拡散テンソル磁気共鳴画像法において脳内水分子拡散異方性に寄与する要因について

課題番号　　13670930

平成 13 年度～平成 14 年度科学研究費補助金(基盤研究(C)(2))研究成果報告書

平成 16 年 3 月

研究代表者　井藤　隆太
(滋賀医科大学医学部講師)

研究組織

研究代表者：井藤　隆太　(滋賀医科大学医学部講師)

(研究協力者：森　進　　(ジョンスホプキンス大学医学部助教授))

交付決定額 (配分額)　　　　　　　　　　　　　　　　(金額単位：千円)

	直接経費	間接経費	合計
平成 13 年度	1,800		1,800
平成 14 年度	1,700		1,700
総計	3,500		3,500

研究発表 (口頭発表)

Itoh R, Melhem ER, Murata K, et al: Evaluation of Human Brain White Matter Integrity using Diffusion Tensor MR Histograms [abstract 1162]. In Proceeding of International Society for Magnetic Resonance in Medicine, Honolulu, Hawaii, USA, 1162, May, 2002

Ito R, Kitahara S, Hayakawa S, Sho K, Nakasu S, Mori S, Murata K.　Evaluation of Radiation-Induced Cerebral White Matter Injury Using Diffusion Tensor MR Imaging: Initial Experience. American Society of Neuroradology 41st Annual Meeting, April 26-May 2, 2003 at the Marriott Wardman Park Hotel in Washington, DC, USA.

脳の拡散テンソル(Diffusion-Tensor)磁気共鳴(MR)画像法は、脳組織内の水分子の拡散を非侵襲的に定量測定する事により、脳の発達、加齢、病変に伴った組織構築の変化を評価可能な手段として提案された(1)。

MR画像法は従来、コントラストとしてプロトン密度、T1緩和やT2緩和の差、流れの情報などを画像化してきたが、臨床医学の場でよく用いられているT1緩和やT2緩和効果を画像化した場合の信号強度は様々な因子が複雑に反映された結果であり、これらの画像情報から組織内に生じている病理学的変化や微細な構造変化を特定する事は容易ではなかった。

拡散MR画像法は、組織中水分子の熱運動や衝突(ブラウン運動)による方向と大きさが無秩序な動きで、一定時間後に微粒子(水分子)の存在する可能性のある空間的な範囲がガウス分布に従っているような微視的な動きを信号変化として取り出す。この水分子の微視的な動きは、組織における微小構造及び細胞外液、細胞質などの媒質の水分量、粘調度、温度といった生理学的状態により修飾されると考えられており、拡散MR画像法は、この水分子拡散を撮像voxel単位で測定することで水分子をprobeとして間接的に組織の微小構造、生理学的状態を解析、評価しようとするものである。

脳組織は大きくは灰白質と白質とに分けられる。灰白質は主に神経細胞が密に存在し不規則な構造を持つのに対し、白質は主に神経線維とその支持組織からなり規則正しい配列構造を有することで両者の微小構造は特徴づけられる。規則正しい配列構造による方向性を有する白質内においては、配列構造に垂直方向への水分子の動きが妨げられることにより一定時間後の水分子の存在する可能性のある領域は楕円体で表現されると考えられ、このような特徴を持つ拡散は異方性拡散(anisotropic diffusion)と呼ばれる(fig.1a)。一方、灰白質のような特に方向性を持たない構造内では、すべての方向への水分子の動きが等しい確率で起こりえることから、ある一定時間後の水分子の存在する可能性のある領域は球で表現されると考えられ、このような特徴を持つ拡散は等方性拡散(isotropic diffusion)と呼ばれる(fig.1b)。

脳の拡散テンソルMR画像法は、脳内の水分子拡散が組織の微少構造及び生理的状況の影響を受けその拡がりが楕円体(球状の場合を包括した)で表現できると仮定し、MR画像法を使い撮像voxel単位でその楕円体の形や大きさを決定することで、脳組織内微小構造や生理学的状態を定量的に解析しようとする手法である(1,2)。

拡散テンソルMR画像法によって得られる脳白質内異方性拡散の程度を示す指標を測定することで、多発性硬化症をはじめとした脳白質病変において脳白質の損傷程度の評価を試みたり(3,4)、成長及び加齢に伴う脳白質の成熟、変性の程度の定量的評価が試みられている(5,6)。しかし、脳白質内のどのような構造がこの異方性に寄与しているかについてはすべてが明確になっている訳ではない。

ヒト新生児期の脳白質内水分子拡散異方性の増加が髄鞘化と時期的な関連を示す事から、白質内髄鞘構造がその異方性に大きく寄与する事が考えられている他(7)、軸索及びその走行形態の寄与も示唆されている(8)。また成年のmouseには認められない脳灰白質内の異方性が出生前後の時期の脳では認められる事が観察されており(9)、神経細胞の樹状突起の発育がこの時期に非常に顕著である事から、樹状突起の無方向性の繁茂が異方性の減少に寄与しているのではないかと予想される。

本研究では、出生前後の正常ratの灰白質内異方性を表す指標を測定し、組織標本を作成し比較検討する事により異方性に寄与する因子(特に軸索と樹状突起の関与について)を明らかにするために必要な実験用MR装置を用いた撮像法の確立、画像解析用アプリケーションの作成、出生前後の正常rat脳標本拡散テンソルMR画像の撮像及び脳組織標本を用いた樹状突起の発育程度評価の方法についての検討を行った。

Theory (10)

Determination of the shape of an ellipsoid

The shape of an anisotropic diffusion ellipsoid can be characterized by six parameters; three magnitudes of diffusivity along three principal orthogonal coordinate axes of the ellipsoid and three angles (Euler angles) between the principal coordinate axes and the laboratory coordinate axes (i.e. coordinates of a magnet) (fig. 2). The parameters can be derived from Diffusion-Tensor MR imaging at a macroscopic voxel scale (1, 2).

How to Measure Diffusion Process

The diffusion process can be measured by using a magnetic field gradient. In the presence of the magnetic field gradient, water molecules with random displacements (described as incoherent motion) acquire random phase shifts, resulting in an attenuation of the spin-echo signal due to refocusing failure (11).

Because of the Gaussian shape of the probability distribution of diffusion displacements, the attenuation is described as:

$$S/S_0 = \exp(-bD),$$

where S and S_0 are signal intensities sampled with and without the magnetic field gradient, respectively, and D is the diffusion coefficient (12).

Stejskal-Tanner gradient pulse

For diffusion sensitization, we apply a pair of gradient pulses arranged on each side of the 180° radio-frequency (RF) pulse in a spin-echo sequence (fig. 3) (13). The b-value, which reflects the degree of diffusion sensitization, is determined by the gyromagnetic ratio (γ), the strength of the gradients (with an ideal rectangular shape) (G), the duration of each gradient pulse (δ), and the time interval between their onsets (Δ) according to:

$$b = \gamma^2 G^2 \delta^2 (\Delta - \delta/3).$$

In this case, the b factor can be controlled by only parameters of the applied field gradients (i.e. diffusion sensitizing gradients). The spin-echo method with the "Stejskal-Tanner" gradient pulse is commonly used for "diffusion-weighted" MR imaging (14).

Apparent Diffusion Coefficient

When the attenuation is measured from the series of images with different known magnitudes of b-value (at least two different b-values), the diffusion coefficient D in each image voxel can be calculated. Since there are not only pure diffusion of water molecules but also other incoherent motions (e.g. micro-circulation of blood in the capillary network, laminar or turbulent flow within vessels or cerebrospinal fluid space) in the image voxel, what we can measure is termed an "apparent" diffusion coefficient (ADC) (15). The contribution of the other incoherent motions to the diffusion coefficient can be reduced by applying diffusion sensitizing gradient with larger b-values (15, 16).

The apparent diffusion coefficient along the direction of the applied diffusion sensitizing gradient can be measured. The diffusion sensitization is implemented using one of the gradient devices equipped in x, y, and z directions of a magnet or by combining them, being applied in any desired direction (1, 2). Consequently, apparent diffusion coefficients in any desired directions can be measured.

Effective Diffusion Tensor D^{eff}

In order to estimate anisotropic water diffusion in white matter, the properties of the diffusion ellipsoid (determined by six parameters) should be characterized. The anisotropic diffusion process in each voxel can be characterized by the effective diffusion tensor ($\mathbf{D}^{eff}$), which is a 3 x 3 tensor matrix that consists of nine elements (1, 2).

$$\mathbf{D}^{eff} = \begin{pmatrix} D^{eff}xx & D^{eff}xy & D^{eff}xz \\ D^{eff}yx & D^{eff}yy & D^{eff}yz \\ D^{eff}zx & D^{eff}zy & D^{eff}zz \end{pmatrix}$$

Because the effective diffusion tensor is symmetric for Gaussian diffusion, it is characterized by six scalars: three diagonal elements, $D^{eff}xx$, $D^{eff}yy$, $D^{eff}zz$ and three off-diagonal elements, $D^{eff}xy$, $D^{eff}xz$, $D^{eff}yz$.

By sampling signal attenuation after applying diffusion sensitizing gradient in at least six different non-collinear directions, these six elements can be determined. In other words, the effective diffusion tensor is completed by calculating the apparent diffusion coefficients along at least six non-collinear directions (2). By diagonalizing the effective diffusion tensor, the principal orthogonal coordinate system of the ellipsoid can be constructed (1, 17). This procedure (diagonalization) means rotating the laboratory coordinate system, in which the apparent diffusion coefficients are measured, and adjusting it to the principal coordinate system of the ellipsoid. In this calculation process, three eigenvectors ($\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$), which are intrinsic to the tissue in a voxel, are obtained (1). Their magnitudes (eigenvalues (λ_1, λ_2, λ_3)) represent diffusivities along three principal coordinate axes of the ellipsoid. The eigenvectors have also information of the angles between the principal coordinate axes and the laboratory coordinate axes (fig. 2). From the parameters of the ellipsoid, quantitative indices, which may reflect physiological and structural features of tissues in the voxel can be derived (1, 17, 18, 19, 20). The orientation of fiber tracts in white matter can be also determined by estimating the relationship between eigenvectors of adjacent voxels (1, 2).

Mapping the diffusion process in a voxel by estimating the effective diffusion tensor is called Diffusion-Tensor MR imaging (1).

Basics of Spatial Mapping of Diffusion Process

For spatial mapping of diffusion process, a total of at least seven images, which include diffusion-weighted MR images with the diffusion sensitizing gradients applied in six different non-collinear directions and an image with the same parameters as the diffusion-weighted images but no diffusion sensitization (fig. 4), should be obtained. From the difference in attenuation of signal intensity between non-diffusion-weighted and diffusion-weighted images, six voxel-by-voxel maps of apparent diffusion coefficient are generated. The effective diffusion tensor in each voxel is constructed from these maps. Once the tensor is determined, various Diffusion-Tensor MR imaging based maps can be generated.

Correction Techniques for Geometric Distortion

In DT MR imaging, misregistration affects the spatial resolution and accuracy of diffusion and anisotropy maps calculated on a pixel-by-pixel basis (17, 21, 22). Main causes of the misregistration originate in geometric distortions of each image and misalignment by small motions between sampled images. The application of correction algorithms is proposed.

Geometric distortions are the result of the static field inhomogeneity caused by imperfect shimming and by differences in magnetic properties of adjacent tissues and the result of diffusion sensitization gradient-induced eddy currents (21, 22). For the correction of diffusion sensitization gradient-induced eddy currents distortions, the application of least squares straight line fits and cross-correlation functions to the diffusion-weighted image has been introduced (21).

Quantitative Indices.

Several indices characterizing the diffusion process in tissues can be calculated based on formulas that incorporate the eigenvalues to generate quantitative brain maps. Scalar quantities such as spin density, T1 and T2 relaxation times cannot characterize specific pathophysiological states in tissues. In contrast to these parameters, quantitative indices derived from Diffusion-Tensor MR imaging are thought to be able to measure distinct intrinsic features of water diffusion in tissues, which are determined by microstructural (anatomical and histological) and physiological state of the tissues, although the determinants of water diffusion in tissues are still not completely understood. These intrinsic measurements allow monitoring brain maturation, changes by disease progression, and changes by interventions. They are essential for normalization across subjects and MR scanners. They may also help better define thresholds for segmentation algorithms. The ability to normalize becomes critical for conducting multi-center trials (necessary for uncommon diseases).

The most fundamental quantitative measures are the three principal diffusivities (eigenvalues) of the effective diffusion tensor ($\mathbf{D}^{eff}$), which are the principal diffusion coefficients measured along the three principal coordinates of the ellipsoid in each voxel. The trace of $\mathbf{D}^{eff}$ (trace ($\mathbf{D}^{eff}$)) is the basis for the directionally averaged (or isotropic, mean) diffusivity $D^{eff}_{ave} = (D^{eff}xx + D^{eff}yy + D^{eff}zz)/3$. Several indices to measure the degree of diffusion anisotropy in tissues have been proposed. The most intuitive and simplest indices are ratios of the principal diffusivities (1, 20), such as the dimensionless anisotropy ratio λ_h/λ_l or $2\lambda_h/(\lambda_m+\lambda_l)$, which measure the relative magnitudes of the diffusivities along the longest axis of the diffusion ellipsoid and other orthogonal axes. Here, the eigenvalues are sorted in order of decreasing magnitude (λ_h= highest diffusivity, λ_m= intermediate diffusivity, and λ_l lowest diffusivity). Since simulations have revealed that these sorted indices are statistically biased by noise contamination (20, 23, 24), fractional anisotropy (FA) (20) is one of un-sorted indices. The FA, the ratio of the anisotropic component of the diffusion tensor to the whole diffusion tensor, is defined as

$FA = (3/2)^{1/2} ((\lambda_1- D^{eff}_{ave})^2 + (\lambda_2- D^{eff}_{ave})^2 + (\lambda_3- D^{eff}_{ave})^2)^{1/2} / (\lambda_1^2+ \lambda_2^2+ \lambda_3^2)^{1/2}$.

For a complete isotropic medium, FA = 0; for a cylindrically symmetric anisotropic medium (with $\lambda_1 >> \lambda_2 = \lambda_3$), FA = 1.

Graphical Maps.

In white matter, water diffusion is anisotropic and restricted by highly ordered structures such as neural fiber bundles. Water molecules can move more easily in a direction parallel to the fiber bundles than in a direction perpendicular to them. By estimating a magnitude and direction of diffusion process using Diffusion-Tensor MR imaging, the regional fiber structures in white matter can be revealed. Several approaches that picture this directional information of the local fiber structures have been reported. A color map (fig. 5) (25)' is one of these graphical techniques.

結果

実験用超伝導2.0テスラMRスペクトロメーターを使用したrat脳3次元拡散MR画像撮像方法の確立

(1) ratの脳標本画像撮像用コイルの作成

当初コイル径50mm x 60mm長の既存homemadeコイルを用いて、径約7-8mm、5mm厚のrat脳半球標本撮像を行ったが、得られた画像の信号雑音比(SNR)が信頼できるFractional Anisotropy (FA)値を得るために十分でなく、必要なSNRを有する画像を得るために径12mm x 18mm長の円筒型のコイルを作成した(fig.6)。その結果、撮像条件によっては15-17程度のSNR値が得られるようになり、必要とされるSNR値20に近づいた。

(2) 3次元拡散テンソルMR画像撮像プログラム

実験用超伝導2.0テスラMRスペクトロメーター(OMEGA, General Electric NMR Instruments, Fremont, CA, USA)の基本撮像プログラムusaから撮像matrixとして256 x 128 x 32の3-dimensional spin-echo diffusion weighted image撮像のためのプログラムを作成した(APPENDIX 1a)。MRスペクトロメーターのメモリサイズの制限から256 x 128 x 32の収集データを保持不能であったため、phase stepを変化させた256 x 128の2-dimensional imageを32回収集した後、画像再構成時に3方向にフーリエ変換し、さらに2-dimensional image のphase方向とそれらを重ね合わせる厚み方向のデータをzero-fillingすることにより画像再構成matrixとして256 x 256 x 64の画像撮像を可能とした(APPENDIX 1b)。

拡散テンソル計算画像作成プログラムの開発

(1) diffusion sensitization gradient-induced eddy currentによる画像歪み

補正用プログラムの作成

Theoryの項で述べた拡散検出用傾斜磁場を付加した際にeddy currentにより生じる傾斜磁場を付加する方向によって変化する画像歪みを補正するために、Haselgrovrらの方法(21)をもとにしたアプリケーションをプログラム開発用ツール(IDL; Interactive Data Language: Research Systems Inc, Boulder, CO, USA)を用いて作成した(APPENDIX 2)(fig.7)。

(2) テンソル計算用プログラムの作成

Theoryの項で述べたBasserらの方法(1, 2)をもとに、拡散検出用傾斜磁場を付加しない画像と拡散検出用磁場を6方向(x,y,z)=(1,0,1), (-1,0,1), (0,1,1), (0,1,-1), (1,1,0), (-1,1,0)付加した画像からtensor matrixを計算しeigenvectors (v_1, v_2, v_3)とeigenvalues (λ_1, λ_2, λ_3)を得た後、それらからisotropic diffusivity (iADC)値 (mm^2/sec)とFA値をpixel毎にmappingした計算画像を作成するアプリケーションをプログラム開発用ツール(IDL)を用いて作成した(APPENDIX 3)。

(3) カラーマップ表示用プログラムの作成

Pajevicらの方法(25)をもとに、テンソル計算用プログラムを用いて得られたeigenvectors のなかで最大のものとFA値からpixel毎に異方性の強さを信号の強さ、異方性の方向を赤、緑、青の合成により表現するカラーマップ表示用プログラムをプログラム開発用ツール(IDL)を用いて作成した(APPENDIX 4)。

正常rat脳標本の3D拡散テンソルMR画像撮像

pentobarbitalを腹腔内投与し深麻酔科に生理的食塩水と0.1mol/Lのphosphate-buffered 4% paraformaldehydeを用いて経心還流固定したSprague-Dawley rat脳標本から5mm厚の切片を取り出し撮像に用いた。
3次元拡散テンソルMR画像は実験用超伝導2.0テスラMRスペクトロメーターと前出のhomemadeコイルを用いて撮像した。撮像matrixは256x128x32でzero-fillingにより再構成画像256x256x64を得た。最小のfield of viewは12x12x6mmで再構成画像の空間分解能は47 x 47 x 94μmであった。拡散テンソルMR画像撮像条件はrepetition time 950msec、echo time 50msec、8回加算で拡散検出磁場をかけない画像とb-valueが653.4sec/mm^2の拡散検出傾斜磁場を6

方向(x, y, z)=(1, 0, 1), (-1, 0, 1), (0, 1, 1), (0, 1, -1), (1, 1, 0), (-1, 1, 0)に付加した画像の7セットの画像データを得た(fig. 8)。

正常rat脳組織標本の作成

生後0日、3日、14日のratの脳を取り出し、それらのGFAP(glial fibrillary acidic protein)及びneurofilmentに対する免疫組織標本を作成し脳灰白質内の異方性に関与すると考えられる構造を描出した(fig. 9)。

考察

本研究で開発した実験用MR装置を用いてのrat脳3次元拡散MR画像撮像方法及び拡散テンソル計算画像作成プログラムにより得られるrat脳標本の3D拡散テンソルMR画像と組織標本によって描出される脳灰白質内の線維構造を比較検討することで、脳内の異方性に寄与する因子とそれらが異方性に寄与する程度が明らかにされれば、拡散テンソルMR画像法により生体の脳から非侵襲的に得られた異方性の指標を用いる事で、脳微細構造(髄鞘、軸索、樹状突起)の損傷程度が定量的に評価可能になると考えられる。この事は特に脳白質をおかす疾患(加齢性の変化を含めた)の進行度の評価、予後の推測(脱髄巣に於て軸索が残存しているかどうか)をする上で良い指標になると考えられ、また、拡散テンソルMR画像法を用いて国内外で行われている臨床研究の結果を組織学的微細構造の変化という観点から解釈する上でも有用な情報を提供すると考えられる。

References

1 Basser PJ, Mattiello J, LeBihan D. MR diffusion tensor spectroscopy and imaging. Biophys J. 1994; 66: 259-67.

2 Basser PJ, Mattiello J, LeBihan D. Estimation of the effective self-diffusion tensor from the NMR spin echo. J Magn Reson B. 1994; 103: 247-54.

3 Bammer R, Augustin M, Strasser-Fuchs S,et al. Magnetic resonance diffusion tensor imaging for characterizing diffuse and focal white matter abnormalities in multiple sclerosis. Magn Reson Med. 2000 ; 44: 583-91.

4 Filippi M, Cercignani M, Inglese M,et al. Diffusion tensor magnetic

resonance imaging in multiple sclerosis. Neurology. 2001; 56: 304-11.

5 Huppi PS, Maier SE, Peled S,et al. Microstructural development of human newborn cerebral white matter assessed in vivo by diffusion tensor magnetic resonance imaging. Pediatr Res. 1998; 44: 584-90.

6 Nusbaum AO, Tang CY, Buchsbaum MS,et al. Regional and global changes in cerebral diffusion with normal aging. AJNR Am J Neuroradiol. 2001; 22: 136-42.

7 Sakuma H, Nomura Y, Takeda K, et al. Adult and neonatal human brain: diffusional anisotropy and myelination with diffusion-weighted MR imaging. Radiology. 1991; 180: 229-33.

8 Takahashi M, Ono J, Harada K, et al. Diffusional Anisotropy in Cranial Nerves with Maturation: Quantitative Evaluation with Diffusion MR Imaging in Rats. Radiology 2000; 216: 881-5.

9 Mori S, Itoh R, Zhang J, et al. Diffusion tensor imaging of the developing mouse brain. Magn Reson Med. 2001 Jul;46(1):18-23.

10 Ito R, Mori S, Melhem ER. Diffusion tensor brain imaging and tractography. Neuroimaging Clin N Am. 2002 Feb; 12(1): 1-20.

11 Carr HY, Purcell EM. Effects of diffusion on free precession in nuclear magnetic resonance experiments. Phys Rev 1954; 94: 630-5.

12 Le Bihan D, Turner R. Intravoxel incoherent motion imaging using spin echoes. Magn Reson Med. 1991; 19: 221-27.

13 Stejskal EO, Tanner JE. Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient. J Chem Phys 1965; 42: 288-92.

14 Taylor DG, Bushell MC. The spatial mapping of translational diffusion coefficients by the NMR imaging technique. Phys Med Biol. 1985; 30: 345-9.

15 Le Bihan D, Breton E, Lallemand D, et al. MR imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders. Radiology. 1986; 161: 401-7.

16 Le Bihan D, Breton E, Lallemand D, et al. Separation of diffusion and perfusion in intravoxel incoherent motion MR imaging. Radiology. 1988; 168: 497-505.

17 Pierpaoli C, Jezzard P, Basser PJ, et al. Diffusion tensor MR imaging of the human brain. Radiology. 1996; 201: 637-48.

18 Basser PJ, Pierpaoli C. Microstructural and physiological features of tissues

elucidated by quantitative-diffusion-tensor MRI. J Magn Reson B. 1996; 111: 209-19.

19 Mukherjee P, Bahn MM, McKinstry RC, et al. Differences between gray matter and white matter water diffusion in stroke: diffusion-tensor MR imaging in 12 patients. Radiology. 2000; 215: 211-20.

20 Pierpaoli C, Basser PJ. Toward a quantitative assessment of diffusion anisotropy. Magn Reson Med. 1996; 36: 893-906.

21 Haselgrovr JC, Moore JR. Correction for distortion of Echo-Planar images used to calculate the apparent diffusion coefficient. Magn Reson Med 1996; 36: 960-4.

22 Jezzard P, Barnett AS, Pierpaoli C. Characterization of and correction for eddy current artifacts in echo planar diffusion imaging. Magn Reson Med. 1998; 39: 801-12.

23 Martin KM, Papadakis NG, Huang CL, et al. The reduction of the sorting bias in the eigenvalues of the diffusion tensor. Magn Reson Imaging. 1999; 17: 893-901.

24 Basser PJ, Pajevic S. Statistical artifacts in diffusion tensor MRI (DT-MRI) caused by background noise. Magn Reson Med. 2000; 44: 41-50.

25 Pajevic S, Pierpaoli C. Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: application to white matter fiber tract mapping in the human brain. Magn Reson Med. 1999; 42: 526-40.

fig. 1a Anisotropic diffusion model.
Left, motion (tortuous arrow) of water molecule (black dots) under ordered structures (rods) (e.g. myelin sheath, axon in white matter). Right, the probable location of the molecule after unit time will be within an ellipsoid. This condition is termed anisotropic.

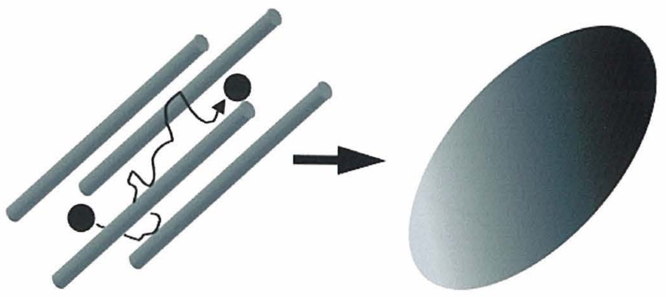


fig. 1b Isotropic diffusion model.
Left, path (tortuous line) of water molecule (black dots) under no spatial constrain. Right, The molecule moves randomly by Brownian motion, resulting in a sphere displacement profile. This condition is termed isotropic.

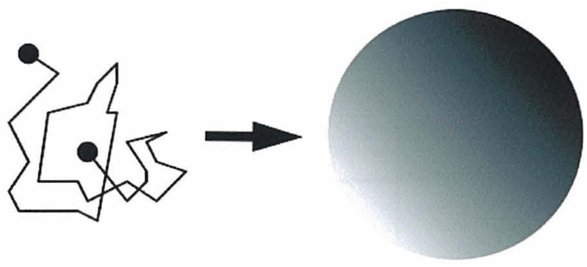


fig. 2 Ellipsoid representing diffusion tensor.
Diffusion properties can be described by an ellipsoid. The ellipsoid can be characterized by three variables (eigenvalues, λ_1, λ_2, λ_3) to describe the shape and relationship between the principal coordinate axes (x', y', and z') and the laboratory coordinate axes (x, y, and z). Each principle axes (x' , y' , z') are represented by three eigenvectors of the diffusion tensor. The radii (white arrows) of the diffusion ellipsoid along these principal axes are $(2\lambda_1)^{1/2}$, $(2\lambda_2)^{1/2}$,

and $(2\lambda_2)^{1/2}$ after unit time. Diagonalization means rotating the laboratory coordinate system and adjusting it to the principal coordinate system of the ellipsoid (curved arrows).

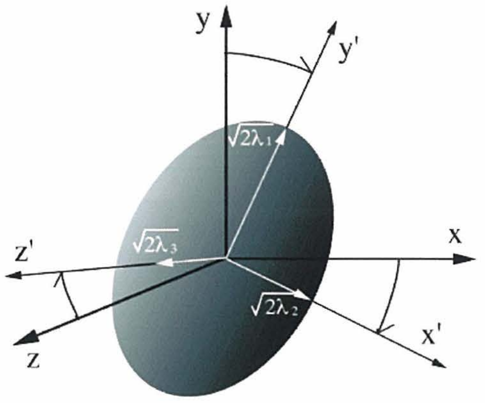


fig. 3 Stejskal-Tanner sequence.
In order to measure signal loss by diffusion, a pair of pulsed diffusion sensitizing gradients (grayish rectangles) is arranged on each side of the 180 degree radio frequency (RF) pulse in spin-echo sequence. G is the strength of the gradients that can be applied to any direction. δ is the duration of each gradient pulse and Δ refers to the time interval between their onsets.

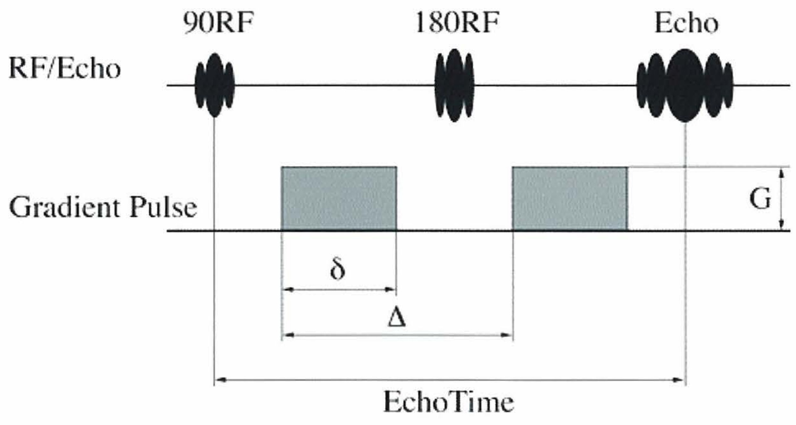


fig. 4 Steps of spatial mapping of diffusion process.
First, diffusion-weighted images with various axes of diffusion sensitizing gradient (upper right row) and an image without diffusion sensitization (upper left) are obtained. (Note: the diffusion-weighted images at the same location have different contrast that depends on the direction of applied diffusion gradient.) From decreases in intensities between "NO-weighted" and "diffusion-weighted" images, apparent diffusion constants at each pixel are calculated (middle). By acquiring at least 7 diffusion weighted images or six ADC maps, the diffusion ellipsoid at each pixel can be fully characterized, from which various maps that indicate water diffusion process can be calculated (bottom row).

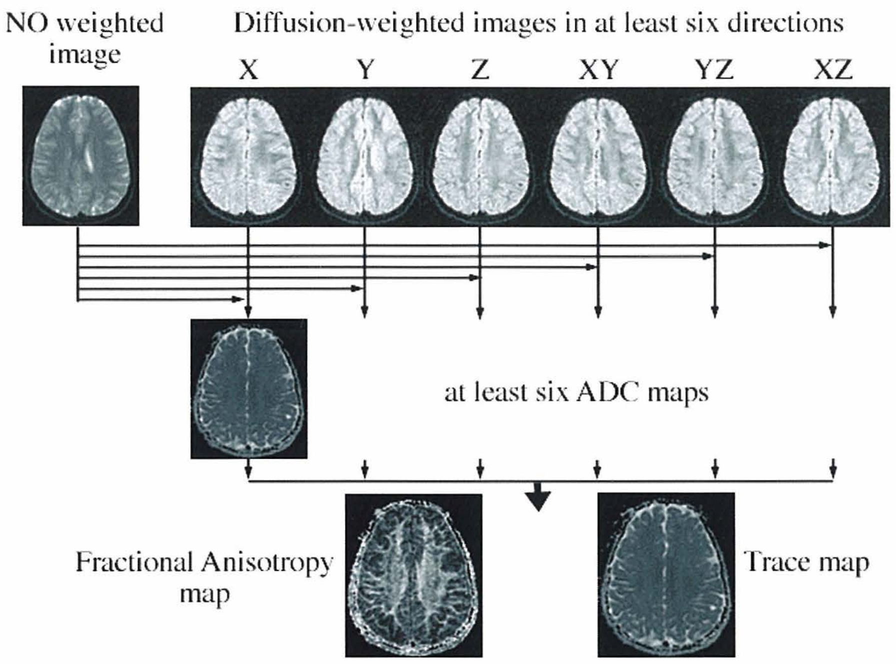

NO weighted image
Diffusion-weighted images in at least six directions
X
Y
Z
XY
YZ
XZ
at least six ADC maps
Fractional Anisotropy map
Trace map

fig. 5 Color map derived from DT MR imaging.
In this color map, brightness represents the degree of anisotropy and color represents fiber orientation (red: horizontal, green: vertical, blue: perpendicular to the plane). Fiber tracts such as the corpus callosum (cc), superior longitudinal fasciculus(slf), corticospinal tract (cst), and inferior longitudinal fasciculus (ilf) can be identified by their color representing their orientation.

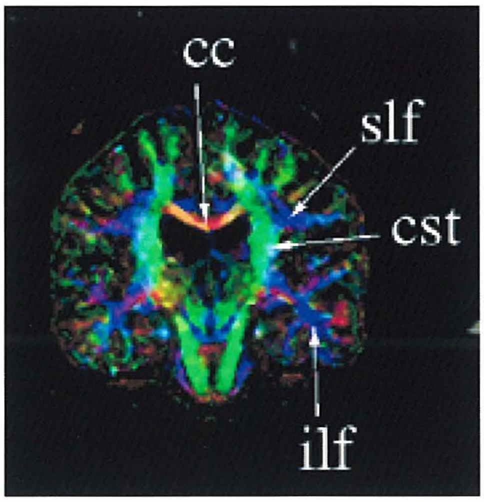


fig. 6 a view of a homemade coil
A coil is a cylinder shape with 12mm in diameter and 18mm in length (upper left). The coil position can be adjusted to the magnet center by moving the table mounting the coil in three directions orthogonally (upper right). The coil and table are fixed to the inside of the magnet by the cylinder shape attachment (bottom).

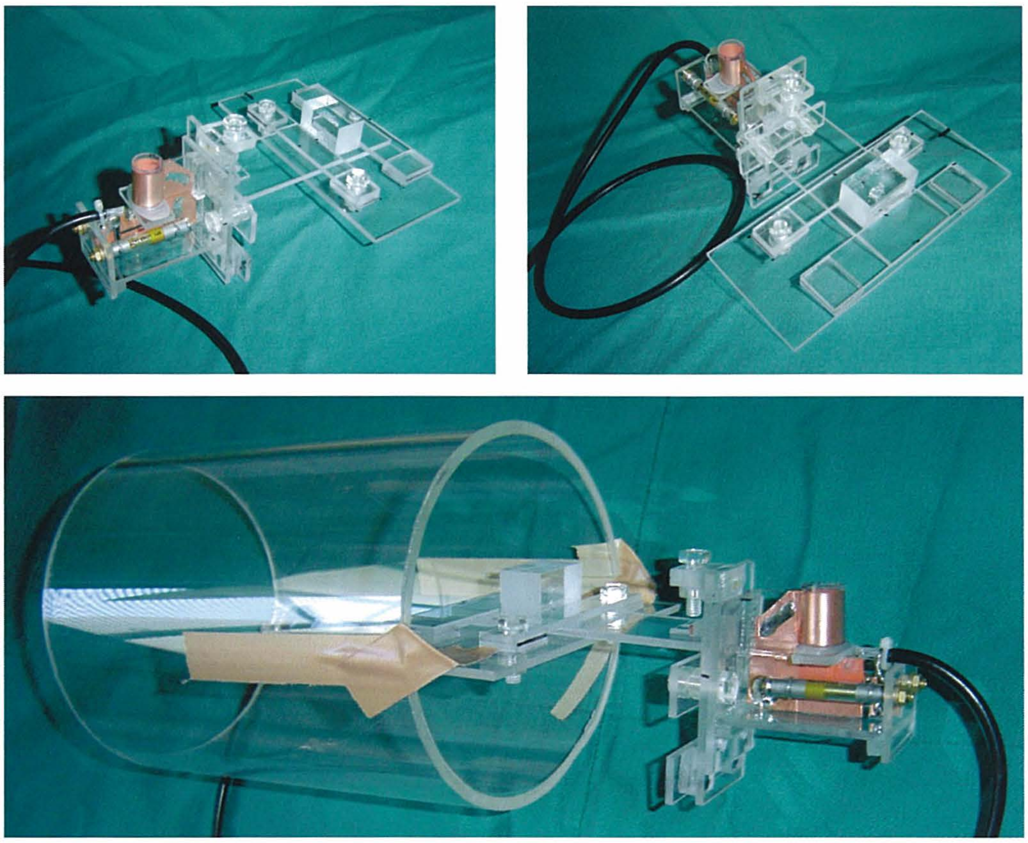


fig. 7 A window view of a distortion correction tool.
A difference image (bottom right) between a diffusion-weighted image with diffusion sensitizing gradient (bottom left) and an image without diffusion sensitization (upper) represents distortion by diffusion sensitization gradient induced eddy current.

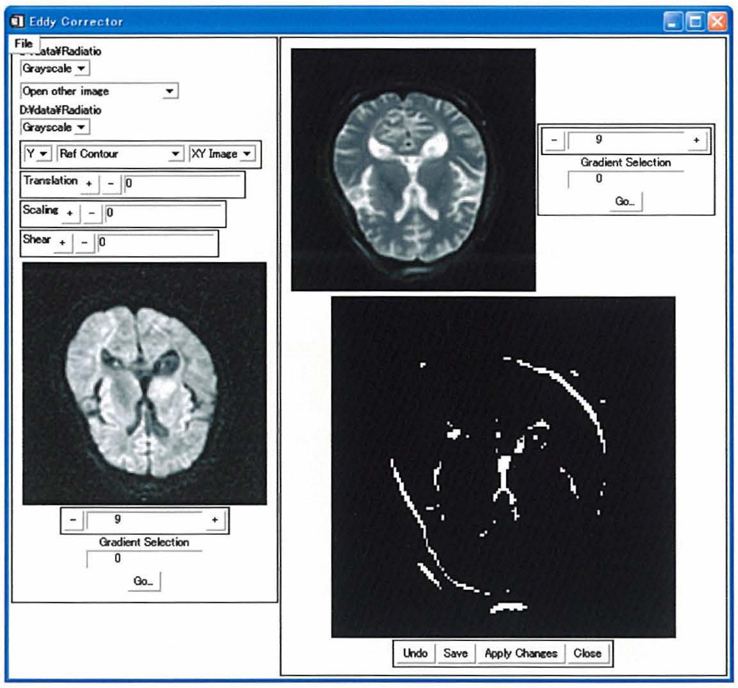


fig. 8 Obtained images of Sprague-Dawley rat brain.
Diffusion-weighted images with various axes ((x,y,z)=(1,0,1), (-1,0,1), (0,1,1), from upper left to right, (0,1,-1), (1,1,0), (-1,1,0), from middle left to right) of diffusion sensitizing gradient and an image without diffusion sensitization (bottom) are obtained.

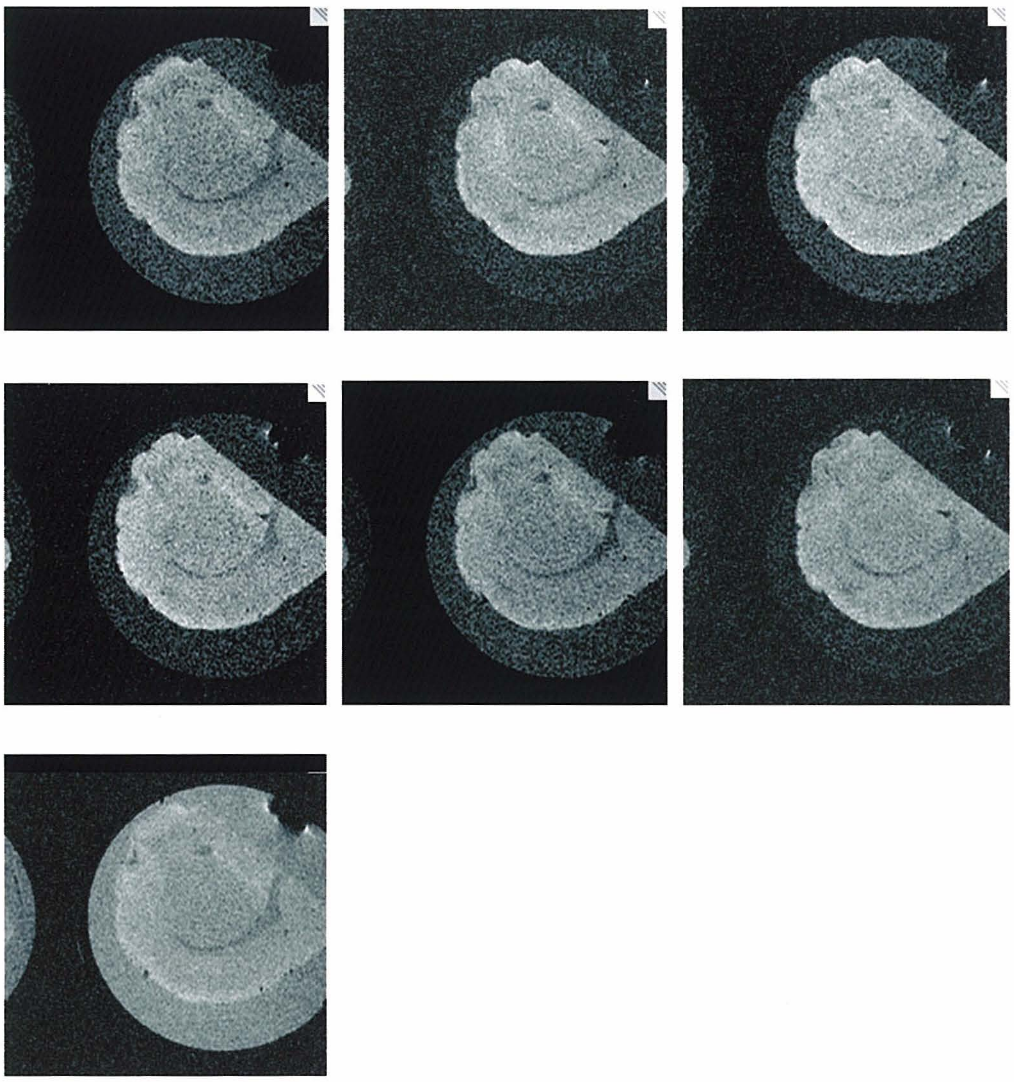
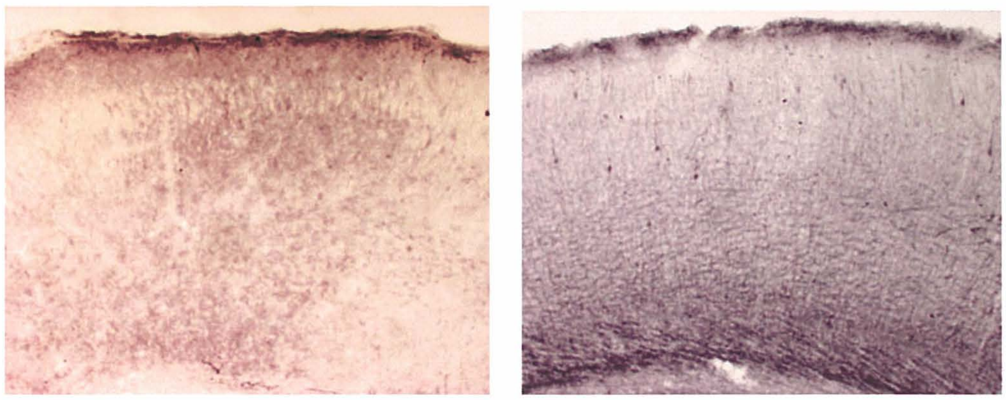


fig. 9 Histological images, immunohistochemical evaluation of day 0 Sprague-Dawley rat brain cortex, GFAP (glial fibrillary acidic protein) (left) and neurofilment (right). A reticulated structure is shown.



APPENDIX

APPENDIX 1a

```
/*Pulse Sequence File
# @(#) usa.s      Version 1.15 Created on 7/9/91 at 16:01:54
# @(#) Path = /home/sw0/omega-scm/omega/pplib/psg/SCCS/s.usa.s
# Copyright (c) 1991 by General Electric Company.   All rights reserved.
#
 ****************************************************************
 *                                                              *
 *      usa.s   - Universal Spin-warp Acquisition Sequence      *
 *                                                              *
 ****************************************************************/

/*************** z-variable definitions ***************/

calx "x-gradient calibration strength" = 3000.0;   /* Hz/mm/32K DAC units */
caly "y-gradient calibration strength" = 3000.0;
calz "z-gradient calibration strength" = 3000.0;
crlist[32];
difd "diffusion delay" = 5ms;
(int) difp "diffusion pulse flag" = 0;
dift "diffusion gradient time" = 5ms;
difx "x-diffusion gradient (G/cm)" = 0;
dify "y-diffusion gradient (G/cm)" = 0;
difz "z-diffusion gradient (G/cm)" = 0;

(int) grfl "read gradient flag" = 1;
(int) echoes "number of echoes" = 1;
f1cal "F1 Amplitude Calibration File" = " ";
(int) flip "RF phase flip" = 0;
fov "field of view" = 100.0;
fovp "phase encode, field of view" = 0.0;
fovx "X-field of view" = 100.0;
fovy "Y-field of view" = 100.0;
fovz "Z-field of view" = 100.0;
grdpa "read dephase adjuster" = 0;
gscr "crusher gradient value" = 0.0;
gscrt "crusher gradient time" = 2ms;
(int) gflag "phase encode gradient on/off" = 0;
(int) grass "gradient echo flag" = 0;

(int) hard180 "hard 180 pulse flag" = 0;
(int) opfl "one pulse flag" = 0;
p1cal "F1 Phase Calibration File" = " ";
pet "phase encode time" = 2ms;
phlimit[16] "phase ramp limit";
(int) plane "image plane tran/sag/cor" = 1;
ppl "hard pulse power level" = 100.0;
pw "90 deg pulse width" = 10us;
(int) res "phase encode resolution" = 128;
(int) resx "X-phase encode resolution" = 1;
(int) resy "Y-phase encode resolution" = 1;
(int) resz "Z-phase encode resolution" = 1;

rt "gradient ramp time" = 500us;

sincc "sinc pulse cycles" = 2.0;
sinct "sinc pulse time" = 4ms;
(int) slices "number of slices" = 1;
slo "slice offset" = 0.0;
slsep "slice separation" = 3.0;
spl "soft pulse power level" = 100.0;
st "slice thickness" = 2.0;
```

```
te "echo time" = 28ms;
(int)threed "3D imaging flag" = 0;
ti "inversion time" = 0.0;
tr "recycle time" = 500ms;
(int) tslice "3D thick-slice flag" = 0;
(int) npea = 0;
$

/*************** pulse sequence expressions ***************/

/* Error code
0 No Errors
1       illegal plane; must be 1,2,3,4"
2       illegal echoes; must be 1,2,4,8,16,32,64"
3       illegal slices; must be 1,2,4,8"
4       te1 < 0; (check te,sw,cb)"
5       te2 < 0; (check te,sw,cb)"
6       gpeb > 100%; (check fov,fovp)"
7       grdp > 100%; (check fov,sw,cb,pet)"
8       gro > 100%; (check fov,sw,cb)"
*/
(int) Error = 0;

acq_mode = 2; /* must be set to 2 for multiple Q (3D) acquisitions */

/* crusher list for multi-echo sequence; maximum 32 echoes */
int i = 0;
int j = 1;
while (i < 32)
            crlist[i] = gscr*j;
            i = i+1;
            crlist[i] = -gscr*j;
            i = i+1;
            j = j+1;
endw

grA = 100;
tf = 0.000001;
gamma = sf1/cf;         /* magnetogyric ratio */
calx1 = gamma * calx;
caly1 = gamma * caly;
calz1 = gamma * calz;
if (plane == 1)
            calsl = calz1;
            calpe = caly1;
            calro = calx1;
    gpesl = grA*1.57/(fovz*pet*tf*calsl);
    gpebsl = ( gpesl * (resz - 1)/2.0 );
            (int) slres = resz;
            g_plane[0] = {1,2,3}; /* default, ga=gx=read,gb=gy=encode,gc=gz=slice */
            difa = 42500*difx/calx1;
            difb = 42500*dify/caly1;
            difc = 42500*difz/calz1;
else if (plane == 2)
            calsl = calx1;
            calpe = calz1;
            calro = caly1;
    gpesl = grA*1.57/(fovx*pet*tf*calsl);
    gpebsl = ( gpesl * (resx - 1)/2.0 );
            (int) slres = resx;
            g_plane[0] = {2,3,1}; /* ga=gy=read,gb=gz=encode,gc=gx=slice */
            difa = 42500*dify/caly1;
            difb = 42500*difz/calz1;
            difc = 42500*difx/calx1;
```

```
else if (plane == 3)
                calsl = caly1;
                calpe = calx1;
                calro = calz1;
    gpesl = grA*1.57/(fovy*pet*tf*calsl);
    gpebsl = ( gpesl * (resy - 1)/2.0 );
                (int) slres = resy;
                g_plane[0] = {3,1,2}; /* ga=gz=read,gb=gx=encode,gc=gy=slice */
                difa = 42500*difz/calz1;
                difb = 42500*difx/calx1;
                difc = 42500*dify/caly1;
else if (plane == 4)
                calsl = calz1;
                calpe = caly1;
                calro = calz1;
                g_plane[0] = {3,2,3}; /* slice profile, ga=gz=read,gb=g.=..,gc=gz=slice */
else if (plane == 10)
                calsl = calz1;
                calpe = calx1;
                calro = -caly1;
    gpesl = grA*1.57/(fovz*pet*tf*calsl);
    gpebsl = ( gpesl * (resz - 1)/2.0 );
                (int) slres = resz;
                g_plane[0] = {2,1,3}; /* ga=gy=read,gb=gx=encode,gc=gz=slice */
else if (plane == 20)
                calsl = calx1;
                calpe = caly1;
                calro = -calz1;
    gpesl = grA*1.57/(fovx*pet*tf*calsl);
    gpebsl = ( gpesl * (resx - 1)/2.0 );
                (int) slres = resx;
                g_plane[0] = {3,2,1}; /* ga=gz=read,gb=gy=encode,gc=gx=slice */
else if (plane == 30)
                calsl = caly1;
                calpe = calz1;
                calro = -calx1;
    gpesl = grA*1.57/(fovy*pet*tf*calsl);
    gpebsl = ( gpesl * (resy - 1)/2.0 );
                (int) slres = resy;
                g_plane[0] = {1,3,2}; /* ga=gx=read,gb=gz=encode,gc=gy=slice */
else
                Error = 1;    /* illegal plane */
endif
endif
endif
endif
endif
endif
endif
if !((threed)||(tslice))    /* check if 2D or 3DFT data */
    (int) slres = 1;
endif

/** pet = (grA * 1.57)/(calpe * gpes * fove); **/
if (fovp == 0)    /* phase encode, FOV can be different to fov */
                fovpe = fov;
else
                fovpe = fovp;
endif

gsl = ((grA * sincc * 2)/(calsl * st * sinct * tf));

if (grass)
                gsrf = (gsl/2.0) * (sinct + rt) * (1.57/pet);
else
```

```
                gsrf = (gsl/2.0) * (sinct + rt) * (1.57/gscrt);
endif

pho = 360 * slo * calsl * gsl * sinct * tf/grA;
phoff = 360 * slsep * calsl * gsl * sinct * tf/grA;

if (slices == 1)
          phlimit[0] = pho;
                sincr = 400;
else if (slices == 2)
          phlimit[0] = 0.5 * phoff + pho;
                phlimit[1] = -0.5 * phoff + pho;
                sincr = 400;
else if (slices == 4)
          phlimit[0] =1.5 * phoff + pho;
          phlimit[1] = -0.5 * phoff + pho;
          phlimit[2] =0.5 * phoff + pho;
          phlimit[3] = -1.5 * phoff + pho;
                sincr = 320;
else if (slices == 8)
          phlimit[0] = 3.5 * phoff + pho;
          phlimit[1] = 1.5 * phoff + pho;
          phlimit[2] = -0.5 * phoff + pho;
          phlimit[3] = -2.5 * phoff + pho;
          phlimit[4] = 2.5 * phoff + pho;
          phlimit[5] = 0.5 * phoff + pho;
          phlimit[6] = -1.5 * phoff + pho;
          phlimit[7] = -3.5 * phoff + pho;
                sincr = 160;
else
                Error = Error+300;
endif
endif
endif
endif

bw = 1/(dw*tf);   /* spectral bandwidth calculated from dw */
gread = bw/fov;   /* read gradient Hz/mm */
gro = -(grA*gread/calro) * grfl;   /* read gradient, note grfl flag */
grdp = gro * ((de + at + rt)/2) * 1.57/pet + grdpa; /* read dephase, note grfl */
if (grdp < -100)       /* check gradient limits */
                Error = Error+7000000;
endif
if (gro < -100)
                Error = Error+80000000;
endif
if (grass && threed)
                grdp = -grdp;
                te1 = te - (((pw + de + at)/2) + rt + pet);
                te2 = 0.0;
else if (grass)
                grdp = -grdp;
                te1 = te - (((sinct + de + at)/2) + (rt*2) + pet);
                te2 = 0.0;
else if (threed)
          te1 = (te/2) - ((1.5*pw) + pet + gscrt + ((dift+difd)*difp));
          te2 = (te/2) - ((at/2) + pw + de + rt + gscrt + ((dift+difd)*difp));
else if (hard180)
          te1 = (te/2) - ((sinct/2) + rt + pet + gscrt + pw + ((dift+difd)*difp));
          te2 = (te/2) - ((at/2) + pw + de + rt + gscrt + ((dift+difd)*difp));
else
          te1 = (te/2) - (sinct + (rt*2) + pet + gscrt + ((dift+difd)*difp) );
          te2 = (te/2) - (((sinct + de + at)/2) + (rt*2) + gscrt + ((dift+difd)*difp) );
endif
endif
```

```
endif
endif

if (difp)
                if (threed || hard180)
                                delta = (dift+difd+gscrt+pw+pw+difd+gscrt)*tf;
                else
                                delta = (dift+difd+gscrt+rt+sinct+rt+gscrt+difd)*tf;
                endif
                /* b values */
b1 = gamma*gamma*0.026754*0.026754*0.4057*dift*dift;
                bx = (delta-((dift*tf)/3.0))*b1*difx*difx/100.0;
                by = (delta-((dift*tf)/3.0))*b1*dify*dify/100.0;
                bz = (delta-((dift*tf)/3.0))*b1*difz*difz/100.0;
endif

if (threed)
                pd = tr - ( (echoes*te) + (pw/2) + de + (at/2) + rt + (pet*grass));
else
                pd = tr - ( (sinct/2) + (de + at/2) + (echoes*te) + rt + rt + (pet*grass));
endif
if (ti > 0)
           pd = pd - (ti + sinct + rt + rt + (pet*grass));
endif
pd = pd/slices;

/* set error flags */
/* maximum echoes allowed, 32 */
if !((echoes==1)||(echoes==2)||(echoes==4)||(echoes==8)||(echoes==16)||(echoes==32))
                Error = Error+20;
endif
if (te1 < 0)
                Error = Error+4000;
endif
if (te2 < 0)
                Error = Error+50000;
endif
gpes = ((grA*1.57)/(fovpe*pet*tf*calpe));
gpeb = ( gpes * (res - 1)/2.0 );
if (gpeb > 100)
                Error = Error+600000;
endif
gpe3 = (gpebsl - (gpesl * (npea)) ) * gflag;


/* run time calculations are written after the @ sign */
@

/* alternate phase on odd encode steps, to avoid zero frequency artifact */
qph = (nbc%2)*flip;

(int) slc = 0;    /* slice counter */
gpe = (gpeb - (gpes * (nbc)) ) * gflag;
if (grass)
                gpe = -gpe;
                gpe3 = -gpe3;
endif

if (opfl == 1)
                T(pd);
                f1 {D pw, M pp1, PH ((nac%2)*180)};
                Q((nac%2)*180);
else
while (slc < slices)
                T(pd);
```

```
    if (ti > 0)
                if (threed || hard180)
                            fl{D pw*2, M ppl, PH 0};
                else
                            SOFT_180A:
                            gc{D rt, M gsl/2, F ramp(P1 0, P2 100), R 50};
                            gc{D sinct, M gsl/2},
                            fl{D sinct, M 100, N spl, F sinc(C sincc), R sincr},
                            ph{D sinct, M 0, F ramp(P1 0, P2 phlimit[slc]/2), R sincr};
                            SOFT_180B:
                            gc{D rt, M gsl/2, F ramp(P1 100, P2 0), R 50};
                endif

                T(ti);
endif

    if (threed)
                fl{D pw, M ppl, PH ((nac%2)*180)};
    else
                gc{D rt, M gsl, F ramp(P1 0, P2 100), R 50};
                gc{D sinct, M gsl},
                fl{D sinct, M 100, N spl/2.0, F sinc(C sincc), R sincr},
                ph{D sinct, M (nac%2)*180, F ramp(P1 0, P2 phlimit[slc]), R sincr};
                gc{D rt, M gsl, F ramp(P1 100, P2 0), R 50};
    endif


    if (grass && tslice)
                gb{D pet, M gpe, F sine(P 0, C 0.5), R 100},
                ga{D pet, M grdp, F sine(P 0, C 0.5), R 100},
                gc{D pet, M (gpe3-gsrf), F sine(P 0, C 0.5), R 100};
    else if (grass && threed)
                gb{D pet, M gpe, F sine(P 0, C 0.5), R 100},
                ga{D pet, M grdp, F sine(P 0, C 0.5), R 100},
                gc{D pet, M gpe3, F sine(P 0, C 0.5), R 100};
    else if (grass)
                gb{D pet, M gpe, F sine(P 0, C 0.5), R 100},
                ga{D pet, M grdp, F sine(P 0, C 0.5), R 100},
                gc{D pet, M -gsrf, F sine(P 0, C 0.5), R 100};
    else if (tslice || threed)
                gb{D pet, M gpe, F sine(P 0, C 0.5), R 100},
                ga{D pet, M grdp, F sine(P 0, C 0.5), R 100},
                gc{D pet, M gpe3, F sine(P 0, C 0.5), R 100};
    else
                gb{D pet, M gpe, F sine(P 0, C 0.5), R 100},
                ga{D pet, M grdp, F sine(P 0, C 0.5), R 100};
    endif
    endif
    endif
    endif

    T(te1);
    if (difp)
                DIFFUSION_PULSE:
                ga{D dift, M difa, F sine(P 0, C 0.5), R 200},
                gb{D dift, M difb, F sine(P 0, C 0.5), R 200},
                gc{D dift, M difc, F sine(P 0, C 0.5), R 200};
                T(difd);
    endif

    if (!grass)
                if (threed || hard180)
                            gscrm = gscr;
                            CRUSHER:
                            gc{D gscrt, M gscrm, F sine(P 0, C 0.5), R 100};
```

```
                                        f1{D pw*2, M ppl, PH 0};
                                        /*
                                        gscrm = gscr+(gsrf*hard180);
                                        repeat(CRUSHER);
                                        */
                                        gc{D gscrt, M gscr+(gsrf*hard180), F sine(P 0, C 0.5), R 100};
                        else
                                        gscrm = gscr;
                                        repeat(CRUSHER);
                                        repeat(SOFT_180A,SOFT_180B);
                                        /*
                                        gscrm = gscr+gsrf;
                                        repeat(CRUSHER);
                                        */
                                        gc{D gscrt, M gscr+gsrf, F sine(P 0, C 0.5), R 100};
                        endif
                        if (difp)
                        T(difd);
                                        repeat(DIFFUSION_PULSE);
                        endif
                        T(te2);
            endif
            GARAMPUP:
            ga{D rt, M gro, F ramp(P1 0, P2 100), R 50};
            ga{D at+de, M gro},
            /* flip alternates phase on every odd phase encode step */
            Q(((nac%2)+qph)*180);
            GARAMPDN:
            ga{D rt, M gro, F ramp(P1 100, P2 0), R 50};
            (int) ec = 1;
            if (echoes > 1)
                        while (ec < echoes)
                                        T(te2);
                                        gscra = crlist[ec];
                                        CRUSHER2:
                                        gc{D gscrt, M gscra, F sine(P 0, C 0.5), R 100};
                                        if (threed || hard180)
                                                    f1{D pw*2, M ppl, PH 0};
                                        else
                                                    repeat(SOFT_180A,SOFT_180B);
                                                    repeat(CRUSHER2);
                                        endif

                                                    T(te2);
                                                    repeat(GARAMPUP);
                                                    ga{D at+de, M gro},
                                                    Q(((nac%2)+qph)*180);
                                                    repeat(GARAMPDN);
                                        ec = ec+1;
                        endw
            endif
            slc = slc+1;
            /* rewinder pulse */
            if (grass)
                        gb{D pet, M -gpe, F sine(P 0, C 0.5), R 100},
                        gc{D pet, M -gpe3, F sine(P 0, C 0.5), R 100};
            endif
endw

endif

/********************** modification history *******************************

02Feb91: Diffusion gradients added.
            Bugs fixed - planes 20,30. (ss)
```

```
25Mar91: Error flags set (ss)
        Bug fix - slres,pd
28Mar91: Phase-flip flag to flip phase on odd encode steps (ss)
15Apr91: Bug - pd for 3D (ss)
                                        Multi-echo with hard180 pulse
02May91: header installed;sincr for multi-slice changed (ss)
06May91: planes 10,20,30 fixed for clockwise rotation by 90deg. (ss)
27Jun91: repeat(labels), statements added (ss)
03Jul91: grfl flag to turn of read gradient
                                        bug in b-value calculation fixed (ss)
09Jul91: bug in soft 90 pulse, gsl/2 => gsl (ss)
              rewinder (gb) pulse added at end of sequence
26Aug91: Upto 32 echoes allowed. Crusher list changed. (ss)
27Aug91: Bug, te2 calculation for 3D fixed. (ss)
              Work-around for bug in repeat statements.
29Aug91: Rewinder activated for grass; gpe3 added. (ss)

*****************************************************************************/
```

APPENDIX 1b

```
#Omega Sourceable Script

onintr quit
echo -n "Enter 2D raw data name (ex. /mnt/030604.1)..."
set SOF = $<
echo -n "Enter 1st extension (ex. 0)..."
set FIR = $<
echo -n "Enter last extension (ex. 63)..."
set LAST = $<
echo -n "Enter file name of processed 3D data (ex. 3D030604.1)..."
set PROC = $<

memory 0x800000

cp $SOF.$FIR $PROC
srm -w $PROC
header -i par1 dim0
header -i par2 dim1
@ par3 = 1 + $LAST - $FIR
resize $par1 $par2 $par3
@ EXT = $FIR
@ EXT2 = 1
while ($EXT <= $LAST)
view 2 1 2 $EXT2
get $SOF.$EXT
math 1 = 1 + 0
echo "$SOF.$EXT was added..."
stackrm
@ EXT += 1
@ EXT2 += 1
end
# save test3D

header -i par3 dim2
header -i qs nq
# header -i ncs nc
header -i nex na
header -i echoes echoes
set ncs = 2
set zf = 0
```

```
            put -f *
            echo " Processing $par1*$par2*$par3 image data"
            set yesno = n
            echo -n "Do you want zerofilling in dimension 1 (y/n)?    "
            set yesno = $<
            if ( $yesno == y ) then
            zerofill 1 1
            endif
            view 3 1 2 3
            next -n 0
            header -i numb groupnum
            @ cntr = 0
            echo -n " Processing dimension 1 ($numb groups)"
            while ( $cntr < $numb )
                        @ cntr++
                        if ($nex == 1) bc
                        if ( $yesno == y ) then
                        trapmult -p 0 15 35 50
                        else
                        trapmult -p 0 30 70 100
                        endif
                        ft
                        next
                        echo -n " ."
            end
            echo ""
set yesno = n
echo -n "Do you want zerofilling in dimension 2 (y/n)?    "
            set yesno = $<
            if ( $yesno == y ) then
            echo " zerofilling data"
            zerofill 2 1
            endif
            view 3 2 1 3
            next -n 0
            header -i numb groupnum
            @ cntr = 0
            echo ""
            echo -n " Processing dimension 2 ($numb groups)"
            while ( $cntr < $numb )
                        @ cntr++
                        ft
                        next
                        echo -n " ."
            end
            echo ""
            set yesno = n
            echo -n "Do you want zerofilling in dimension 3 (y/n)?    "
                        set yesno = $<
                        if ( $yesno == y ) then
                        echo " zerofilling data"
                        zerofill 3 1
                        endif
            view 3 3 1 2
            next -n 0
            header -i numb groupnum
            @ cntr = 0
            echo ""
            echo -n " Processing dimension 3 ($numb groups)"
            while ( $cntr < $numb )
                        @ cntr++
                        ft
                        magcalc
                        next
```

```
                                echo -n " ."
                end
                echo ""
                echo " Done"

quit:
memory 0x100000
```

APPENDIX 2

```
; corr_epi program.

; J.Y.Zhang 2000
; Last change. 5,3,2000
; Purpose:
; a GUI tool to correct the artifacts caused by eddy current in
; the phase encoding direction of MR EPI images.


; This two functions below are   originally written by Susumo Mori,
; I changed it
; to use it write and read data
; Jiangyang Zhang, Last Updated. 4,19,2000
; To use these functions, you must be able to access Susumu's
; idlprogram directory, because they need some subroutines there.

; R.Ito
; modified in the parts for I/O image data.
; Last updated. July 21/2002

; bugs, in part of "APPLY", fixed. July 27/2002.
;                corrected the last arg. of 'CORRECT' function
;                Because of this bug, other previous versions could not work correctly!!
; modified for operating on WinOS, Sep 13 /2002

FUNCTION FILE_READY, datafile, tdim

; datafile is the name of file that contains all the information
; about data.


;pro tensor_im4
;This program is a main driver to process the diffusion images
;tensor_im1 + bunch of anisotropy indecis

;************************  input data files  ************************
get_filenames:

get_data:

num_files=1
cb=128
nb=128
nc=20

;------------------------------------
; make the array of filenames.

filename=datafile
;------------------------------------

  num_imagefile=num_files

;********************** echo data file read **************
```

```
; echo filenames
echo:
tdim=uintarr(cb,nb,nc,num_imagefile)

print, 'x_size,y_size,slice_num', cb, nb, nc
print, 'num_files', num_files

for i=0, num_imagefile-1 do begin

rawdata=uintarr(cb,nb,nc)
get_lun, unit
openr, unit, filename
readu, unit, rawdata
close, unit
free_lun, unit
tdim(*,*,*,i)=rawdata
print, filename, 'read OK'
rawdata=0

endfor

;tdim=swap_endian(tdim)

return, 1
goto, endofproc

error:

return, 0

endofproc:
end

FUNCTION FILE_READ_DATA, filename, tdim
tdim=0
datafile_name=''
damy=''
ss=intarr(10)

get_lun, unit
openr, unit, filename
readf, unit, damy
if (damy ne 'data file name') then goto, error_end
readf, unit, datafile_name
print, 'datafile name', datafile_name
readf, unit, damy
readf, unit, damy
if (damy ne 'dimensions') then goto, error_end
readf, unit, damy
ss[0]=UINT(damy)
for i=1, ss[0] do begin
readf, unit, damy
ss[i]=UINT(damy)
end
print, 'dimensions', ss
close, unit
free_lun, unit

if( ss[0] lt 2) then goto, error_end
if (ss[0] eq 2) then begin
tdim=flttarr(ss[1], ss[2])
endif
if (ss[0] eq 3 ) then begin
tdim=fltarr(ss[1], ss[2], ss[3])
```

```
endif
if (ss[0] eq 4 ) then begin
tdim=fltarr(ss[1], ss[2], ss[3], ss[4])
endif
if (ss[0] gt 4) then goto, error_end

get_lun, unit
openr, unit, datafile_name
readu, unit, tdim
close, unit
free_lun, unit

error_end:
return, 0
end;



FUNCTION FILE_SAVEY,    tdim, filename

pieces=str_sep(filename,'.')
header_file_name=strcompress(pieces(0)+'.hdr')
data_file_name=strcompress(pieces(0)+'.imgc')

ss=size(tdim)
; write the information about the data into header file.

get_lun, unit
openw, unit, header_file_name
printf, unit, 'data file name'
printf, unit, data_file_name
printf, unit, ' '

printf, unit, 'dimensions'
for i=0, ss[0] do begin
printf, unit, ss[i]
end

printf, unit, ' '
close, unit
free_lun, unit

get_lun, unit
openw, unit, data_file_name

;for index=0, nc*(num_files)-1 do begin
;d=assoc(unit, fltarr(a,b,/nozero))
;d(index)=tdim_temp(*,*,index)
;end
writeu, unit, tdim

close, unit
free_lun, unit

void=DIALOG_MESSAGE(['The FILEs: ', $
                                        +header_file_name, $
                                        +data_file_name, $
                                        +'were saved successfully.'],/INFORMATION)


tdim_temp=1
return, 0
end
```

```
FUNCTION CORRECTION, img, trans, scale, shear, dir
; This routine correct the translation, shearing and scaling caused by
; eddy current.

newone=img
s=size(img)
if (s[0] ne 2 )    then begin
                    return, newone
endif

xdim=s[1]
ydim=s[2]

; if the artifact is on x direction.
if (dir eq 0 ) then begin
; first deal with translation.
if ( trans NE 0 ) then begin
                    x=indgen(xdim)
                    x1=x+trans
                    for i=0, ydim-1 do newone[*,i]=interpolate( img[*,i],x1)
endif

; then the scaling.
if (scale NE 0 ) then begin
                    x=indgen(xdim)
                    x1=x-xdim/2
                    x1=x1*scale
                    x1=x1+xdim/2
                    for i=0, ydim-1 do newone[*,i]=interpolate( newone[*,i],x1)
endif

; the the shearing
if (shear NE 0 ) then begin
                    x=indgen(xdim)
                    for i=0, ydim-1 do begin
                                        rt=shear*0.1*(ydim/2-i)
                                        x1=x+rt
                                        newone[*,i]=interpolate( newone[*,i],x1)
                    endfor
endif
endif

; if the artifact is on y direction.
if (dir eq 1 ) then begin

newone=transpose(newone)
img=transpose(img)
; first deal with translation.
if ( trans NE 0 ) then begin
                    y=indgen(ydim)
                    y1=y+trans
                    for i=0, xdim-1 do newone[*,i]=interpolate( img[*,i],y1)
endif

; then the scaling.
if (scale NE 0 ) then begin
                    y=indgen(ydim)
                    y1=y-ydim/2
                    y1=y1*scale
                    y1=y1+ydim/2
                    for i=0, xdim-1 do newone[*,i]=interpolate( newone[*,i],y1)
endif

; the the shearing
if (shear NE 0 ) then begin
```

```
                y=indgen(ydim)
                for i=0, xdim-1 do begin
                                rt=shear*0.1*(xdim/2-i)
                                y1=y+rt
                                newone[*,i]=interpolate( newone[*,i],y1)
                endfor
endif
newone=transpose(newone)
img=transpose(img)
endif

return, newone


end

; image display function.

PRO SHOW_IMAGE, image, win_idx, xdim, ydim, options
DEVICE, DECOMPOSED=0
;if( image eq 0 ) then return

;print, 'size of image to be displayed',size(image)
temp=congrid(image, xdim, ydim,1) ; change the image size.
if (options eq 1) then begin ; 1-boundary, 0-intensity.
        temp=BYTSCL(sobel(temp))
        endif else begin
        temp=BYTSCL(temp)
        endelse

 wset, win_idx & tvscl, temp
 temp=0
 end

; image display function 2.

PRO SHOW_IMAGE2, image1, image2, win_idx, xdim, ydim, option1, option2
; option1 while it equals 1, it will display the image, 2 it will display
; the contour,
; option2 while it equals 1, it will display the contour overlay, else nothing.
ss1=size(image1)
ss2=size(image2)
DEVICE, DECOMPOSED=0
if ( ss1[0] eq 2 AND ss2[0] eq 2 ) then begin
t_image1=sobel(image1)
t_image2=sobel(image2)
t_image1=congrid(t_image1, xdim, ydim)
t_image2=congrid(t_image2, xdim, ydim)
max1=max(t_image1)
max2=max(t_image2)
min1=min(t_image1)
min2=min(t_image2)
avg1=(max1+min1)/3.0
avg2=(max2+min2)/3.0
res=where(t_image1 ge avg1)
t_image1=0
t_image1=bytarr(xdim, ydim)
t_image1(res)=1
res=0
res=where(t_image2 ge avg2)
t_image2=0
t_image2=bytarr(xdim, ydim)
t_image2(res)=1

 wset, win_idx
```

```
if (option1 eq 1 ) then begin
tvscl, congrid(image1, xdim, ydim)
endif
if (option1 eq 2 ) then begin
contour, t_image1, XSTYLE=12, YSTYLE=12
endif
if (option2 eq 1 ) then begin
contour, t_image2, C_COLORS=[300, 100], /OVERPLOT, XSTYLE=12, YSTYLE=12
endif

endif


end

FUNCTION PUT_IMAGE, dataset, options, idx, g, image

ss=size(dataset)
; if it is already 2D image, just return it.
if (ss[0] eq 2) then begin
            dataset=image
        return, 1
endif

; if the dimension is less than 2,
if (ss[0] lt 2) then return, 0

; if want xy image.
   if (options eq 0 ) then begin   ; in xy direction
        if (idx gt ss[3] ) then idx=ss[3]-1
        if (idx lt 0 ) then idx=0
            if( ss[0] eq 3) then begin
            dataset(*,*,idx)=congrid(image, ss[1], ss[2])
        return, 1
            endif else begin
            if (g gt ss[4]) then g=ss[4]-1
            if (g lt 0) then g=0
            dataset(*,*,idx,g)=congrid(image, ss[1], ss[2])
            ;print, 'size', size(image)
            return,1
            endelse
            endif


   if (options eq 1 ) then begin   ; in yz direction
        if (idx gt ss[1] ) then idx=ss[1]-1
        if (idx lt 0 ) then idx=0
            if( ss[0] eq 3) then begin
        dataset(idx,*,*)=congrid(image, ss[1], ss[2])
            return, 1
            endif else begin
            if (g gt ss[4]) then g=ss[4]-1
            if (g lt 0) then g=0
            dataset(idx,*,*,g)=congrid(image, ss[1], ss[2])
            return, 1
            endelse
            endif


   if (options eq 2 ) then begin   ; in xz direction
        if (idx gt ss[2] ) then idx=ss[2]-1
        if (idx lt 0 ) then idx=0
            if( ss[0] eq 3) then begin
            dataset(*,idx,*)=congrid(image, ss[1], ss[2])
             return, 1
```

```
                endif else begin
                if (g gt ss[4]) then g=ss[4]-1
                if (g lt 0) then g=0
                dataset(*,idx,*,g)=congrid(image, ss[1], ss[2])
                return, 1
                endelse
                endif


return, 0

end


FUNCTION GET_IMAGE, dataset, options, idx,g,image


ss=size(dataset)
; if it is already 2D image, just return it.
if (ss[0] eq 2) then begin
                image=dataset
            return, 1
endif

; if the dimension is less than 2,
if (ss[0] lt 2) then return, 0

; if want xy image.
    if (options eq 0 ) then begin    ; in xy direction
            if (idx gt ss[3]-1 ) then idx=ss[3]-1
            if (idx lt 0 ) then idx=0
                if( ss[0] eq 3) then begin
                image=dataset(*,*,idx)
            return, 1
                endif else begin
                if (g gt ss[4]-1) then g=ss[4]-1
                if (g lt 0) then g=0
                image= dataset(*,*,idx,g)
                ;print, 'size', size(image)
                return,1
                endelse
                endif


    if (options eq 1 ) then begin    ; in yz direction
            if (idx gt ss[1]-1 ) then idx=ss[1]-1
            if (idx lt 0 ) then idx=0
                if( ss[0] eq 3) then begin
            image=dataset(idx,*,*)
                return, 1
                endif else begin
                if (g gt ss[4]-1) then g=ss[4]-1
                if (g lt 0) then g=0
                image= dataset(idx,*,*,g)
                return, 1
                endelse
                endif


    if (options eq 2 ) then begin    ; in xz direction
            if (idx gt ss[2]-1 ) then idx=ss[2]-1
            if (idx lt 0 ) then idx=0
                if( ss[0] eq 3) then begin
                image=dataset(*,idx,*)
                 return, 1
            endif else begin
            if (g gt ss[4]-1) then g=ss[4]-1
            if (g lt 0) then g=0
            image=dataset(*,idx,*,g)
            return, 1
            endelse
            endif


return, 0

end


; Widge event handler.
PRO    widgetEvent, sEvent ; event structure.
COMMON SHARE1, image1, image2, image3, trans, transtep, scale, scalestep, $
                                        shear, shearstep, fn1, fn2, xdim, ydim, epi_data, $
                                        ref_data

    ;   Quit the application using the close box.
    ;
    if (TAG_NAMES(sEvent, /STRUCTURE_NAME) EQ $
        'WIDGET_KILL_REQUEST') then begin
        WIDGET_CONTROL, sEvent.top, /DESTROY ;    Get the info structure from top-level
        RETURN ;
    endif

    WIDGET_CONTROL, sEvent.top, GET_UVALUE=sState, /NO_COPY ;
    ;   Determine which event.
    ;
    WIDGET_CONTROL, sEvent.id, GET_UVALUE=eventval ;
    ;   Take the following action based on the corresponding event.
    ;
            temp=image1
    case eventval of ;
    ;
            "EXIT": begin
                        ; Restore the info structure before destroying event.top
                        ;
                        WIDGET_CONTROL, sEvent.top, SET_UVALUE=sState, /NO_COPY
                        ; Destroy widget hierarchy.
                        ;
                        WIDGET_CONTROL, sEvent.top, /DESTROY
                        epi_data=0 ;release the memory before exit.
                        ref_data=0 ;
                        image1=0;
                        image2=0;

                        RETURN
                        end
            "CLOSE": begin
                        ; Restore the info structure before destroying event.top
                        ;
                        WIDGET_CONTROL, sEvent.top, SET_UVALUE=sState, /NO_COPY
                        ; Destroy widget hierarchy.
                        ;
                        WIDGET_CONTROL, sEvent.top, /DESTROY

                        RETURN
            end

            "EPIIMG": begin
                        ; open an image.
                        fn1=DIALOG_PICKFILE(/READ, path='D:¥tensor¥')
        IF (fn1 EQ '') THEN BEGIN
                return
        ENDIF

        WIDGET_CONTROL, /HOURGLASS


        flag=FILE_READY(fn1, epi_data)
        if (flag eq 0 ) then begin
            print, 'EPI image read failed!'
        endif else begin

        ; show the data info.
        ss=size(epi_data)
        temptext=''
        for i=1, ss[0] do temptext=temptext+STRING(ss[i])
        void=DIALOG_MESSAGE(['The dataset dimension is'+STRING(ss[0]), $
                                        'Size of each dimension', $
                                        temptext],/INFORMATION)
        sState.image1_num=0;
        sState.image1_max=ss[3]; set the data information.
        sState.image1_gnum=0;
        sState.image1_gmax=ss[4];
    if (sState.option0 eq 0 ) then begin
     ; void=DIALOG_MESSAGE('You have to change the setting to use different image as reference!')
      sState.image2_num=sState.image1_num
      sState.image2_max=sState.image1_max
      sState.image2_gnum=sState.image1_gnum
      sState.image2_gmax=sState.image1_gmax
    endif
        WIDGET_CONTROL, sState.wImageText1, SET_VALUE=STRING(sState.image1_num, /PRINT)
        WIDGET_CONTROL,     sState.wImageGnum1,     SET_VALUE=STRINg(sState.image1_gnum,
/PRINT)

        ;get the image to be displayed, the flag tells you whether
        ; the operation is successful.
        flag=get_image(epi_data, sState.option5, 0,0, image1)
        ;display the image at window.
        show_image, image1,   sState.drawIndex1, 256,256, sState.option1

        if (sState.option0 eq 0) then begin
        flag=get_image(epi_data, sState.option5, 0,0,image2)
        show_image, image2,   sState.drawIndex2, 256,256, sState.option2
        endif


        WIDGET_CONTROL, sState.wFileLabel1, SET_VALUE=fn1, /NO_COPY

        endelse
        end
"SINGLESHOT": begin
        fn1=''
        fn1=DIALOG_PICKFILE(/READ, FILTER='*.data')
        if( fn1 ne '') then begin
        epi_data=0
        signa_reader, fn1, epi_data

        ; show the data info.

        ss=size(epi_data)
        temptext=''
        for i=1, ss[0] do temptext=temptext+STRING(ss[i])
        void=DIALOG_MESSAGE(['The dataset dimension is'+STRING(ss[0]), $
                                        'Size of each dimension', $
                                        temptext],/INFORMATION)
        sState.image1_num=0;
```

```
            sState.image1_max=ss[3]; set the data information.
            sState.image1_gnum=0;
            sState.image1_gmax=ss[4];
        if (sState.option0 eq 0 ) then begin
         ; void=DIALOG_MESSAGE('You have to change the setting to use different image as reference!')
          sState.image2_num=sState.image1_num
          sState.image2_max=sState.image1_max
          sState.image2_gnum=sState.image1_gnum
          sState.image2_gmax=sState.image1_gmax
        endif
            WIDGET_CONTROL, sState.wImageText1, SET_VALUE=STRING(sState.image1_num, /PRINT)
            WIDGET_CONTROL,       sState.wImageGnum1,       SET_VALUE=STRINg(sState.image1_gnum,
/PRINT)
            ;get the image to be displayed, the flag tells you whether
            ; the operation is successful.
            flag=get_image(epi_data, sState.option5, 0,0, image1)
            ;display the image at window.
            show_image, image1,    sState.drawIndex1, 256,256, sState.option1

            if (sState.option0 eq 0) then begin
            flag=get_image(epi_data, sState.option5, 0,0,image2)
            show_image, image2,    sState.drawIndex2, 256,256, sState.option2
            endif


            WIDGET_CONTROL, sState.wFileLabel1, SET_VALUE=fn1, /NO_COPY
            endif
            end


    "OPEN_SAVED": begin
            fn1=''
            fn1=DIALOG_PICKFILE(/READ,FILTER='*.hdr')
            if( fn1 ne '') then begin
             epi_data=0
            flag=FILE_READ_DATA(fn1, epi_data)
            ; show the data info.

            ss=size(epi_data)
            temptext=''
            for i=1, ss[0] do temptext=temptext+STRING(ss[i])
            void=DIALOG_MESSAGE(['The dataset dimension is'+STRING(ss[0]), $
                                                        'Size of each dimension', $
                                                        temptext])
            sState.image1_num=0;
            sState.image1_max=ss[3]; set the data information.
            sState.image1_gnum=0;
            sState.image1_gmax=ss[4];
        if (sState.option0 eq 0 ) then begin
         ; void=DIALOG_MESSAGE('You have to change the setting to use different image as reference!')
          sState.image2_num=sState.image1_num
          sState.image2_max=sState.image1_max
          sState.image2_gnum=sState.image1_gnum
          sState.image2_gmax=sState.image1_gmax
        endif
            WIDGET_CONTROL, sState.wImageText1, SET_VALUE=STRING(sState.image1_num, /PRINT)
            WIDGET_CONTROL,       sState.wImageGnum1,       SET_VALUE=STRINg(sState.image1_gnum,
/PRINT)
            ;get the image to be displayed, the flag tells you whether
            ; the operation is successful.
            flag=get_image(epi_data, sState.option5, 0,0, image1)
            ;display the image at window.
            show_image, image1,    sState.drawIndex1, 256,256, sState.option1

            if (sState.option0 eq 0) then begin
```

```
                flag=get_image(epi_data, sState.option5, 0,0,image2)
                show_image, image2,    sState.drawIndex2, 256,256, sState.option2
                endif


                WIDGET_CONTROL, sState.wFileLabel1, SET_VALUE=fn1, /NO_COPY
                endif
                end


"REFERENCE": begin
        if (sState.option0 eq 0 ) then begin
            void=DIALOG_MESSAGE('You have to change the setting to use different image as reference!')
            sState.image2_num=sState.image1_num
            sState.image2_max=sState.image1_max
            sState.image2_gnum=sState.image1_gnum
            sState.image2_gmax=sState.image1_gmax
        endif else begin

                ; open an image.
                fn2=DIALOG_PICKFILE(/READ, path='D:¥tensor¥')
                IF (fn2 EQ '') THEN BEGIN
                        return
                ENDIF

                WIDGET_CONTROL, /HOURGLASS
                flag=FILE_READY(fn2, ref_data)
                if(flag eq 0) then begin
                    print, 'REF image read failed!'
                 endif else begin

                ss=size(ref_data)
                temptext=''
                for i=1, ss[0] do temptext=temptext+STRING(ss[i])
                void=DIALOG_MESSAGE(['The dataset dimension is'+STRING(ss[0]), $
                                                'Size of each dimension', $
                                                temptext],/INFORMATION)

                flag=get_image(ref_data, sState.option5, 0,0,image2 )
                show_image, image2, sState.drawIndex2, 256,256, sState.option2
                WIDGET_CONTROL,    sState.wFileLabel2, SET_VALUE=fn2, /NO_COPY

                sState.image2_num=0;
                sState.image2_max=ss[3]
                sState.image2_gnum=0;
                sState.image2_gmax=ss[4];
                WIDGET_CONTROL, sState.wImageText2, SET_VALUE=STRING(sState.image2_num, /PRINT)
                WIDGET_CONTROL,       sState.wImageGnum2,       SET_VALUE=STRINg(sState.image2_gnum,
/PRINT)

                 endelse

                endelse

                end

"STYLELIST1": begin ; select how to display the image.
                 sState.option1=WIDGET_INFO( sEvent.id, /DROPLIST_SELECT)

                if (fn1 EQ '') then begin
                        return
                endif

                show_image, image1,    sState.drawIndex1, 256,256, sState.option1

                end
```

```
"STYLELIST2": begin
                  sState.option2=WIDGET_INFO( sEvent.id, /DROPLIST_SELECT)
                 if(sState.option0 eq 0 ) then begin
                 show_image, image2, sState.drawIndex2, 256,256,sState.option2
                 endif else begin

                 if (fn2 ne '') then begin
                     show_image, image2, sState.drawIndex2, 256,256, sState.option2
                 endif
                 endelse

                 end
"STYLELIST3": begin ; whether use epi image as reference.
                 sState.option0=WIDGET_INFO( sEvent.id, /DROPLIST_SELECT)
                 end

"OPTIONA": begin ;the artifact is in x or y direction.
                  sState.option3=WIDGET_INFO(sEvent.id, /DROPLIST_SELECT)
                 end
"OPTIONB": begin ;show the epi image or epi-ref image.
                  sState.option4=WIDGET_INFO(sEvent.id, /DROPLIST_SELECT)
                 end
"OPTIONC": begin ; show xy,yz,or zx image.
                 image2=0
                  sState.option5=WIDGET_INFO(sEvent.id, /DROPLIST_SELECT)
                 if (sState.option5 eq 0 ) then begin
                                ss1=size(epi_data)
                                ss2=size(ref_data)
                       if( ss1[0] gt 1) then begin

                                flag=get_image(epi_data, sState.option5, 0,sState.image1_gnum,image1)
                                show_image, image1,    sState.drawIndex1, 256,256, sState.option1

                                ss=size(epi_data)
                                sState.image1_num=0
                                sState.image1_max=ss[3]

                        endif
                       if( ss2[0] gt 1) then begin

                                flag=get_image(ref_data, sState.option5, 0,sState.image2_gnum, image2)
                                show_image, image2,    sState.drawIndex2, 256,256, sState.option2


                                sState.image2_num=0
                                sState.image2_max=ss2[3]

                        endif else begin
                            flag=get_image(epi_data, sState.option5, 0, sState.image2_gnum,image2)
                            show_image, image2, sState.drawIndex2, 256,256, sState.option2


                            sState.image2_num=0 ; update the info. about the image.
                            sState.image2_max=ss1[3]

                            endelse

                    endif

                 if (sState.option5 eq 1 ) then begin
                                ss1=size(epi_data)
                                ss2=size(ref_data)
                       if( ss1[0] gt 1) then begin

                                flag=get_image(epi_data, sState.option5, 0,sState.image1_gnum,iamge1)
```

```
                show_image, image1,    sState.drawIndex1, 256,256, sState.option1

                ;ss=size(epi_data)
                sState.image1_num=0
                sState.image1_max=ss1[1]

          endif
         if( ss2[0] gt 1) then begin

                flag=get_image(ref_data, sState.option5, 0,sState.image2_gnum, image2)
                show_image, image2,    sState.drawIndex2, 256,256, sState.option2

         ;      ss=size(ref_data)
                sState.image2_num=0
                sState.image2_max=ss2[1]

          endif else begin
             flag=get_image(epi_data, sState.option5, 0, sState.image2_gnum,image2)
             show_image, image2, sState.drawIndex2, 256,256, sState.option2

             ;ss=size(epi_data)
             sState.image2_num=0
             sState.image2_max=ss1[1]

             endelse
      endif

    if (sState.option5 eq 2 ) then begin
                ss1=size(epi_data)
                ss2=size(ref_data)
         if( ss1[0] gt 1) then begin

                flag=get_image( epi_data, sState.option5, 0,sState.image1_gnum,image1)
                show_image, image1,    sState.drawIndex1, 256,256, sState.option1

                ;ss=size(epi_data)
                sState.image1_num=0
                sState.image1_max=ss1[2]

          endif
         if( ss2[0] gt 1) then begin

                flag=get_image(ref_data, sState.option5, 0,sState.image2_gnum,image2)
                show_image, image2,    sState.drawIndex2, 256,256, sState.option2

                ;ss=size(ref_data)
                sState.image2_num=0
                sState.image2_max=ss2[2]

          endif else begin
             flag=get_image(epi_data, sState.option5, 0, sState.image2_gnum, image2)
             show_image, image2, sState.drawIndex2, 256,256, sState.option2

           ; ss=size(epi_data)
             sState.image2_num=0
             sState.image2_max=ss1[2]

             endelse
      endif

     WIDGET_CONTROL, sState.wImageText1, SET_VALUE=STRING(sState.image1_num, /PRINT)
     WIDGET_CONTROL, sState.wImageText2, SET_VALUE=STRING(sState.image2_num, /PRINT)
    end

"IMAGE1_DEC": begin ; show the previous image.
```

```
                        sState.image1_num=sState.image1_num-1
                        if (sState.image1_num lt 0 ) then begin
                              sState.image1_num=0
                        endif
                        WIDGET_CONTROL, sState.wImageText1, SET_VALUE=STRING(sState.image1_num, /PRINT)
                           flag=get_image(epi_data, sState.option5, sState.image1_num, sState.image1_gnum, image1)
                           show_image, image1, sState.drawIndex1, 256,256, sState.option1

                        end
   "IMAGE1_INC": begin ; show the next image.
                       sState.image1_num=sState.image1_num+1
                       if (sState.image1_num gt    sState.image1_max-1) then begin
                                 sState.image1_num=sState.image1_max
                       endif
                        WIDGET_CONTROL, sState.wImageText1, SET_VALUE=STRING(sState.image1_num, /PRINT)
                           flag=get_image(epi_data, sState.option5, sState.image1_num, sState.image1_gnum,image1)
                           show_image, image1, sState.drawIndex1, 256,256, sState.option1

                       end


"IMAGE2_DEC": begin
                        sState.image2_num=sState.image2_num-1
                        if (sState.image2_num lt 0 ) then begin
                              sState.image2_num=0
                        endif
                        WIDGET_CONTROL, sState.wImageText2, SET_VALUE=STRING(sState.image2_num, /PRINT)
                           if( sState.option0 eq 0) then    begin
                           flag=get_image(epi_data, sState.option5, sState.image2_num, sState.image2_gnum,image2)
                           endif else begin
                           image2=0
                           flag=get_image(ref_data, sState.option5, sState.image2_num, sState.image2_gnum,image2)
                           endelse

                           show_image, image2, sState.drawIndex2, 256,256, sState.option2

                        end
   "IMAGE2_INC": begin
                       sState.image2_num=sState.image2_num+1
                       if (sState.image2_num gt    sState.image2_max-1) then begin
                                 sState.image2_num=sState.image2_max
                       endif
                        WIDGET_CONTROL, sState.wImageText2, SET_VALUE=STRING(sState.image2_num, /PRINT)

                        if( sState.option0 eq 0) then    begin
                           flag=get_image(epi_data, sState.option5, sState.image2_num, sState.image2_gnum,image2)
                           endif else begin
                           image2=0
                           flag=get_image(ref_data, sState.option5, sState.image2_num, sState.image2_gnum,image2)
                           endelse

                           show_image, image2, sState.drawIndex2, 256,256, sState.option2
                       end


 "GOTO_1": begin

                                 WIDGET_CONTROL, sState.wImageText1, Get_VALUE=tmp_text, $
                                           /NO_COPY
                                 tmp=UINT(tmp_text)
                                 if (tmp[0] lt 0) then begin
                                           tmp=0
                                 endif
```

```
                    if (tmp[0] gt sState.image1_max ) then begin
                                tmp=sState.image1_max
                    endif
                    sState.image1_num=tmp

                    WIDGET_CONTROL, sState.wImageGnum1, Get_VALUE=tmp_text, $
                                /NO_COPY
                    tmp=UINT(tmp_text)
                    if (tmp[0] lt 0) then begin
                                tmp=0
                    endif
                    if (tmp[0] gt sState.image1_gmax ) then begin
                                tmp=sState.image1_gmax
                    endif
                    sState.image1_gnum=tmp

        flag=get_image(epi_data, sState.option5,sState.image1_num , sState.image1_gnum,image1)
                show_image, image1, sState.drawIndex1, 256,256, sState.option1
                end

"GOTO_2": begin

                    WIDGET_CONTROL, sState.wImageText2, Get_VALUE=tmp_text, $
                                /NO_COPY
                    tmp=UINT(tmp_text)
                    if (tmp[0] lt 0) then begin
                                tmp=0
                    endif
                    if (tmp[0] gt sState.image2_max ) then begin
                                tmp=sState.image2_max
                    endif
                    sState.image2_num=tmp

                    WIDGET_CONTROL, sState.wImageGnum2, Get_VALUE=tmp_text, $
                                /NO_COPY
                    tmp=UINT(tmp_text)
                    if (tmp[0] lt 0) then begin
                                tmp=0
                    endif
                    if (tmp[0] gt sState.image2_gmax ) then begin
                                tmp=sState.image2_gmax
                    endif
                    sState.image2_gnum=tmp

        if (sState.option0 eq 0) then begin
        flag=get_image(epi_data, sState.option5, sState.image2_num, sState.image2_gnum,image2)
        endif else begin
        flag=get_image(ref_data, sState.option5, sState.image2_num, sState.image2_gnum,image2)
        endelse

                show_image, image2, sState.drawIndex2, 256,256, sState.option2
                end

  "TRAN_INC": begin
            trans=trans+transtep;
            WIDGET_CONTROL , sState.wCtrlText1, SET_VALUE=STRING(trans, /PRINT)
            end;
  "TRAN_DEC": begin
            trans=trans-transtep;
            WIDGET_CONTROL,    sState.wCtrlText1, SET_VALUE=STRING(trans, /PRINT)
            end;
  "SCALE_INC": begin
            scale=scale+scalestep;
            WIDGET_CONTROL,    sState.wCtrlText2, SET_VALUE=STRING(scale, /PRINT)
            end;
```

```
"SCALE_DEC": begin
                scale=scale-scalestep;
                WIDGET_CONTROL,    sState.wCtrlText2, SET_VALUE=STRING(scale, /PRINT)
                end;
"SHEAR_INC": begin
                shear=shear+shearstep;
                WIDGET_CONTROL,    sState.wCtrlText3, SET_VALUE=STRING(shear, /PRINT)
                end;
"SHEAR_DEC":begin
                shear=shear-shearstep;
                WIDGET_CONTROL,    sState.wCtrlText3, SET_VALUE=STRING(shear, /PRINT)
                end;

"APPLY": begin

                WIDGET_CONTROL,    sState.wCtrlText1, Get_VALUE=tmp_text, /NO_COPY
                tmp=FLOAT(tmp_text)
                trans=tmp[0]

                WIDGET_CONTROL,    sState.wCtrlText2, Get_VALUE=tmp_text, /NO_COPY
                tmp=FLOAT(tmp_text)
                scale=tmp[0]

                WIDGET_CONTROL,    sState.wCtrlText3, Get_VALUE=tmp_text, /NO_COPY
                tmp=FLOAT(tmp_text)
                shear=tmp[0]

                print, trans, scale, shear

                image3=CORRECTION(image1, trans,scale,shear,sState.option3)


                if ( sState.option4 eq 0) then begin
                wset,    sState.drawIndex3 & tvscl, congrid(image3, 360,360,1)
                endif
                if(    sState.option4 eq 1) then begin

                max_t=max(image3)
                max_r=max(image2)
                paren_t=where(image3 gt max_t/6)
                paren_r=where(image2 gt max_r/6)
                t_img=intarr(128,128)          ;tensor_imgaes
                r_img=intarr(128,128)          ;ref_images
                gap_img=intarr(128,128)
                temp_match=intarr(128,128)
                t_img(paren_t)=-1              ;tensor_imgaes
                r_img(paren_r)=1               ;ref_images
                temp_match=t_img+r_img
                mismatch=where(temp_match eq -1)          ;residual_timage
                gap_img(mismatch)=255B
                wset,    sState.drawIndex3 & tvscl, congrid(gap_img,360,360,1)
                endif

                if(    sState.option4 eq 2) then begin

                max_t=max(image3)
                max_r=max(image2)
                paren_t=where(image3 gt max_t/6)
                paren_r=where(image2 gt max_r/6)
                t_img=intarr(128,128)          ;tensor_imgaes
                r_img=intarr(128,128)          ;ref_images
                gap_img=intarr(128,128)
                temp_match=intarr(128,128)
                t_img(paren_t)=-1
                r_img(paren_r)=1
```

```
                        temp_match=t_img+r_img
                        mismatch=where(temp_match eq 1)                  ;residual_rimage
                        gap_img(mismatch)=255B
                        wset,   sState.drawIndex3 & tvscl, congrid(gap_img,360,360,1)
                        endif

                        if( sState.option4 eq 3 ) then begin
                        show_image2, image3, image2, sState.drawIndex3,360,360,2,1
                        endif

                        if( sState.option4 eq 4 ) then begin
                        show_image2, image3, image2, sState.drawIndex3,360,360,1,1
                        endif

                        result=DIALOG_MESSAGE('Do you want to apply the change to all image', /QUESTION)
                        if( result eq 'Yes') then begin
                                for i=0, sState.image1_max-1 do begin
                                        flag=get_image(epi_data, sState.option5, i, $
                                            sState.image1_gnum,imaget)
                                        imaget=CORRECTION(imaget, trans, scale, shear, sState.option3)
                                        flag=put_image(epi_data, sState.option5, i, $
                                                sState.image1_gnum, imaget)
                                end
                        endif

                        end;
                "SAVE": begin
                        output_fn=''
                        output_fn=fn1
                        output_fn=DIALOG_PICKFILE(path='/usr/people/itoh/data/tensor', $
                        title='just path to directory for saving, not file name.')
                        if( output_fn ne '') then begin
                        flag=FILE_SAVEY(epi_data,output_fn)
                        endif
                        end;




        ELSE: begin
                end
        endcase

        WIDGET_CONTROL, sEvent.top, Set_UValue=sState, /No_Copy

                end

PRO corr_w

COMMON SHARE1, image1, image2, image3,trans, transtep, scale, scalestep, $
                                shear, shearstep, fn1, fn2, xdim, ydim, epi_data, $
                                ref_data

; Initialize global vars.
epi_data=0;
ref_data=0;
xdim=256
ydim=256
image1=bytarr(xdim,ydim)
image2=bytarr(xdim,ydim)
image3=bytarr(xdim,ydim)
trans=0
transtep=1
scale=1
```

```
scalestep=0.1
shear=0
shearstep=0.1
fn1=''
fn2=''
DEVICE, DECOMPOSED=0
; Get the screen size,
            Device, GET_SCREEN_SIZE=screenSize
        xdim=screenSize[0]*0.4
        ydim=xdim*0.75

;make the system have a maximum of 256 colors.

    numcolors = !d.N_COLORS
    if( (( !D.NAME EQ 'X') or (!D.NAME EQ 'MAC')) $
        and (!d.N_COLORS GE 256L)) then $
        DEVICE, PSEUDO_COLOR=8

    DEVICE, DECOMPOSED=0, BYPASS_TRANSLATION=0

      Get the current color table

    TVLCT, savedR, savedG, savedB, /GET

      Build color table from color vectors

    colorTable = [[savedR],[savedG],[savedB]]


            ; Define a main widget base.

            w_base=WIDGET_BASE(TITLE="Eddy Corrector",$
                                    /TLB_KILL_REQUEST_EVENTS, $
                                    MBAR=barBase)


            ; Create the File menu.
            wFileButton=WIDGET_BUTTON(w_base, VALUE='File', /MENU)
            wImgFileButton1=WIDGET_BUTTON(wFileButton, $
                                    VALUE='Open EPI Image', UVALUE='EPIIMG')
            wImgFileButton2=WIDGET_BUTTON(wFileButton, $
                                    VALUE='Open Ref Image', UVAIUE='REFERENCE')
            wImgFileButton3=WIDGET_BUTTON(wFileButton, $
                                    VALUE='Open saved image', UVALUE='OPEN_SAVED')
            wImgFileButton4=WIDGET_BUTTON(wFileButton, $
                                    VALUE='Open Single Shot data', UVALUE='SINGLESHOT')
            wQuitButton=WIDGET_BUTTON(wFileButton, VALUE='Exit', UVALUE='EXIT')

            ;Create the working area.
            wSubBase=WIDGET_BASE(w_base, COLUMN=2)

            ; Create a base for the left column.
            wLeftBase=WIDGET_BASE(wSubBase, /BASE_ALIGN_LEFT, $
                        XPAD=5, YPAD=5,/FRAME, /COLUMN)

            wFileLabel1=WIDGET_LABEL(wLeftBase, VALUE='No image loaded')
            wStyleDroplist1=WIDGET_DROPLIST(wLeftBase, VALUE=['Grayscale',$
                        'Boundary'], UVALUE='STYLELIST1')
            wStyleDropList3=WIDGET_DROPLIST(wLeftBase, VALUE=['Use same EPI image as Ref', $
                                                'Open other image'], UVALUE='STYLELIST3')
            wFileLabel2=WIDGET_LABEL(wLeftBase, VALUE='No image loaded')
            wStyleDropList2=WIDGET_DROPLIST(wLeftBase, VALUE=['Grayscale',$
                        'Boundary'], UVALUE='STYLELIST2')

            ;Greate the control for correction.
```

```
wLeft_sub_Base=WIDGET_BASE(wLeftBase, /BASE_ALIGN_LEFT, /ROW, /FRAME)
wOptionDropList1=WIDGET_DROPLIST(wLeft_sub_Base, VALUE=['X','Y'], $
                                          UVALUE='OPTIONA')
wOptionDropList2=WIDGET_DROPLIST(wLeft_sub_Base, Value=['Tensor Image Only', $
                                          'Tensor Distortion', 'Ref Contour','Contour Overlay', $
                                          'Ref Contour on Image'], UVALUE='OPTIONB')
wOptionDropList3=WIDGET_DROPLIST(wLeft_sub_Base, Value=['XY Image', $
                                          'YZ Image', 'XZ Image'], UVALUE='OPTIONC')

wPanel1=WIDGET_BASE(wLeftBase,/BASE_ALIGN_LEFT, $
            /FRAME, COLUMN=4)

wCtrlLabel1=WIDGET_LABEL(wPanel1,VALUE='Translation')
wCtrlInc1=WIDGET_BUTTON(wPanel1,VALUE='+',UVALUE='TRAN_INC')
wCtrlDec1=WIDGET_BUTTON(wPanel1,VALUE='-',UVALUE='TRAN_DEC')
wCtrlText1=WIDGET_TEXT(wPanel1,/EDITABLE,UVALUE='TEXT1',$
                                          VALUE='0',    /ALL_EVENTS)

wPanel2=WIDGET_BASE(wLeftBase,/BASE_ALIGN_LEFT, $
            /FRAME, COLUMN=4)
wCtrlLabel2=WIDGET_LABEL(wPanel2,VALUE='Scaling')
wCtrlInc2=WIDGET_BUTTON(wPanel2,VALUE='+',UVALUE='SCALE_INC')
wCtrlDec2=WIDGET_BUTTON(wPanel2,VALUE='-',UVALUE='SCALE_DEC')
wCtrlText2=WIDGET_TEXT(wPanel2,/EDITABLE,UVALUE='TEXT2',$
                                          VALUE='0', /ALL_EVENTS)

wPanel3=WIDGET_BASE(wLeftBase,/BASE_ALIGN_LEFT, $
            /FRAME, COLUMN=4)
wCtrlLabel3=WIDGET_LABEL(wPanel3,VALUE='Shear')
wCtrlInc3=WIDGET_BUTTON(wPanel3,VALUE='+',UVALUE='SHEAR_INC')
wCtrlDec3=WIDGET_BUTTON(wPanel3,VALUE='-',UVALUE='SHEAR_DEC')
wCtrlText3=WIDGET_TEXT(wPanel3,/EDITABLE,UVALUE='TEXT3',$
                                          VALUE='0', /ALL_EVENTS)

; Then place the draw widget.
w_left_drawbase=WIDGET_BASE(wLeftBase, /BASE_ALIGN_CENTEr, /COLUMN)
w_draw_1=WIDGET_DRAW(w_left_drawbase, XSIZE=256, YSIZE=256, $
              /EXPOSE_EVENTS, UVALUE='DRAW1', RETAIN=2)

;
; Then the image control
w_Image_Panel1=WIDGET_BASE(w_left_drawbase, /BASE_ALIGN_CENTER, /FRAME, $
                                          /ROW)
w_Image1_Btn1=WIDGET_BUTTON(w_Image_Panel1, VALUE='-', UVALUE='IMAGE1_DEC')
w_Image_Num1=WIDGET_TEXT(w_Image_Panel1, /EDITABLE, UVALUE='IMAGE1_NUM',$
                                                      VALUE='0',     /ALL_EVENTS)
w_Image1_Btn2=WIDGET_BUTTON(w_Image_Panel1, VALUE='+', UVALUE='IMAGE1_INC')
wCtrlLabel3=WIDGET_LABEL(w_left_drawbase, VALUE='Gradient Selection')
w_Image_gnum1=WIDGET_TEXT(w_left_drawbase, /EDITABLE, /ALL_EVENTS, $
                                                      UVALUE='IMAGE1_GNUM', VALUE='0')
w_gobtn1=WIDGET_BUTTON(w_left_drawbase, VALUE='Go...', UVALUE='GOTO_1')

; Buttons on the right.
wRightBase=WIDGET_BASE(wSubBase,/BASE_ALIGN_CENTER,$
            XPAD=5, YPAD=5, /FRAME,    /COLUMN)

wRef_Base=WIDGET_BASE(wRightBase,/BASE_ALIGN_CENTER, /ROW)
w_draw_2=WIDGET_DRAW(wRef_Base, XSIZE=256, YSIZE=256, $
              /EXPOSE_EVENTS, UVALUE='DRAW2', RETAIN=2)

w_Image_Panel20=WIDGET_BASE(wRef_Base, /BASE_ALIGN_CENTER, /FRAME, $
                                          /COLUMN)
w_Image_Panel2=WIDGET_BASE(w_Image_Panel20, /BASE_ALIGN_CENTER, /FRAME, $
                                          /ROW)
w_Image2_Btn1=WIDGET_BUTTON(w_Image_Panel2, VALUE='-', UVALUE='IMAGE2_DEC')
```

```
    w_Image_Num2=WIDGET_TEXT(w_Image_Panel2, /EDITABLE, UVALUE='IMAGE2_NUM',$
                                                          VALUE='0',   /ALL_EVENTS)
    w_Image2_Btn2=WIDGET_BUTTON(w_Image_Panel2, VALUE='+', UVALUE='IMAGE2_INC')
    wCtrlLabel4=WIDGET_LABEL(w_Image_Panel20, VALUE='Gradient Selection')
    w_Image_gnum2=WIDGET_TEXT(w_Image_Panel20, /EDITABLE, /ALL_EVENTS, $
                                              UVALUE='IMAGE2_GNUM', VALUE='0')
    w_gobtn2=WIDGET_BUTTON(w_Image_Panel20, VALUE='Go...', UVALUE='GOTO_2')

    w_draw_3=WIDGET_DRAW(wRightBase, XSIZE=360, YSIZE=360, $
                   /EXPOSE_EVENTS, UVALUE='DRAW3', RETAIN=2)
    wCtrlPanel4=WIDGET_BASE(wRightBase, /FRAME, /ROW)
    wBtn_Undo=WIDGET_BUTTON(wCtrlPanel4,VALUE='Undo', UVALUE='UNDO')
    wBtn_Save=WIDGET_BUTTON(wCtrlPanel4,VALUE='Save',UVALUE='SAVE')
    wBtn_Apply=WIDGET_BUTTON(wCtrlPanel4,VALUE='Apply Changes',UVALUE='APPLY')
    wBtn_Close=WIDGET_BUTTON(wCtrlPanel4,VALUE='Close',UVALUE='CLOSE')


    WIDGET_CONTROL, w_base, /REALIZE
; Create the info structure.
    ;
    ; Get the index of the graphic window
    WIDGET_CONTROL, w_draw_1, GET_VALUE=drawIndex1
    WIDGET_CONTROL, w_draw_2, GET_VALUE=drawIndex2
    WIDGET_CONTROL, w_draw_3, GET_VALUE=drawIndex3

    sState= { $
             BtnDown: 0,   $                        ;Mouse button down flag.
             option0:  0,  $     ; whether use the same image as EPI and REF or not.
             option1: 0,  $      ; image display option for the epi iamge
             option2:  0,  $  ; image display option for the ref image.
             option3: 0,  $  ;   the artifact is in x or y direction
             option4:  0,  $ ; display epi or epi-ref image in window 3.
             option5:  0,  $  ; show xy, yz or zx image.
             image1_num:  0, $ ; current image 1 slice number.
             image1_max:  0, $  ; maximum image 1 slice number.
             image1_gnum:  0, $  ; image #1 gradient number.
             image1_gmax:    0, $  ; maximum gradient number.
             image2_num:    0,  $
             image2_max:   0,  $
             image2_gnum:    0,  $
             image2_gmax:      0,  $
             draw_1:  w_draw_1, $
             draw_2:  w_draw_2,  $
             draw_3:  w_draw_3,  $
             drawIndex1: drawIndex1,  $
             drawIndex2:  drawIndex2,  $
             drawIndex3:  drawIndex3,  $
             wFileLabel1: wFileLabel1, $
             wFileLabel2: wFileLabel2, $
             wCtrlText1: wCtrlText1, $
             wCtrlText2: wCtrlText2, $
             wCtrlText3: wCtrlText3, $
             wImageText1: w_Image_Num1, $
             wImageGnum1: w_Image_gnum1, $
             wImageGnum2: w_Image_gnum2, $
             wImageText2: w_Image_Num2, $
             wStyleDropList1: wStyleDroplist1, $
             wStyleDropList2: wStyleDroplist2, $
             wStyleDropList3: wStyleDroplist3   $
             }

    WIDGET_CONTROL, w_base, SET_UVALUE=sState
    WIDGET_CONTROL, w_base, SENSITIVE=1
    WIDGET_CONTROL, w_base, MAP=1
```

```
XMANAGER, "corr_epi", w_base, /NO_BLOCK, $
                EVENT_HANDLER="widgetEvent"
end
```

APPENDIX 3

```
pro ten_m

;This program is a main driver to process the diffusion images
;tensor_im1 + bunch of anisotropy indicis
;tensor_im for multislice
;difference from tensor_ms is that this has image quality check routine
;tensor_ms4 + philips default data acquisition scheme. All direction in one file
;tensor_ms5 modified by Itoh     Sep 27/1999
;               addition         data_output part for Ei images
;tensor_ms6 modified by Itoh     Sep 28/1999
;               addition         data_output part for trace images
;               elimination                    some data_output parts
;tensor_ms_ge.pro, modified by Itoh     July27/2002
;               for corrected GE
;tensor_ms_ge.pro + ani_sat=fix.pro by Itoh    Aug14/2002
;
;It has not checked whether this works correctly or not yet.
;fixed bugs and checked working by Itoh, Sep7/2002
;for MacOSX by Itoh, Oct5/2003


;************************  input data files  ************************
damy=' '
datafile=''
get_filenames:
;read,'input the name of datafile: ',datafile
datafile =dialog_pickfile(/read, path='/Users/idl/data/tensor/')
;get data
get_data:

get_lun,unit
openr, unit,datafile

readf,unit,damy
readf,unit,damy
readf,unit,damy
if (damy ne 'number_of_files') then goto, error
print,'number_of_series OK'
readf,unit,num_files
readf,unit,damy
readf,unit,damy
if (damy ne 'cb') then goto, error
print,'cb OK'
readf,unit,cb
readf,unit,damy
readf,unit,damy
if (damy ne 'nb') then goto, error
print,'nb OK'
readf,unit,nb
readf,unit,damy
readf,unit,damy
if (damy ne 'nc') then goto, error
print,'nc OK'
readf,unit,nc
readf,unit,damy
readf,unit,damy
if (damy ne 'g_length') then goto, error
```

```
print,'filename OK'
readf,unit,gl
readf,unit,damy
readf,unit,damy
if (damy ne 'dif_time') then goto, error
print,'filename OK'
readf,unit,dt
readf,unit,damy
readf,unit,damy
if (damy ne 'number_of_pair') then goto, error
print,'filename OK'
readf,unit,pair
readf,unit,damy

readf,unit,damy
if (damy ne 'b_values') then goto, error
print,'b_values    OK'
gra=fltarr(3)
b=fltarr(3,7*num_files)

;***************** calculation of b-value array *******************

for i=0,6 do begin

   readf,unit,'gradient(G/m) (x,y,z)=',gra
   b(0,i)=pair*gra(0)*(gl/1000*26.75*10^3)*sqrt(dt/1000-gl/3000)
   b(1,i)=pair*gra(1)*(gl/1000*26.75*10^3)*sqrt(dt/1000-gl/3000)
   b(2,i)=pair*gra(2)*(gl/1000*26.75*10^3)*sqrt(dt/1000-gl/3000)

end

;btemp = b

;for j=0, 6 do begin
;for i=0,num_files-1 do begin

;b(*,i+j*(num_files)) = btemp(*,j)

;endfor
;endfor

echo_b:
print, ' Echoing b-factors: '
for index=0, 6 do print,'b values are = ', b(*,index)

print,''

close, unit
free_lun, unit


;************** data processing I : threed transform    ***************

tdim=fltarr(cb,nb,nc,num_files*7)

;read,'do you want to go 3d vector presentation ? y(1) ;',ans
;if (ans eq 1) then goto, threecal

read,'do you have image matrix already ? y(1) ;',ans
if (ans eq 1) then begin
           ;tdim=1
           call_g,cb,nb,nc,damat
           damat=reform(temporary(damat),cb,nb,nc,num_files*7)
           tdim(*,*,*,*)=damat(*,*,*,*)
           goto, menu
```

```
endif


for i = 0, num_files -1 do begin

matrix = intarr(cb,nb,nc*8)

get_lun,unit

openr, unit, '/usr/people/itoh/data/' + filename(i)

for index=0,8*nc-1 do begin
d=assoc(unit,intarr(cb,nb,/nozero))
matrix(*,*,index)=d(index)
end
matrix=reform(matrix,cb,nb,8,nc)


close, unit
free_lun, unit

matrix = swap_endian(matrix)

                for j = 0, 7 do begin

                if j ne 7 then tdim(*,*,*,j*num_files+i) = matrix(*,*,j,*)
                if j eq 7 then tdim(*,*,*,(j-1)*num_files+i) = matrix(*,*,j,*)

                endfor


endfor

matrix = 1
;tdim=shift(tdim,0,0,nc/2,0)

damyimage=fltarr(cb,nb)
ra_matrix = bytarr(nc,num_files*7)
ra_matrix(*,*)=1

skip_ra:
; ************* menu ****************

menu:
read,'MENU : 1->Display images, .run , 3->end ,4->adjct, 5->save , 6->threed calculation     :', select

if (select eq 2) then goto, calc
if (select eq 3) then goto, end_of_program
if (select eq 4) then goto, adjct
if (select eq 5) then goto, save_it
if (select eq 6) then goto, threecal


;***************** calculation of tensor ********************

calc:

slice_no:

read,'which direction? x(1)/y(2)/z(3) : ',direction

if (direction eq 0) then goto, menu

window,0,xsize=512,ysize=512
```

```
if (direction eq 1) then begin
                tvscl,congrid(reform(tdim(*,*,nc/2-4,1)),512,512)
                nxe=512/cb
                nye=512/nb
                nxe256=256/nb
                nye256=256/nc
endif
if (direction eq 2) then begin
                tvscl,congrid(reform(tdim(*,*,nc/2-4,1)),512,512)
                nxe=512/cb
                nye=512/nb
                nxe256=256/cb
                nye256=256/nc
endif
if (direction eq 3) then begin
                tvscl,congrid(reform(tdim(*,nb/2-4,*,1)),512,512)
                nxe=512/cb
                nye=512/nc
                nxe256=256/cb
                nye256=256/nb
endif

print,'indicate the slice you want'
cursor,x,y,/device

if (direction eq 1) then i=fix(x/nxe)

if (direction eq 2) then i=fix(y/nye)

if (direction eq 3) then i=fix(y/nye)


if (direction eq 1) then begin
                im=reverse(reform(tdim(i,*,*,*)),2)
                n1=nb
                n2=nc
endif
if (direction eq 2) then begin
                im=reverse(reform(tdim(*,i,*,*)),2)
                n1=cb
                n2=nc
endif
if (direction eq 3) then begin
                im=reform(tdim(*,*,i,*))
                n1=cb
                n2=nb
endif

slice = i

; *************** get noise values ***************

window,0,xsize=512,ysize=512
tvscl,congrid(im(*,*,1),512,512)

print,'please input the region of intrest'

box_cursor,x,y,ax,by
x=round(x/(512/n1))
y=round(y/(512/n2))
ax=round(ax/(512/n1))
by=round(by/(512/n2))

av=total(im(x:x+ax,y:y+by,1))/(ax*by)
print,av
```

```
;*************** fitting ********************

d_matrix = fltarr(6,n1,n2)
b_matrix = fltarr(6,num_files*7)
a0_matrix = fltarr(n1,n2)
lin_idx = fltarr(n1,n2)
pla_idx = fltarr(n1,n2)
sph_idx = fltarr(n1,n2)
ani_idx = fltarr(n1,n2)
max_d_ele = fltarr(n1,n2)
med_d_ele = fltarr(n1,n2)
min_d_ele = fltarr(n1,n2)
matvr = fltarr(n1,n2)
matddd= fltarr(n1,n2)
matfa = fltarr(n1,n2)
matra = fltarr(n1,n2)

tensor_data=fltarr(4,n1,n2)

for i = 0, 7*num_files-1 do begin
            b_matrix(0,i) = b(0,i)^2
            b_matrix(1,i) = b(1,i)^2
            b_matrix(2,i) = b(2,i)^2
            b_matrix(3,i) = 2*b(0,i)*b(1,i)
            b_matrix(4,i) = 2*b(0,i)*b(2,i)
            b_matrix(5,i) = 2*b(1,i)*b(2,i)
endfor

zero_these = where(im(*,*,num_files*3-1) lt av*3.0)

for index = 0, 7*num_files-1 do begin
temp = im(*,*,index)
temp(zero_these) = 0
tdim(*,*,index) = temp
endfor

omit_point = where(ra_matrix(slice,*) eq 0)
if (omit_point(0) ne -1) then begin
            b_matrix(*,omit_point) = 100000000000000
endif

for j=0, n2-1 do begin
            for i = 0, n1-1 do begin

test_array = where(im(i,j,*) eq 0)

                        if test_array(0) ne -1 then begin
                        d_matrix(*,i,j) = [0,0,0,0,0,0]
                        a0_matrix(i,j) = 0


            ;******* following values are set to zero ******

            tensor_data(0,i,j) = 0
            tensor_data(1,i,j) = [0,0,0]
            lin_idx(i,j) = 0
            pla_idx(i,j) = 0
            sph_idx(i,j) = 0
            ani_idx(i,j) = 0

            ;**********************************************
                        goto, skip
                        endif
```

```
data_array = tdim(i,j,slice,*)
if (omit_point(0) ne -1) then begin
                data_array(omit_point) = 0.000000000000000000001
                check = 1
endif

                d=regress(-b_matrix,alog(reform(data_array)),(reform(data_array))/(av),yfit,a0)
                d=regress(-b_matrix,alog(reform(im(i,j,*))),alog(reform(im(i,j,*)))/alog(av),yfit,a0)
                d=regress(-b_matrix,alog(reform(im(i,j,*))),replicate(1.0, n_elements(im(i,j,*))),yfit,a0,/relative_weight)


                d_matrix(*,i,j) = d      ;dif tensor before diagonalize
                a0_matrix(i,j) = exp(a0)      ;a0 image

                ;******* diagonalizatin ********
                d_tensor=[[d(0),d(3),d(4)],[d(3),d(1),d(5)],[d(4),d(5),d(2)]]
                print,d_tensor
                nr_tred2,d_tensor,d_out,e
                nr_tqli,d_out,e,d_tensor
                print,'d_out = ',d_out,' d_tensor = ',d_tensor
                read,'ok?',damy
                ;******* sorting *******
                max_d_ele(i,j) = max(d_out,maxi)
                min_d_ele(i,j) = min(d_out)
                med_d_ele(i,j) = median(d_out)
                v_direction = d_tensor(*,maxi)
                trace_val = (d_out(0)+d_out(1)+d_out(2))
                ;******* data for vector presentation *****
                tensor_data(0,i,j) = max_d_ele(i,j)
                tensor_data(1,i,j) = v_direction
                ;******* anisotropy indecies ******
                lin_idx(i,j) = (max_d_ele(i,j) - med_d_ele(i,j))/trace_val
                pla_idx(i,j) = 2*(med_d_ele(i,j) - min_d_ele(i,j))/trace_val
                sph_idx(i,j) = 3*min_d_ele(i,j)/trace_val
                ani_idx(i,j) = 1-sph_idx(i,j)

ani_idx(i,j)=(max_d_ele(i,j)-0.5*(min_d_ele(i,j)+med_d_ele(i,j)))/trace_val
if (ani_idx(i,j) gt 1.0) then ani_idx(i,j) = 1.0
if (ani_idx(i,j) lt 0) then ani_idx(i,j) = 0

i1=d_out(0)+d_out(1)+d_out(2)
i2=d_out(0)*d_out(1)+d_out(0)*d_out(2)+d_out(1)*d_out(2)
i3=d_out(0)*d_out(1)*d_out(2)
i4=i1^2.0-2.0*i2
dav=i1/3.0
dsurf=sqrt(i2/3.0)
dvol=i3^(1.0/3.0)
dmag=sqrt(i4/3.0)
matvr(i,j)=(dvol/dav)^3.0
if (matvr(i,j) gt 1.0) then matvr(i,j) = 1.0
if (matvr(i,j) lt 0) then matvr(i,j) = 0

matddd(i,j)=2.0*(dmag^2-dvol^2)
if (matddd(i,j) gt 1.0) then matddd(i,j) = 1.0
if (matddd(i,j) lt 0) then matddd(i,j) = 0

matfa(i,j)=sqrt(1.0-(dsurf^2.0)/(dmag^2.0))
if (matfa(i,j) gt 1.0) then matfa(i,j) = 1.0
if (matfa(i,j) lt 0) then matfa(i,j) = 0

                        skip:

                endfor
endfor
```

```
traceim=reform((d_matrix(0,*,*)+d_matrix(1,*,*)+d_matrix(2,*,*))/3)
lin_idx(abs(where(lin_idx lt 0)))=0
pla_idx(abs(where(pla_idx lt 0)))=0
sph_idx(abs(where(sph_idx lt 0)))=0
ani_idx(abs(where(ani_idx lt 0)))=0


window,0,ysize=768,xsize=1024
tvscl,congrid(a0_matrix,256,256),0
tvscl,congrid(reform(d_matrix(0,*,*)),256,256),1
tvscl,congrid(reform(d_matrix(1,*,*)),256,256),2
tvscl,congrid(reform(d_matrix(2,*,*)),256,256),3
tvscl,congrid(reform(d_matrix(3,*,*)),256,256),4
tvscl,congrid(reform(d_matrix(4,*,*)),256,256),5
tvscl,congrid(reform(d_matrix(5,*,*)),256,256),6
tvscl,congrid(traceim,256,256),7
tvscl,congrid(matvr,256,256)<0.5,8
tvscl,congrid(matddd,256,256)<0.5,9
tvscl,congrid(matfa,256,256)<0.5,10
tvscl,congrid(ani_idx,256,256)<0.5,11

read,'go ahead for tensor presentation? y(1) : ',ans
if (ans eq 1) then goto, tensor_pre

goto,calc

;*** get noise value***

window,0,xsize=512,ysize=512
tvscl,congrid(tdim(*,*,nc/2,num_files-1),512,512)

print,'please input the region of interest'

box_cursor,x,y,ax,by
x=round(x/(512/cb))
y=round(y/(512/nb))
ax=round(ax/(512/cb))
by=round(by/(512/nb))

av=total(tdim(x:x+ax,y:y+by,nc/2,1))/(ax*by)
print,av

;***********just for 121497 data**
;av=50000
;***********just for 121497 data**
check = 0
;*************** fitting *********************

d_matrix = fltarr(6,cb,nb,nc)
b_matrix = fltarr(6,num_files*7)
a0_matrix = fltarr(cb,nb,nc)
v_d = fltarr(3,cb,nb,nc)
ani_idx = fltarr(cb,nb,nc)
e0_mat=fltarr(cb,nb,nc)
e1_mat=fltarr(cb,nb,nc)
e2_mat=fltarr(cb,nb,nc)
trace_mat=fltarr(cb,nb,nc)
m_adc=fltarr(cb,nb,nc)
m_adc_mat=fltarr(cb,nb,nc)

for i = 0, 7*num_files-1 do begin
                b_matrix(0,i) = b(0,i)^2
                b_matrix(1,i) = b(1,i)^2
                b_matrix(2,i) = b(2,i)^2
                b_matrix(3,i) = 2*b(0,i)*b(1,i)
                b_matrix(4,i) = 2*b(0,i)*b(2,i)
                b_matrix(5,i) = 2*b(1,i)*b(2,i)
endfor

orib=b_matrix

zero_these = where(tdim(*,*,*,num_files-1) lt av*3.0)

for index = 0, 7*num_files-1 do begin
temp = tdim(*,*,*,index)
temp(zero_these) = 0
tdim(*,*,*,index) = temp
endfor

for k=0, nc-1 do begin

for j=0, nb-1 do begin
                for i = 0, cb-1 do begin

test_array = where(tdim(i,j,k,*) eq 0)

                                if test_array(0) ne -1 then begin
                                d_matrix(*,i,j,k) = [0,0,0,0,0,0]
                                a0_matrix(i,j,k) = 0
                                v_d(*,i,j,k)=[0,0,0]

                                goto, skip3d
                                endif

;******find omitted points and change array size********
data_array = tdim(i,j,k,*)
;if (omit_point(0) ne -1) then begin
;               data_array(omit_point) = 0.0000000000000000000001
;               check = 1
;endif

                d=regress(-b_matrix,alog(reform(data_array)),(reform(data_array))/(av),yfit,a0)
;if check eq 1 then begin
                ;print,'data array = ',data_array
                ;print,'b matrix = ',b_matrix
                ;print,'weighting = ',data_array/av
                ;print,'d = ',d
;endif

                m_adc = (d(0)+d(1)+d(2))/3
                d_matrix(*,i,j,k) = d      ;dif tensor before diagonalize
                a0_matrix(i,j,k) = a0      ;a0 image

                ;******* diagonalizatin ********
                d_tensor=[[d(0),d(3),d(4)],[d(3),d(1),d(5)],[d(4),d(5),d(2)]]
                print,d_tensor
                nr_tred2,d_tensor,d_out,e
                nr_tqli,d_out,e,d_tensor
                print,'d_out = ',d_out,' d_tensor = ',d_tensor
                read,'ok?',damy
                ;******* sorting *******
                max_d_ele = max(d_out,maxi)
                min_d_ele = min(d_out)
                med_d_ele = median(d_out)
                trace_val = (d_out(0)+d_out(1)+d_out(2))

                v_d(*,i,j,k) = d_tensor(*,maxi)
                if (v_d(2,i,j,k) lt 0) then v_d(*,i,j,k) = -v_d(*,i,j,k)

                e0_mat(i,j,k)=max_d_ele
```

```
                    if (e0_mat(i,j,k) lt 0) then e0_mat(i,j,k)=0

                    e1_mat(i,j,k)=med_d_ele
                    if (e1_mat(i,j,k) lt 0) then e1_mat(i,j,k)=0

                    e2_mat(i,j,k)=min_d_ele
                    if (e2_mat(i,j,k) lt 0) then e2_mat(i,j,k)=0

                    trace_mat(i,j,k)=trace_val
                    if (trace_mat(i,j,k) lt 0) then trace_mat(i,j,k)=0

                    m_adc_mat(i,j,k)=m_adc
                    if (m_adc_mat(i,j,k) lt 0) then m_adc_mat(i,j,k)=0

i1=d_out(0)+d_out(1)+d_out(2)
i2=d_out(0)*d_out(1)+d_out(0)*d_out(2)+d_out(1)*d_out(2)
i3=d_out(0)*d_out(1)*d_out(2)
i4=i1^2.0-2.0*i2
dav=i1/3.0
dsurf=sqrt(i2/3.0)
dvol=i3^(1.0/3.0)
dmag=sqrt(i4/3.0)

ani_idx(i,j,k)=sqrt(1.0-(dsurf^2.0)/(dmag^2.0))
if (ani_idx(i,j,k) gt 1.0) then ani_idx(i,j,k) = 1.0
if (ani_idx(i,j,k) lt 0) then ani_idx(i,j,k) = 0
                                        skip3d:

                    endfor
endfor
b_matrix = orib
endfor


window,0,ysize=512,xsize=1024
tvscl,congrid(a0_matrix(*,*,nc/2),256,256),0
tvscl,congrid(reform(d_matrix(0,*,*,nc/2)),256,256),1
tvscl,congrid(reform(d_matrix(1,*,*,nc/2)),256,256),2
tvscl,congrid(reform(d_matrix(2,*,*,nc/2)),256,256),3
tvscl,congrid(reform(d_matrix(3,*,*,nc/2)),256,256),4
tvscl,congrid(reform(d_matrix(4,*,*,nc/2)),256,256),5
tvscl,congrid(reform(d_matrix(5,*,*,nc/2)),256,256),6


menu3:
read,'save (1) or vector presentation(2) or goto menu(3)',ans
if (ans eq 3) then goto, menu
if (ans eq 2) then goto, threed_vector

name=''
print,'output file name for vectors : '

v_d_temp=reform(v_d, 3*cb,nb,nc)
m_store_sub_f,v_d_temp,3*cb,nb,nc,name_w
v_d_temp=1

pieces=str_sep(name_w,'.')

print,'output aniso_index file; '
                    for i=0, nc-1 do begin
                              norm_temp=intarr(cb,nb,nc)
                              tep_mat=ani_idx(*,*,i)

                              noral=where((tep_mat ge 0.0) and (tep_mat le 1.0))
                              norm_temp(noral)=1
```

```
                                        satu=where(norm_temp ne 1)
                                        tep_mat(satu)=1
                                        ani_idx(*,*,i)=tep_mat
                    endfor
name_ani=strcompress(pieces(0)+'.ani')
m_store_sub,ani_idx,cb,nb,nc,name_ani

print,'output e0 image file: '
name_e0=strcompress(pieces(0)+'.e0')
m_store_sub,e0_mat,cb,nb,nc,name_e0

print,'output e1 image file: '
name_e1=strcompress(pieces(0)+'.e1')
m_store_sub,e1_mat,cb,nb,nc,name_e1

print,'output e2 image file: '
name_e2=strcompress(pieces(0)+'.e2')
m_store_sub,e2_mat,cb,nb,nc,name_e2

print,'output Trace image file: '
name_tra=strcompress(pieces(0)+'.tra')
m_store_sub,trace_mat,cb,nb,nc,name_tra

print,'output mean_ADC image file: '
name_adc=strcompress(pieces(0)+'.adc')
m_store_sub,m_adc_mat,cb,nb,nc,name_adc

goto,menu3

print,'store dxx values'
store, reform(d_matrix(0,*,*,*))
print,'store dyy values'
store, reform(d_matrix(1,*,*,*))
print,'store dzz values'
store, reform(d_matrix(2,*,*,*))
print,'store dxy values'
store, reform(d_matrix(3,*,*,*))
print,'store dxz values'
store, reform(d_matrix(4,*,*,*))
print,'store dyz values'
store, reform(d_matrix(5,*,*,*))
print,'store a0 values'
store, a0_matrix
goto,menu

threed_vector:

spacewalk_sub1,v_d,a0_matrix,ani_idx
goto, menu3

 error:
print,'mismach'

end_of_program:

return
end


pro m_store_sub_f, matrix,a,b,c,name

;this procedure store the unformatted matrix    to disc
;modified by Itoh    Sep28/1999
;for Mac by Itoh    Oct5/2003
```

```
name=dialog_pickfile(/write,path='/Users/idl/data/tensor/')
openw, unit, name, /get_lun
                for index=0,c-1 do begin
                                d=assoc(unit,fltarr(a,b,/nozero))
                                d(index)= matrix(*,*,index)
                endfor

close, unit
free_lun, unit

end

pro m_store_sub, matrix,a,b,c,name

;this procedure store the unformatted matrix    to disc
;modified by Itoh    Sep28/1999

openw, unit, name, /get_lun
                for index=0,c-1 do begin
                                d=assoc(unit,fltarr(a,b,/nozero))
                                d(index)= matrix(*,*,index)
                endfor

close, unit
free_lun, unit

end

pro store_sub_byte_m, matrix,a,b,c,name
;this procedure store the unformatted matrix    to disc

get_lun,unit
;openw, unit,'/mri/kirin/susumu/data/'+ name
openw, unit, name

for index=0,c-1 do begin
d=assoc(unit,bytarr(a,b,/nozero))
d(index)= matrix(*,*,index)
end

close, unit
free_lun, unit
end

pro call_byte_m, matrix

;this procedure call the unformatted matrix from disc to matrix
read,'number of slice?: ',a
read,'Direction x NSA?: ',b
;read,'how many matrix?: ',c

c=1

matrix=bytarr(a,b,c)

print,'name for rejection/acceptance matrix .ra '
name=dialog_pickfile(/read,path='/usr/people/itoh/data')
openr, unit, name,/get_lun

for index=0,c-1 do begin
                d=assoc(unit,bytarr(a,b,/nozero))
                matrix(*,*,index)=d(index)
end
matrix=reform(temporary(matrix))
```

```
close, unit
free_lun, unit
return
end

pro call_g, x_mat, y_mat, slice, matrix

matrix=intarr(x_mat, y_mat, slice*7)

print,'name of IMAGE matrix (intarr)'
name=dialog_pickfile(/read,path='/Users/idl/data/tensor/')
openr, unit, name,/get_lun

for index=0,slice*7-1 do begin
                d=assoc(unit,intarr(x_mat,y_mat,/nozero))
                matrix(*,*,index)=d(index)
endfor

close, unit
free_lun, unit

matrix=swap_endian(matrix)

return
end
```

APPENDIX 4

```
pro co_m

; for display and generating TIFF file of color-map.
; The original pro. was written by S Mori.

; measure ave ADC and FA values within the ROI on color-map.
; modified by R Ito., Aug18/2002.
; last modification of auto-reading ani.file by R.Ito., Aug21/2002
; modification for Win, by R Ito, Sep 21/2002

n1=128
n2=128
n3=20

slope = 1.5
intercep = 0.1

read,'x matrix?: ',n1
read,'y matrix?: ',n2
read,'number of slice?: ',n3

ani = fltarr(n1,n2,n3)
v_d = fltarr(3*n1,n2,n3)

;******** read a0 ***********
print,'read .vec file'
path_vec=dialog_pickfile(/read,path='/Users/idl/data/tensor/')
get_lun,unit
openr, unit,path_vec

for index=0,n3-1 do begin
d=assoc(unit,fltarr(3*n1,n2,/nozero))
v_d(*,*,index)=d(index)
```

```
end
close, unit
free_lun, unit

v_d = reform(temporary(v_d),3,n1,n2,n3)

;******* flip x-directin *****

;colonal philips******
;v_d(0,*,*,*)=temporary(-v_d(0,*,*,*))
;v_d(1,*,*,*)=temporary(-v_d(1,*,*,*))
;v_d(2,*,*,*)=temporary(-v_d(2,*,*,*))
;tempvd=fltarr(3,n1,n2,n3)
;tempvd(0,*,*,*) = v_d(1,*,*,*)
;tempvd(1,*,*,*) = v_d(0,*,*,*)
;tempvd(2,*,*,*) = v_d(2,*,*,*)
;v_d = tempvd
;tempvd = 1


;******** read ani ***********
print,'read .ani file'
pieces=str_sep(path_vec,'.')
path_ani=strcompress(pieces(0)+'.ani')
;path_ani=dialog_pickfile(/read,path='/usr/people/itoh/data/tensor/')
get_lun,unit
openr, unit, path_ani

for index=0,n3-1 do begin
d=assoc(unit,fltarr(n1,n2,/nozero))
ani(*,*,index)=d(index)
end
close, unit
free_lun, unit

display:

read,'slice location? ',slice
                    if (slice le 0) or (slice gt n3) then begin
                                        print,'out of range!'
                                        goto,display
                    endif

tempv = reform(v_d(*,*,*,slice-1))
tempani = reform(ani(*,*,slice-1))
measure_ani = ani(*,*,slice-1)

tempani = (tempani*slope)-intercep
here = where(tempani gt 1)
if here(0) ne -1 then tempani(here) = 1
here = where(tempani lt 0)
if here(0) ne -1 then tempani(here) = 0

tempv = tempv*255

tempv(0,*,*) = tempv(0,*,*)    * tempani
tempv(1,*,*) = tempv(1,*,*)    * tempani
tempv(2,*,*) = tempv(2,*,*)    * tempani

window,0,xsize=320,ysize=320
;tv,congrid(abs(tempv),3,320,320),0,0,true = 1
temp=congrid(abs(tempv),3,320,320)

redis:
tv,temp,0,0,true=1
```

```
select:
read,'finish;0, other location;1, measure;2, save PICT;3 ',finish
if finish eq 0 then goto, endofprogram
if finish eq 1 then goto, display
if finish eq 3 then goto, tif_file
if finish ne 2 then goto, select

put_roi:
;pro put_roi,x_mat,y_mat,disp_m,region,dammy_img2
                roi_sett:

                dammy_img=intarr(320,320)
                dammy_imgr=intarr(n1,n2)

                roi_tt:
                t_region=defroi(320,320)

                dammy_img(t_region)=1

                dammy_imgr=congrid(dammy_img,n1,n2)
                region=where(dammy_imgr eq 1)

                fa_out=measure_ani(region)
                ;adc_out=adc_m(roi)

                k=n_elements(fa_out)

                m_fa_out=total(fa_out)/k
                ;m_adc_out=total(adc_out)/k

                sd_fa=stddev(fa_out)
                ;sd_adc=stddev(adc_out)

                print,'Num of data,',k
                ;print,'ADC; mean value,',m_adc_out,' SD,', sd_adc
                print,'FA; mean value,',m_fa_out,' SD,', sd_fa

                select2:
                sele2=''
                read,sele2,prompt='save PICT? [y/n] '
                if sele2 eq 'n' then goto, redis
                if sele2 ne 'y' then goto, select2

tif_file:
print,'write .pict file'
wset,0
path_tif=dialog_pickfile(/write,path='/Users/idl/data/tensor/PICT/')
path_tif=strcompress(path_tif + '.pict')
write_PICT,path_tif
goto,redis

endofprogram:

return
end
```