP(-TR/FLOAT(T1[*,*])))
```

```
SizeInfo = SIZE(INVERSION_RECOVERY_Image)
```

```
WINDOW, !D.WINDOW+1,$  
XSIZE=SizeInfo[1],$  
YSIZE=SizeInfo[2],$  
TITLE='INVERSION_RECOVERY_Image (' + STRTRIM( STRING( !D.WINDOW+1 ), 2 ) + ')
```

```
TV, INVERSION_RECOVERY_Image  
PTR_FREE, (*state.callingState).lastImageLoaded  
(*state.callingState).lastImageLoaded = PTR_NEW (INVERSION_RECOVERY_Image)
```

```
END
```

8. Spin_Echo_process.pro

```
PRO SPIN_ECHO_process, state

t1 =>(*state.callingState).t1image
t2 =>(*state.callingState).t2image
r =>(*state.callingState).rimage

systemtype = (*state.callingState).SystemType

TR = state.TR
TE = state.TE

SPIN_ECHO_Image = FLOAT(r[*,*]) * (1 - exp(-TR/FLOAT(T1[*,*]))) $
    * exp(-TE/FLOAT(T2[*,*]))

SizeInfo = SIZE(SPIN_ECHO_Image)

WINDOW, !D.WINDOW+1,$
    /FREE,$
    XSIZE=SizeInfo[1],$
    YSIZE=SizeInfo[2],$
    TITLE='SPIN_ECHO_Image'

IF ( systemtype EQ 1 ) THEN BEGIN

    SPIN_ECHO_Image = SWAP_ENDIAN ( SPIN_ECHO_Image )
    TVSCL, ROTATE(SPIN_ECHO_Image, 2)

ENDIF ELSE BEGIN

    TVSCL, ROTATE(FIX(SPIN_ECHO_Image), 2)

ENDELSE

print, 'total = ', total (FIX(SPIN_ECHO_Image))
PTR_FREE, (*state.callingState).lastImageLoaded
(*state.callingState).lastImageLoaded = $
    PTR_NEW (FIX(SPIN_ECHO_image))
(*state.callingstate).currentimage = PTR_NEW(SPIN_ECHO_Image)
write_tiff, 'image1', tvrd()
END
```

9. Gradient_Recall_echo.pro

```
PRO GRADIENT_RECALLED_ECHO_gui, callingState

state = {TR:0.0D, $
        TE:0.0D, $
        DEGREE:0.0D, $
        callingState:PTR_NEW( callingState ) }

functionDialog = $
WIDGET_BASE( COLUMN=1, $
            TLB_FRAME_ATTR=1, $
            TITLE='Gradient Recalled Echo pulse sequence' )

WIDGET_CONTROL, functionDialog, SET_UVALUE=state

parameterSection = $
WIDGET_BASE( functionDialog, $
            ROW=3, $
            /ALIGN_CENTER, $
            MAP=1, $
            FRAME=0 )
TR_Textbox = $
WIDGET_TEXT( parameterSection, $
            UNAME='TR_Textbox', $
            UVALUE='TR_Textbox', $
            XSIZE=10, $
            /EDITABLE )
TR_Label = $
WIDGET_LABEL( parameterSection, $
            VALUE='TR (Repetition Time)' )

TE_Textbox = $
WIDGET_TEXT( parameterSection, $
            UNAME='TE_Textbox', $
            UVALUE='TE_Textbox', $
            XSIZE=10, $
            /EDITABLE )
TE_Label = $
WIDGET_LABEL( parameterSection, $
            VALUE='TE (Echo Time)' )

DEGREE_Textbox = $
WIDGET_TEXT( parameterSection, $
            UNAME='DEGREE_Textbox', $
            UVALUE='DEGREE_Textbox', $
            XSIZE=10, $
            /EDITABLE )
DEGREE_Label = $
WIDGET_LABEL( parameterSection, $
            VALUE='Degree' )

actionSection = $
WIDGET_BASE( functionDialog, $
            ROW=1, $
            /ALIGN_CENTER, $
            MAP=1, $
            FRAME=0 )

applyButton = $
WIDGET_BUTTON( actionSection, $
            UNAME='applyButton', $
```

```

        UVALUE='applyButton', $
        VALUE='Apply' )

cancelButton = $
    WIDGET_BUTTON( actionSection, $
        UNAME='cancelButton', $
        UVALUE='cancelButton', $
        VALUE='Cancel' )

WIDGET_CONTROL, /REALIZE, functionDialog

XMANAGER, 'GRADIENT_RECALLED_ECHO_gui', $
    functionDialog, $
    EVENT_HANDLER='GRADIENT_RECALLED_ECHO_event'

END

PRO GRADIENT_RECALLED_ECHO, callingState

    GRADIENT_RECALLED_ECHO_gui, callingState

END

PRO GRADIENT_RECALLED_ECHO_event, event

    WIDGET_CONTROL, event.TOP, GET_UVALUE=state

    WIDGET_CONTROL, event.ID, GET_UVALUE=widget

    CASE widget OF

        'applyButton': $
            BEGIN

                TR_Textbox = $
                    WIDGET_INFO( event.TOP, $
                        FIND_BY_UNAME='TR_Textbox' )
                    WIDGET_CONTROL, TR_Textbox, GET_VALUE=TR

                TE_Textbox = $
                    WIDGET_INFO( event.TOP, $
                        FIND_BY_UNAME='TE_Textbox' )
                    WIDGET_CONTROL, TE_Textbox, GET_VALUE=TE

                DEGREE_Textbox = $
                    WIDGET_INFO( event.TOP, $
                        FIND_BY_UNAME='DEGREE_Textbox' )
                    WIDGET_CONTROL, DEGREE_Textbox, GET_VALUE=DEGREE

                state.TR = DOUBLE( TR[0] )

                state.TE = DOUBLE( TE[0] )

                state.DEGREE = DOUBLE( DEGREE[0] )

                GRADIENT_RECALLED_ECHO_process, state

```

```
        WIDGET_CONTROL, event.TOP, SET_UVALUE=state
        WIDGET_CONTROL, (*state.callingState).mainEvent, $
            SET_UVALUE=(*state.callingState)
    END

    'cancelButton': $
    BEGIN
        WIDGET_CONTROL, event.TOP, /DESTROY
    RETURN
    END

ENDCASE

END
```

10. Inversion_Recovery.pro

```
PRO INVERSION_RECOVERY_gui, callingState
```

```
state = {TR:0.0D, $  
        TI:0.0D, $  
        callingState:PTR_NEW( callingState ) }
```

```
functionDialog = $  
  WIDGET_BASE( COLUMN=1, $  
    TLB_FRAME_ATTR=1, $  
    TITLE='Inversion Recovery pulse sequence' )
```

```
WIDGET_CONTROL, functionDialog, SET_UVALUE=state
```

```
parameterSection = $  
  WIDGET_BASE( functionDialog, $  
    ROW=2, $  
    /ALIGN_CENTER, $  
    MAP=1, $  
    FRAME=0 )
```

```
TR_Textbox = $  
  WIDGET_TEXT( parameterSection, $  
    UNAME='TR_Textbox', $  
    UVALUE='TR_Textbox', $  
    XSIZE=10, $  
    /EDITABLE )
```

```
TR_Label = $  
  WIDGET_LABEL( parameterSection, $  
    VALUE='TR (Repetition Time)' )
```

```
TI_Textbox = $  
  WIDGET_TEXT( parameterSection, $  
    UNAME='TI_Textbox', $  
    UVALUE='TI_Textbox', $  
    XSIZE=10, $  
    /EDITABLE )
```

```
TI_Label = $  
  WIDGET_LABEL( parameterSection, $  
    VALUE='TI (Inversion Time)' )
```

```
actionSection = $  
  WIDGET_BASE( functionDialog, $  
    ROW=1, $  
    /ALIGN_CENTER, $  
    MAP=1, $  
    FRAME=0 )
```

```
applyButton = $  
  WIDGET_BUTTON( actionSection, $  
    UNAME='applyButton', $  
    UVALUE='applyButton', $  
    VALUE='Apply' )
```

```
cancelButton = $  
  WIDGET_BUTTON( actionSection, $  
    UNAME='cancelButton', $  
    UVALUE='cancelButton', $  
    VALUE='Cancel' )
```

```

WIDGET_CONTROL, /REALIZE, functionDialog

XMANAGER, 'INVERSION_RECOVERY_gui', $
    functionDialog, $
    EVENT_HANDLER='INVERSION_RECOVERY_event'

END

PRO INVERSION_RECOVERY, callingState

    INVERSION_RECOVERY_gui, callingState

END

PRO INVERSION_RECOVERY_event, event

    WIDGET_CONTROL, event.TOP, GET_UVALUE=state
    WIDGET_CONTROL, event.ID, GET_UVALUE=widget

CASE widget OF

    'applyButton': $
        BEGIN

            TR_Textbox = $
                WIDGET_INFO( event.TOP, $
                    FIND_BY_UNAME='TR_Textbox' )
            WIDGET_CONTROL, TR_Textbox, GET_VALUE=TR

            TI_Textbox = $
                WIDGET_INFO( event.TOP, $
                    FIND_BY_UNAME='TI_Textbox' )
            WIDGET_CONTROL, TI_Textbox, GET_VALUE=TI

            state.TR = DOUBLE( TR[0] )

            state.TI = DOUBLE( TI[0] )

            INVERSION_RECOVERY_process, state

            WIDGET_CONTROL, event.TOP, SET_UVALUE=state
            WIDGET_CONTROL, (*state.callingState).mainEvent, $
                SET_UVALUE=(*state.callingState)
        END

    'cancelButton': $
        BEGIN
            WIDGET_CONTROL, event.TOP, /DESTROY
            RETURN
        END

ENDCASE

END

```

11. Spin_Echo.pro

PRO SPIN_ECHO_gui, callingState

```
state = {TR:0.0D, $
        TE:0.0D, $
        callingState:PTR_NEW( callingState ) }
```

```
functionDialog = $
WIDGET_BASE( COLUMN=1, $
            TLB_FRAME_ATTR=1, $
            TITLE='Spin Echo pulse sequence' )
```

WIDGET_CONTROL, functionDialog, SET_UVALUE=state

```
parameterSection = $
WIDGET_BASE( functionDialog, $
            ROW=2, $
            /ALIGN_CENTER, $
            MAP=1, $
            FRAME=0 )
TR_Textbox = $
WIDGET_TEXT( parameterSection, $
            UNAME='TR_Textbox', $
            UVALUE='TR_Textbox', $
            XSIZE=10, $
            /EDITABLE )
TR_Label = $
WIDGET_LABEL( parameterSection, $
            VALUE='TR (Repetition Time)' )
```

```
TE_Textbox = $
WIDGET_TEXT( parameterSection, $
            UNAME='TE_Textbox', $
            UVALUE='TE_Textbox', $
            XSIZE=10, $
            /EDITABLE )
TE_Label = $
WIDGET_LABEL( parameterSection, $
            VALUE='TE (Echo Time)' )
```

```
actionSection = $
WIDGET_BASE( functionDialog, $
            ROW=1, $
            /ALIGN_CENTER, $
            MAP=1, $
            FRAME=0 )
```

```
applyButton = $
WIDGET_BUTTON( actionSection, $
            UNAME='applyButton', $
            UVALUE='applyButton', $
            VALUE='Apply' )
```

```
cancelButton = $
WIDGET_BUTTON( actionSection, $
            UNAME='cancelButton', $
            UVALUE='cancelButton', $
            VALUE='Cancel' )
```

WIDGET_CONTROL, /REALIZE, functionDialog

```

XMANAGER, 'SPIN_ECHO_gui', $
    functionDialog, $
    EVENT_HANDLER='SPIN_ECHO_event'

END

PRO SPIN_ECHO, callingState
    SPIN_ECHO_gui, callingState

END

PRO SPIN_ECHO_event, event
    WIDGET_CONTROL, event.TOP, GET_UVALUE=state
    WIDGET_CONTROL, event.ID, GET_UVALUE=widget

CASE widget OF

'applyButton': $
    BEGIN

        TR_Textbox = $
            WIDGET_INFO( event.TOP, $
                FIND_BY_UNAME='TR_Textbox' )
            WIDGET_CONTROL, TR_Textbox, GET_VALUE=TR

        TE_Textbox = $
            WIDGET_INFO( event.TOP, $
                FIND_BY_UNAME='TE_Textbox' )
            WIDGET_CONTROL, TE_Textbox, GET_VALUE=TE

        state.TR = DOUBLE( TR[0] )

        state.TE = DOUBLE( TE[0] )

        SPIN_ECHO_process, state

        WIDGET_CONTROL, event.TOP, SET_UVALUE=state, /DESTROY
        WIDGET_CONTROL, (*state.callingState).mainEvent, $
            SET_UVALUE=(*state.callingState)

    END

'cancelButton': $
    BEGIN
        WIDGET_CONTROL, event.TOP, /DESTROY
    RETURN
    END

ENDCASE

END

```

12. Result.pro

PRO Result_event, event

WIDGET_CONTROL, event.TOP, GET_UVALUE=state

WIDGET_CONTROL, event.ID, GET_UVALUE=widget

CASE widget OF

'cancelButton': \$

BEGIN

WIDGET_CONTROL, event.TOP, /DESTROY

RETURN

END

ENDCASE

END

PRO Result_gui,difference_previous,k

mainWindow = \$

WIDGET_BASE(COLUMN=1, \$

TLB_FRAME_ATTR=0, \$

TITLE='Result')

fileSection = \$

WIDGET_BASE(mainWindow, \$

ROW=2, \$

/ALIGN_CENTER, \$

MAP=1, \$

FRAME=1)

fileLabel = \$

WIDGET_LABEL(fileSection, \$

VALUE='RMS (Root Mean Square)')

fileTextbox1 = \$

WIDGET_TEXT(fileSection, \$

UNAME='fileTextbox1', \$

UVALUE='fileTextbox1', \$

XSIZE=10)

fileLabe2 = \$

WIDGET_LABEL(fileSection, \$

VALUE='k (Scaling Factor)')

fileTextbox2 = \$

WIDGET_TEXT(fileSection, \$

UNAME='fileTextbox2', \$

UVALUE='fileTextbox2', \$

XSIZE=10)

difference_previous = \$

STRTRIM(STRING(difference_previous), 2)

WIDGET_CONTROL, fileTextbox1, \$

SET_VALUE=difference_previous

k = STRTRIM(STRING(k), 2)

WIDGET_CONTROL, fileTextbox2,SET_VALUE = k

```

actionSection = $
    WIDGET_BASE( mainWindow, $
        ROW=1, $
        /ALIGN_CENTER, $
        MAP=1, $
        FRAME=0 )

cancelButton = $
    WIDGET_BUTTON( actionSection, $
        UNAME='cancelButton', $
        UVALUE='cancelButton', $
        VALUE='Close' )

WIDGET_CONTROL, mainWindow, /REALIZE

XMANAGER, 'Result_gui', $
    mainWindow, $
    EVENT_HANDLER='Result_event'
END

```

```

PRO Result, difference_later,k, state

    Result_gui,difference_later,k

END

```

13. RMS.pro

```
PRO RMS_process, callingState
```

```
state = {difference:0.0D, $  
callingState:PTR_NEW( callingState ) }
```

```
Final_Image =>(*state.callingState).t1 image
```

```
Real_Image =>(*state.callingState).RealImage
```

```
help, Final_Image,Real_Image
```

```
difference = state. difference
```

```
K = 0.0001
```

```
difference_previous = SQRT(TOTAL((K * Final_Image - Real_Image)^2))
```

```
difference_later = SQRT(TOTAL(((K+0.001) * Final_Image - Real_Image)^2))
```

```
WHILE ( difference_previous GE difference_later) DO BEGIN
```

```
difference_previous = difference_later
```

```
K = K+0.01
```

```
difference_later = SQRT(TOTAL((K * Final_Image - Real_Image)^2))
```

```
ENDWHILE
```

```
Result, difference_previous, k, state
```

```
END
```

```
PRO RMS, callingState
```

```
RMS_process, callingState
```

```
END
```