Graphical Interface for Quantitative T1 and T2 Mapping of MRI Data

by

Bruce C. Po

Bachelor of Science in Electrical Engineering and Computer Science

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 23, 2001
[June 2001]

Author _____
Department of Electrical Engineering and Computer Science
May 23, 2001

Certified by _____
Deborah Burstein
Thesis Supervisor

Certified by _____
Martha L. Gray
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Graphical Interface for Quantitative T1 and T2 Mapping of MRI Data

by

Bruce C. Po

Submitted to the
Department of Electrical Engineering and Computer Science

May 23, 2001

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Clinical research using MRI has found the need to quantitatively map T1 and T2 parameters in tissue as part of early diagnosis of disease. Using models of T1 and T2 relaxation and an appropriate function minimization algorithm, T1 and T2 can be calculated on a pixel-by-pixel basis from a sequence of T1- or T2-weighted images. The purpose of the software presented in this project is to enable clinicians and researchers to generate T1 and T2 maps by providing an intuitive interface for applying T1 and T2 mapping algorithms to their MRI data. The software implements T1 and T2 mapping algorithms, procedures for loading and viewing the MRI data and calculated maps, and a graphical user interface for controlling operations. The overall flow of the software is: (1) load a set of MRI data from a study, (2) perform calculations on the data set, (3) store the calculated data sets and parameters to disk, and (4) visually display the results.

Thesis Supervisors:

Deborah Burstein, Ph.D.
Associate Professor of Radiology
Beth Israel Hospital
Harvard Medical School

Martha L. Gray, Ph.D.
Professor of Electrical and Medical Engineering
Department of Electrical Engineering and Computer Science
MIT and Harvard-MIT Division of Health Sciences and Technology

Table of Contents

1 Introduction

The considerable contrast and spatial resolution of magnetic resonance imaging has allowed clinicians and researchers to noninvasively diagnose and study a variety of diseases in increasing detail. Clinical research using MRI has found the need to quantitatively map T1 and T2 parameters in tissue as part of early diagnosis of disease. Commercial medical imaging systems and software are capable of generating T1- and T2-weighted images, but they do not have provisions for calculating the T1 or T2 values themselves. The project presented here is a software application that enables clinicians and researchers to map T1 and T2 parameters in their MRI data.

The objectives of this project were:

- Implementation of T1/T2 mapping algorithms.
- Design of a user interface that economizes the process of loading MRI data from the imaging system, analyzing the data, and displaying the results.
- Improvement of algorithms' performance.
- Verification of the implemented algorithms.

The project objectives are discussed in detail in Section 3.

A key feature of the T1/T2 mapping program is a graphical user interface. The user interface offers researchers and clinicians easy access to the algorithms to generate T1 and T2 maps of their MRI data. The overall flow of the software is: (1) load a set of MRI data from a study, (2) perform calculations on the data set, (3) store the calculated data sets and parameters to disk, and (4) visually display the results. Loading MRI data requires the user to input a number of parameters associated with the data. The user then selects which portions of the data to generate T1/T2 maps. Selection is done by graphically drawing out the regions of interest. Following T1/T2 mapping the software

displays the calculated map and offers a variety of ways to analyze and display the results.

2 Background

2.1 Nuclear Magnetic Resonance

When nuclei have an odd number of protons or an odd number of neutrons or both they will have a net magnetic dipole moment and a net angular momentum. In the presence of an external magnetic field $\vec{B}_0$, the magnetic moment precesses around the direction of $\vec{B}_0$, whose direction is defined to be along the z-axis. According to quantum mechanics, both the angular momentum and the z-component of the magnet moment are quantized. Every nucleus has an associated spin number in units of half-integers. A nucleus with spin ½ will have a magnetic dipole moment with one of two possible states, m=½ or m=-½. The two states will have the same energy level in the absence of an external magnetic field, but in the presence of an external magnetic field each state will have a different energy level such that at equilibrium more nuclei will be in the lower energy state.

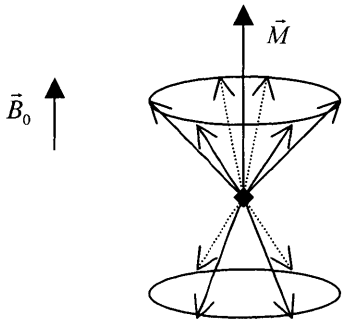


Figure 1: Precession of magnetic dipole moment.

The magnetic moments precess at a frequency described by the Larmor equation:

$$\omega = \gamma \cdot B$$

where ω is the precessional frequency in hertz, B is the field strength in Tesla, and γ is the gyromagnetic ratio specific and constant for each specie of nucleus. ω is also known as the Larmor frequency. It is impossible to observe the magnetic moments of individual nuclei, so the net magnetic moment $\vec{M}$ is considered. At equilibrium, $\vec{M}$ is aligned with the z-axis (M_x and M_y are zero) since the individual magnetic dipole moments are precessing at random phases with respect to each other.

To study the nuclear properties of the subject, $\vec{M}$ is tipped into the x-y plane – or equivalently the distribution of m=½ and m=-½ is disrupted – by introducing another magnetic field $\vec{B}_1$ perpendicular to $\vec{B}_0$. Because the magnetic dipole moments are precessing at the Larmor frequency, $\vec{B}_1$ will affect $\vec{M}$ only if it is also oscillating at the same resonant frequency. The Larmor frequency for hydrogen in typical magnetic field strengths is in the radio frequency range, so the momentary application of $\vec{B}_1$ is accomplished through an RF pulse. As $\vec{M}$ tips into the x-y plane it also precesses at the Larmor frequency. This is due to the RF pulse causing the individual nuclei's magnetic moments to precess together rather than randomly. When the RF wave is removed, entropy will cause $\vec{M}$ to gradually realign itself with $\vec{B}_0$ in a process called relaxation.

2.2 Relaxation

NMR studies are interested in the relaxation properties of $\vec{M}$. One component of relaxation is the re-growth, or recovery, of M_z, known as T1 relaxation. Another component of relaxation is the decay of M_{xy}, in the x-y plane, known as T2 relaxation. The quantities T1 and T2 represent rate constants in the respective recovery equations for T1 and T2 relaxation. In general, M_z regenerates to M_0, the value of M_z at equilibrium,

much slower than M_{xy} decays to zero, that is, T1 is much larger than T2. T1 and T2 are properties of the substance being imaged. Water, for example, has a long T1 as compared to tissue.

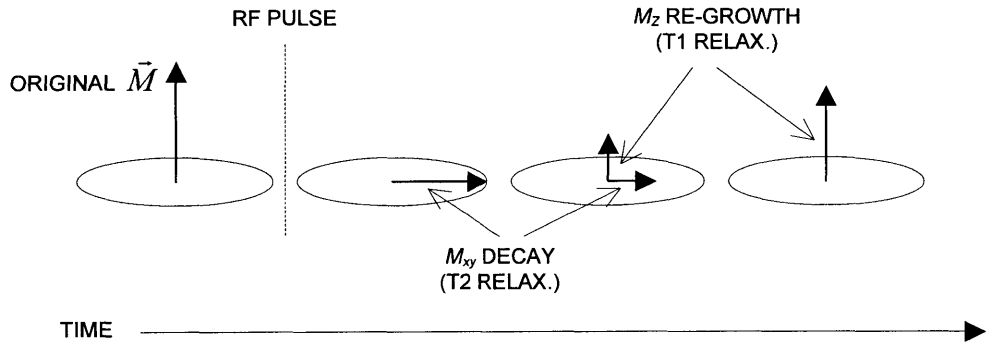


Figure 2: T1 and T2 relaxation.

The transverse component, M_{xy}, of the magnetic moment vector is spinning in the x-y plane, so it generates an electromagnetic field that can be detected up by an antenna. It is not possible, however, to observe the longitudinal component, M_z. At equilibrium, M_{xy} is zero. Suppose a 90-degree RF pulse is introduced, tipping $\vec{M}$ into the x-y plane. Immediately following the pulse the NMR signal jumps from zero to some value and oscillates at the same frequency as M_{xy}, namely the Larmor frequency. Then as M_{xy} decays the NMR signal's amplitude decreases as well. This signal that immediately follows the RF pulse is called the free induction decay (FID).

Although the signal from the FID is representative of relaxation, it alone is not sufficient for measuring T1 and T2. One reason is that T1 relaxation is a function of the M_z component, which is not directly observable. The second reason is that magnetic field inhomogeneities lead to dephasing of the nuclei, so the FID decays more rapidly than true M_{xy} relaxation. In order to overcome these difficulties, MRI scanners use multiple RF pulses at various angles.

2.3 Pulse Sequences

2.3.1 Spin Echo

Because the FID represents the decay of M_{xy} due to the inherent dephasing of the magnetic moments, it would appear that T2 can be calculated by observing the FID following a 90-degree RF pulse. However, slight field inhomogeneities in the magnetic field also cause the FID signal to decay faster than the true T2.

To overcome dephasing of the magnetic dipole moments, a 180-degree RF pulse is introduced some time after the initial 90-degree RF pulse to rephase the MR signal. The 180-degree pulse reverses the polarity of the M_{xy} component so that the field inhomogeneity that caused the dephasing now has an opposite effect with respect to M_{xy}. If the 180-degree pulse is delivered t milliseconds after the initial 90-degree pulse, the magnetic dipole moments will completely rephase t milliseconds after the 180-degree pulse and an NMR signal will be observed. This signal is called the spin echo and reaches its peak at time $2t$. Note that NMR signal loss due to other factors such as spin-spin interaction is irrecoverable, so repeated spin echoes will decrease in amplitude. This doesn't matter since, however, since the loss is in fact T2 relaxation. Consequently, the true T2 curve can be measured by repeatedly applying 180-degree pulses and sampling the spin echoes.

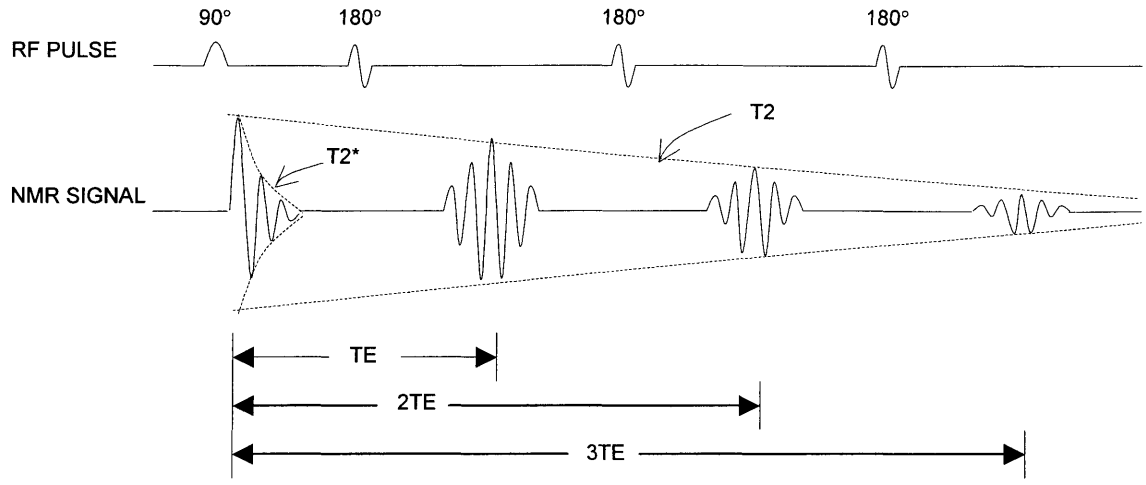
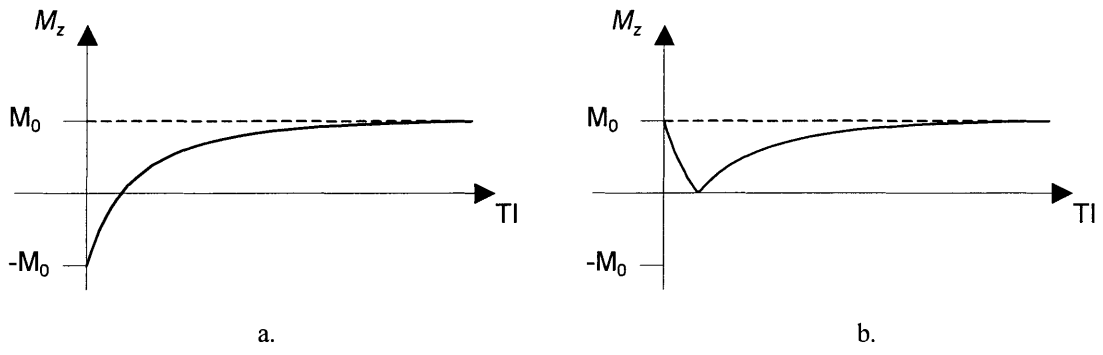


Figure 3: Spin echo for T2 measurement.

Many spin-echo acquisitions can be performed since each acquisition does not add to the total scanning time. The repetition time (TR) between successive scans must be long enough to allow T2 to recover fully.

2.3.2 *Inversion Recovery*

Suppose the imaging specimen receives a 180-degree RF pulse. The M_z vector is flipped to $-M_0$. But because this is an unstable state so M_z recovers back to M_0 at an inverse exponential rate. From $-M_0$, M_z first shrinks and passes through zero then grows until it reaches $+M_0$ again, as shown in Figure 4a.



a. b.

Figure 4: Inversion recovery plot.

But since the imaging system takes the magnitude of the signal, the observed recovery of M_z is all positive, following Figure 4b.

At equilibrium the nuclei emit no NMR signal because M_{xy} is zero and M_z is not directly observable. When $\vec{M}$ is inverted (by the 180-degree pulse) there will still be no NMR signal because M_{xy} remains 0. However, if a 90-degree pulse is sent some time after the initial 180-degree pulse, M_z would be rotated into the x-y plane where an NMR signal would be observable. The time between the initial 180-degree pulse and the 90-degree pulse is called the inversion time (TI). If this process were repeated with increasing TI, the signal received can be interpreted as points along the T1 curve. The repetition time TR for beginning another inversion should be long enough to let M_z recover fully.

The observed M_z recovery following a 180-degree inversion can be modeled by

$$M_z = \left| M_0 (1 - 2e^{-t/TI} + e^{-TR/TI}) \right| \qquad [1]$$

where t takes on the values of TI for acquisitions.

2.3.3 Saturation Recovery

Saturation recovery follows the re-growth of M_z following a 90-degree pulse. An initial 90-degree pulse is applied to the nuclei. After a delay TR' to allow the M_z vector to re-grow from zero, M_z's magnitude is acquired. When this is repeated with different values of TR', the data values can be interpreted to be points along a T1 recovery curve. The recovery of M_z can be modeled by

$$M_z = M_0 (1 - e^{-t/TI}) \qquad [2]$$

where t takes the different values of TR'.

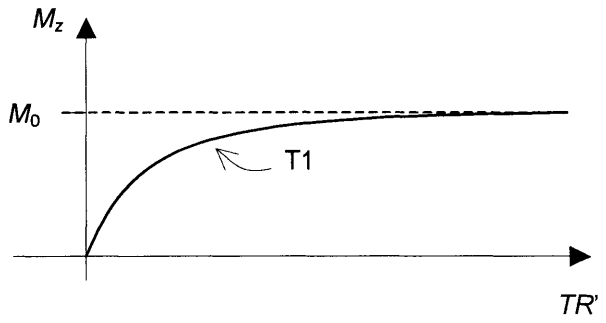


Figure 5: Saturation recovery plot.

2.4 Estimating T1 and T2 Relaxation

For imaging T1 and T2, an MRI imaging sequence is repeated with a series of TI or TR' or TE values with a resultant two dimensional data set where the set of images are all from the same slice in the imaging specimen but at different acquisition times. For the remainder of this document, the term "acquisition times" will refer to TI for inversion recovery, TR' for saturation recovery, and TE for T2. Each pixel value in an image corresponds to the acquired NMR signal strength of a small spatial region in the sample. Suppose a pixel at location (x_0, y_0) in the first image is selected. Then looking at the same location (x_0, y_0) in subsequent images is equivalent to observing the same region in the imaging specimen over different acquisition times.

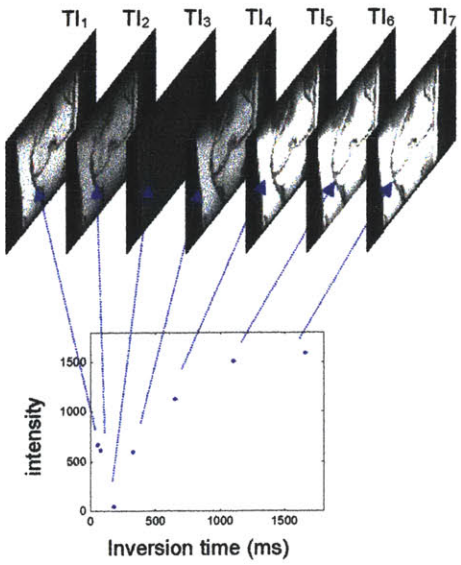


Figure 6: MRI source data

Having the acquisition data for a given pixel, it is feasible to quantitatively estimate the point's time constant T1 or T2 using an appropriate relaxation equation and a function minimization algorithm, such as the Simplex algorithm. In this manner a complete map of T1 or T2 values can be generated (see Figure 7).

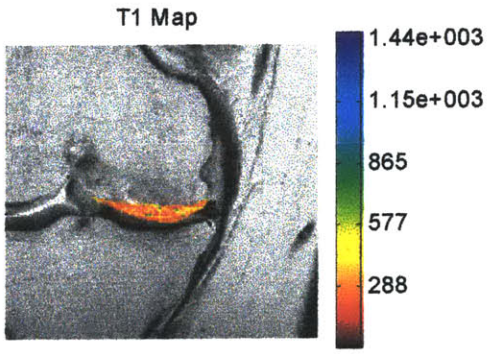


Figure 7: MRI image overlaid with calculated T1 map.

2.5 Applications

With a false-color overlay of the calculated T1 or T2 map, a clinician can quickly identify regions of high and low T1 or T2. Diseased tissue will often have different T1

and T2 characteristics as compared to healthy tissue. Finding the correlation between T1/T2 and various tissue diseases is an enormous area of research.

One such ongoing research area is in cartilage degradation. A major constituent of human cartilage is the macromolecule Glycosaminoglycan (GAG), which is important for the tissue's mechanical function. Deterioration of cartilage can take the form of GAG depletion. Measurement of [GAG] in a non-invasive manner using MRI is thus desirable. Through spectroscopy experiments it has been shown that there is an approximately linear relationship between [GAG] and T1 in the presence of the contrast agent $Gd(DPTA)^{2-}$. Pixel-by-pixel mapping of T1 allows locating precisely where GAG loss is occurring (1).

3 Project Objectives

3.1 Implementation of Mapping Algorithms

The mapping algorithms process the input MRI image sequence and generate T1 or T2 maps. The algorithms process the data pixel by pixel, calculating the T1 or T2 for each pixel. For each type of pulse sequence there is a corresponding relaxation equation that the mapping algorithm will use. Section 4.1 describes the T1 and T2 mapping algorithm in more detail. Once the T1 or T2 parameters have been mapped, the results are saved to disk.

3.2 User Interface Design

The user interface is foremost responsible for enabling the user to apply the mapping algorithms onto his/her MRI data. The goal of the interface design is to streamline and automate, wherever possible, the process of loading MRI data and its parameters, analyzing the data, and displaying the results. A completed T1 or T2 map is saved into a results file, allowing the user to retrieve and edit the map and its parameters at a future time.

There are a number of parameters associated with MRI data from the scanning system. The parameters include pulse sequence type, matrix size, and acquisition times (TI for inversion recovery, TR for saturation recovery, TE for T2 imaging), as well as option parameters for the mapping algorithm. The user interface groups these parameters into a dialog box so that they can be easily viewed and edited from a single location. The interface also allows the user to retrieve parameters from an existing results file so that the user doesn't have to reenter it all.

The interface allows the user to view the input data and select which portions of the data to analyze. Prior to executing the parameter estimation algorithm, the user can specify which pixels to include in T1/T2 mapping by graphically defining regions of interest. In displaying the data as images, the software has provisions for automatically setting the optimal contrast, dynamically scaling the T1/T2 map color scale, highlighting regions of interest, and adding text labels.

Finally, the user interface makes it possible and intuitive to view previously calculated T1/T2 maps without loading and recalculating the data from scratch. The user will be able to add new regions of interest as well.

3.3 Improvement of the Algorithms' Performance

An important function useful for MRI parameter mapping is selecting which data acquisition times to include. Selecting which data acquisition times to include is equivalent to selecting which data points to fit the relaxation curve to. This is useful in cases when certain acquisition times are to be excluded from the T1/T2 mapping (due to excessive noise or misalignment, for example), which will give better curve fits.

Misalignment between the images can lead to particularly bad curve fits, especially in regions with detailed features. Motion during the scans or misregistration of the MRI data are possible sources for misalignment. Many methods for reducing motion-induced artifacts exist (2). This project uses an autocorrelation-based method for correcting the alignment between images, which only estimates motion but does not address motion blur.

3.4 Verification of the Implemented Algorithms

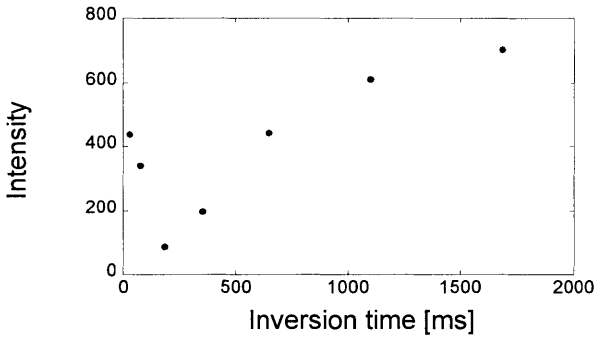
The final objective of this project is to verify the correctness and effectiveness of the T1/T2 mapping algorithms and the image alignment algorithm. A phantom imaging specimen contains regions of uniform, known T1. The phantoms can be imaged with various pulse sequences and scan parameters to compare calculated T1 values. The image alignment algorithm is tested by running it on several image sets and visually inspecting the output.

4 Algorithms

4.1 T1 and T2 Mapping

The multiple data acquisitions for a given pixel allow estimation of the T1 or T2 parameter in the relaxation equation. Suppose the data acquisitions for a given pixel are as given below:

Inversion Time $TI(n)$ **[ms]**	**Signal intensity** $d(n)$
25	439
75	345
180	90
350	200
650	442
1100	613
1680	703



n=1 refers to the first acquisition, n=2 refers to the second acquisition, and so on.

To determine *T1*, an algorithm determines a weighting of the relaxation equation's parameters based on the data $d(n)$. Consider the relaxation equation for inversion recovery:

$$M_z = \left| M_0 (1 - 2e^{-TI/T1} + e^{-TR/T1}) \right| \qquad [3]$$

The three parameters for this equation are M_0, a contrived coefficient, and T1 so that equation [3] becomes

$$M_z = \left| P_1 (1 - 2P_2 e^{-TI/P_3} + e^{-TR/P_3}) \right| \qquad [4]$$

The goal is to determine a set of P_1, P_2, P_3 that minimizes the mean square difference between the data d(n) and the relaxation equation evaluated at acquisition times TI(n), viz.

$$l = \min_{P_1,P_2,P_3} \sum_{n=1}^{7} \left[d(n) - \left| P_1 (1 - 2P_2 e^{-TI(n)/P_3} + e^{TR/P_3}) \right| \right]^2 \qquad [5]$$

The search for a global minimum is a nonlinear operation in three dimensions, which can be accomplished using the widely used Nelder-Mead simplex algorithm. The Nelder-Mead simplex algorithm is based on the concept of the simplex, a geometrical object having one more vertex than the number of dimensions of the space. At each iteration the algorithm computes the objective function (relaxation equation) at every vertex, where each vertex is a possible solution set (P_1, P_2, P_3). The algorithm ranks the vertices and attempts to reposition the worst vertex by reflection, expansion, contraction, or shrinkage. When the simplex converges to a minimum, the algorithm terminates. In order for the global minimum to be achieved, the algorithm must be given a reasonable initial point (3).

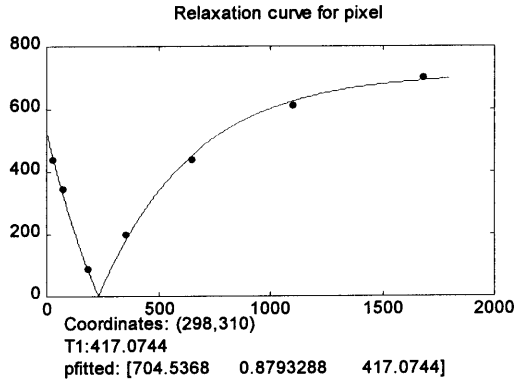


Figure 8: Estimated relaxation equation.

A greater number of parameters in the relaxation equation increases the degrees of freedom for the minimization but the at expense of computations required. The original inversion recovery equation only has the number 2 as the coefficient for the first exponential whereas the minimization algorithm's objective function has introduced a P_2 parameter. The presence of P_2 ensures better estimation of the other parameters to the data points. By not constraining P_2 to equal 1, one may also gain an idea of quality of the input data for estimating T1. If at the end of minimization P_2 is very near to 1, then the data fits very well with the standard model for inversion recovery relaxation (equation [3]). If, on the other hand, P_2 is farther from 1, then one can assume that the data's quality has suffered to some extent. Section 6.1 discusses the results of analyzing phantom specimens with degraded acquisitions.

4.2 Image Alignment

When there is misalignment between the images of the MRI data set, the data array $d(n)$ will contain values originating from slightly different parts of the imaging specimen, hampering accurate estimation of *T1* or *T2*. Image alignment corrects this problem by rotating and translating images so that their features and contours approximately line up with a base image. The alignment algorithm first selects a sub-area from the base image and new image. The translation between the two sub-areas is then estimated and this amount is used to de-translate the entire new image. The two images should now be aligned at the sub-area region, which is defined to be the new origin for rotation estimation. The relative rotation between the base image and the new image is estimated, and the new image is rotated by that amount in the opposite direction. The new image

should now be more closely aligned with the base image. To improve the alignment, the corrections for translation and rotation are iterated twice under certain circumstances.

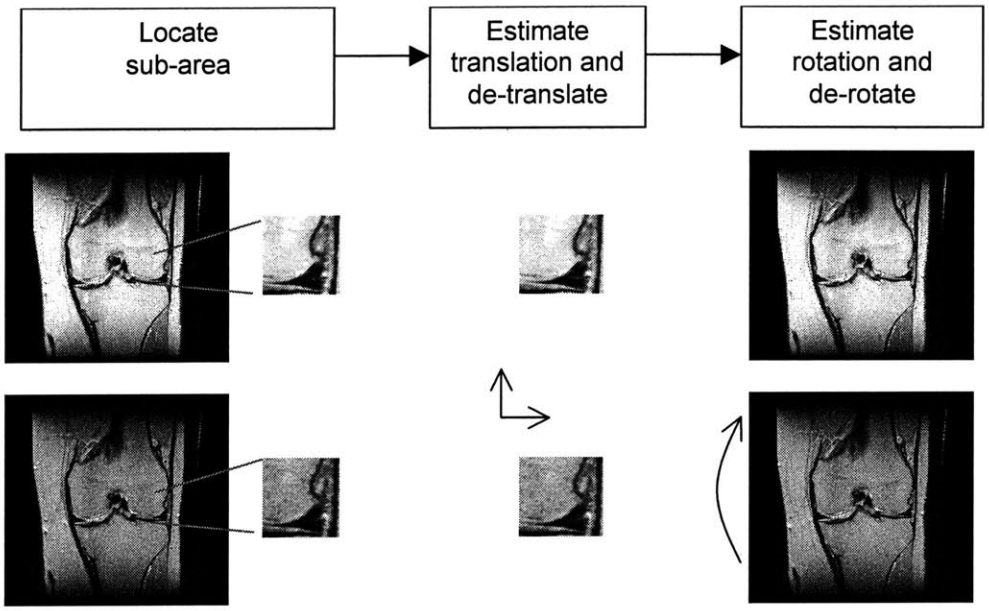


Figure 9: Image alignment.

4.2.1 Block Selection

The sub-area of the images are blocks 64 by 64 pixels in size. If the image size is less than 192 pixels in either dimension, then a smaller block size is chosen. The block region must have "visually interesting" features in order for the ML estimator to perform well. The ML estimator depends on being able to resolve correlation between images. If a block is chosen from a region with relatively uniform values, the ML estimator will likely fail.

To locate the interesting block, an edges-only version of the image is first created and the block that contains the greatest number of edges is selected. The edges-only image can be accomplished efficiently by noting that the differentiation of the image with

respect to x and y somewhat represents the edges in the image. The first derivatives of the image with respect to x and y can be approximated by the difference equations

$$f_x(x,y) = \frac{\partial}{\partial x} f(x,y) \approx \frac{f(x+1) - f(x-1)}{T} \qquad [6]$$

$$f_y(x,y) = \frac{\partial}{\partial y} f(x,y) \approx \frac{f(y+1) - f(y-1)}{T} \qquad [7]$$

The difference equations can be viewed as the convolution of the original image $f(x,y)$ with the $h_x(x,y)$ and $h_y(x,y)$, where $h_x(x,y)$ and $h_y(x,y)$ each consist of two points. A reasonable choice for $h_x(x,y)$ and $h_y(x,y)$ that combines a smoothing operation to the first difference operation is shown in Figure 10.

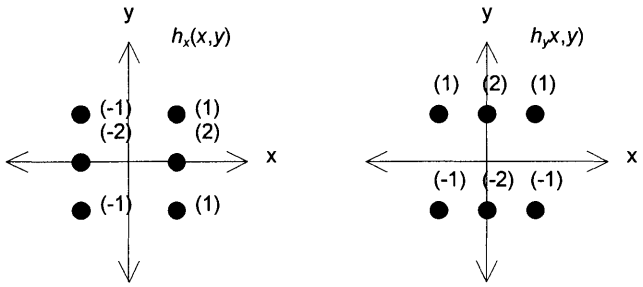


Figure 10: Filters for edge detection.

The edges-only image is created by weighting $f_x(x,y)$ and $f_y(x,y)$ in the following way:

$$\begin{aligned} f_{edges}(x,y) &= \sqrt{f_x^{\,2}(x,y) + f_y^{\,2}(x,y)} \\ &= \sqrt{(f(x,y) * h_x(x,y))^2 + (f(x,y) * h_y(x,y))^2} \end{aligned} \qquad [8]$$

There are numerous variations and approaches to edge detection; this particular selection of $h_x(x,y)$ and $h_y(x,y)$ and the weighting function is known as Sobel's method of edge detection (4).

The edges-only image is then divided into blocks. For each block, the sum of its pixels is a reasonable metric for the amount of edges and features. Thus the block with the greatest sum is chosen as the most "visually interesting" block.

4.2.2 Maximum Likelihood Estimator

Shift compensation uses a maximum likelihood (ML) estimator to estimate the x, y translation between the base image's block and the new image's block. The estimator is structured as follows:

The new image $g(x,y)$ is observed to be a translated version of the base image $f(x,y)$.

$$g(x,y) = \mathrm{T}[f(x-X, y-Y)] + n(x,y) \qquad [9]$$

$f(x,y)$, $g(x,y)$, and $n(x,y)$ are taken to be zero-mean. $n(x,y)$ is noise and is uncorrelated to both $f(x,y)$ and $g(x,y)$. Since $g(x,y)$ is at a later acquisition time, the data values undergo changes from $f(x,y)$ due to NMR relaxation, so T[...] represents this process. Estimating X and Y operates under the assumption that $f(x,y)$ and $g(x,y)$ have sufficient correlation. Specifically, the scalar metric returned by

$$p = \sum_x \sum_y f(x,y) g(x,y) \qquad [10]$$

is a measure of the correlation between $f(x,y)$ and $g(x,y)$ with no relative shift. To evaluate the correlation when there is a shift of (x_0, y_0), the metric is

$$p = \sum_x \sum_y f(x-x_0, y-y_0) g(x,y) \qquad [11]$$

If equation [11] is evaluated over all possible (x_0, y_0) the maximum is the ML estimate of the translation (X,Y) (5). p evaluated over all (x_0, y_0) can be written as

$$\begin{aligned} p(x,y) &= \sum_{k_1}\sum_{k_2} f(k_1 - x, k_2 - y)g(k_1, k_2) \\ &= g(x,y) * f(-x,-y) \end{aligned} \qquad [12]$$

From equation [12], the ML estimator can be realized as processing $g(x,y)$ through the matched filter $f(-x,-y)$. The estimator's speed can be improved without significantly sacrificing accuracy by replacing equation [12]'s convolution operation with a fast Fourier transform, which has order $n\log_2(n)$ execution speed.

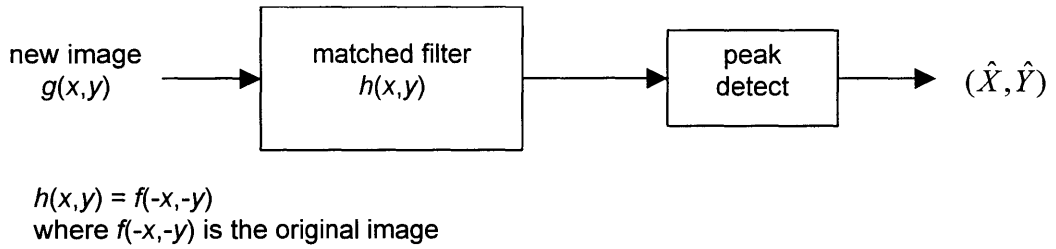


Figure 11: ML estimator.

4.2.3 Rotation Estimation

To estimate rotation between the new image and base image, the same ML estimator procedure as described in the previous section is used. The idea is to run the estimator on data that is indexed in a polar coordinate system. First, pixel data in $f(x,y)$ and $g(x,y)$ are mapped into two new arrays $f'(r,\theta)$ and $g'(r,\theta)$. Mapping requires selecting an origin (x_0, y_0) for the polar coordinate system. The rotation estimator will be effective only if this origin is selected with some reasonable accuracy, that is, the relative rotation between the base image the new image are centered about (x_0, y_0).

$f'(r,\theta)$ and $g'(r,\theta)$ are generated by

$$f'(r,\theta) = f(\cos\theta + x_0, \sin\theta + y_0)$$
$$g'(r,\theta) = g(\cos\theta + x_0, \sin\theta + y_0)$$

Using $f'(r,\theta)$ and $g'(r,\theta)$, the output for the ML estimator is $(\hat{R},\hat{\Theta})$, where $\hat{\Theta}$ is an estimate of the relative rotation and $\hat{R}$ is unused.

5 Implementation

5.1 Software platform

The software was developed in Matlab's programming language (version 5.3 and higher) and consists of source code stored as M-files. Because of the high-level, interpreted language environment of Matlab, M-files can be run directly from Matlab without any compiling, which means the software can run on any computer platform that has Matlab 5 or higher installed. For distribution purposes the M-files are converted into pre-parsed pseudo-code (P-file) format using the `pcode` command in Matlab. Like M-files, P-files are run from within Matlab, but the source code is hidden.

Compiling the source code into MEX files can increase the execution speed of portions of the software. The software can also be compiled to a stand-alone application. Unfortunately, the compiled code is platform dependent with both cases.

5.2 Data Flow

The software maintains a set of global variables for storing the MRI source data and state. The important ones are listed in the table below.

Variable Name	Purpose
`sourceParams`	type of MRI data, location of data files
`dataParams`	MRI scan settings
`viewParams`	image viewing settings
`figs`	source MRI data
`theMapSave`	calculated T1/T2 map
`ROIlist`	ROI definitions

Table 1: Global variables.

The source code is grouped into two primary types of functions: those that contain the logic of the user interface dialog boxes and those that process the MRI data. Each dialog box has an associated function. This function handles initializing and closing the

dialog box and is called whenever the user interacts with any of that dialog box's UI controls. In this manner the user interface functions control the flow of the program. Many of the functions that manipulate the MRI data contain code specific to the source data type. For example, loading MRI data from the imaging system is implemented in three separate functions: one for loading data from Bruker systems, one for loading data from Siemens systems, and one for loading data from GE systems. Parameter mapping is likewise implemented in three different functions: one for T1 inversion recovery, one for T1 saturation recovery, and one for T2 imaging.

The software operations can be grouped into modules according to the tasks performed. The data-loading module consists of a user interface dialog box that sets up `sourceParams` and `dataParams` and loads the MRI data from disk into the array `figs`. The region of interest selection module uses a dialog box to facilitate ROI selection and management. This module also calls the T1/T2 mapping module when the user clicks on the "Make map" button. The mapping module processes the data in `figs` and stores the results into `theMapSave` and saves it to disk. The view results module displays the processed data stored in `theMapSave`.

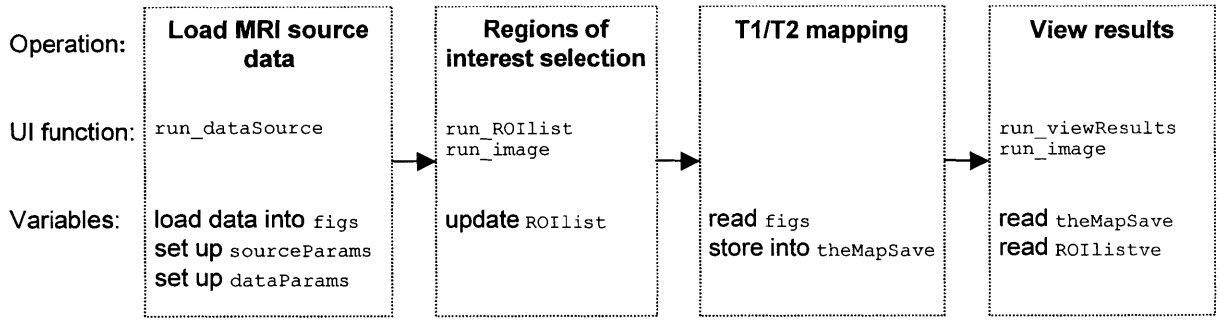


Figure 12: Software operational organization.

When a completed T1/T2 map is stored to disk, the software saves the variables `sourceParams`, `dataParams`, `viewParams`, `theMapSave`, and `ROIlist` into a single results file. The results file is a standard Matlab `.mat` file, so any of the saved variables can be loaded and analyzed in Matlab outside of this software. The MRI source data, which is stored in `figs`, is not saved to disk since `figs` is typically a very large array and can be easily reloaded from the source MRI data file(s). The exception is if the user had opted to run image alignment on the data (see section 5.3.3).

5.3 Components

The user interface is organized as a simple hierarchy of dialog boxes. The Main window is the base of the hierarchy, and its sub-dialog boxes each manage a different stage of the mapping process – loading source data, ROI selection, viewing results. The system of many dialog boxes allows the user to work with any stage at any time as long as the data is available. For example, after calculating a T1 map the user may define additional ROIs to add to the T1 map. The user will be warned if changes made through one dialog box will affect existing data. The interface will also disable buttons and functionality when data is not available. For example, if the MRI source data has not been loaded yet the user will not be able to start the ROI selection dialog box.

The code is structured around the dialog boxes. Each dialog box has an associated function that is called whenever the user interacts with the dialog box's UI components – buttons, sliders, check boxes, etc.

5.3.1 Main Window

When the software is run, the Main window appears. The Main window is a simple dialog box consisting of panels that show what stage of the mapping process the software

and data are. Clicking on a button in the Main window invokes a sub-dialog box associated with that button. If some data is unavailable because it hasn't been loaded or defined yet, then the software will disable certain buttons. Closing the Main window closes all other dialog boxes and quits the program.

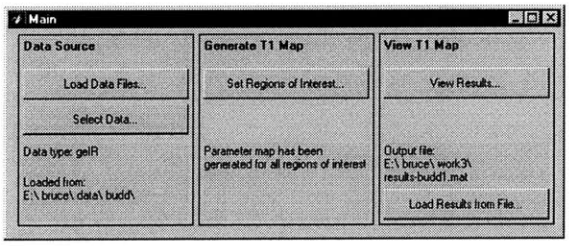


Figure 13: Main window.

5.3.2 Data Source

Initially no MRI data is loaded. Clicking on the button labeled "Load Data Files" in the Main window opens the Data Source dialog box. The Data Source dialog box allows the user to specify the location of the MRI source file(s) and edit the MRI scan parameters. Imaging system selection and pulse sequence is selected from the pull-down list labeled "Magnet type." MRI parameters are set in the main list box. The dialog box automatically sets up default MRI parameters when a new imaging system and pulse sequence is selected. The software will also do some error checking on the user's input and also alert when changes will affect already loaded data. All the settings in this dialog box can also be loaded from a results file by clicking on the button labeled "Get params from file."

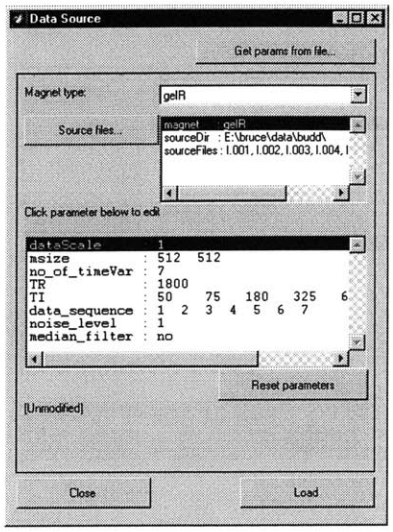


Figure 14: Data Source dialog box.

5.3.3 Select Data

The Select Data dialog box contains functions for managing the source MRI data before running through the mapping algorithm. The dialog box allows the user to view the MRI source data, specify which acquisitions to use and in what order, perform image alignment on the source data, and select the background image for display.

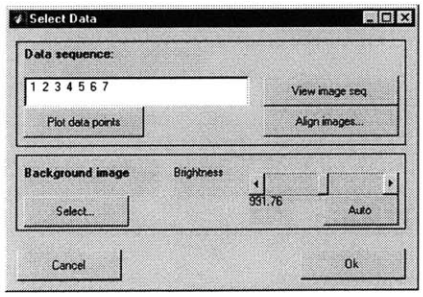


Figure 15: Select Data dialog box.

Clicking on the button "View image seq." views the MRI data as a movie sequence, allowing the user to visually inspect the quality and alignment of the images. A text field lets the user specifying which acquisition times' data to use in the mapping algorithm and in what order. For example, entering "`4:2:10`" or equivalently "`4 6 8 10`" would tell the mapping algorithm to use only the even acquisitions beginning with acquisition

number 4 and ending with number 8. The button labeled "Align images" invokes the image alignment algorithm. The algorithm does not require any interaction from the user except to specify which acquisitions to perform alignment on. When image alignment is complete, the user can click on the "View image seq." button to view the corrected image sequence. This time the image sequence movie will have a button to revert the current image back to the original should the alignment algorithm fail on that image.

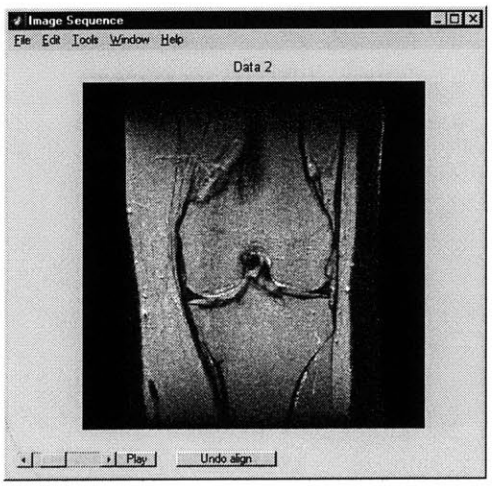


Figure 16: Image Sequence window.

The panel labeled "Background image" lets the user set the brightness of the displayed MRI image and set which image in the MRI source data to use as the background image for display. The background image is only for display purposes and does not affect T1/T2 mapping.

5.3.4 Image Display

The MRI image stored in `img_background` is displayed as a grayscale intensity image. `img_background` is taken directly from the MRI source data, so the range of the data values is some number that depends on the MRI scanner system and many other factors. The function for displaying images must scale the data in `img_background`

before displaying it. It is useful to be able to automatically calculate the scale factor by analyzing the histogram of `img_background`.

A good scale factor results in a displayed image that has a decent balance of darks and lights. In quantitative terms, it is desirable to do a histogram stretch operation so that 95% of `img_background`'s points are distributed throughout the displayed image's histogram, meaning the brightest 5% points will be clipped to white. Let $cum(i)$ be the cumulative sum of the histogram. Then $cum(i = i_0)$ is the number of points in `img_background` that have intensities less than or equal to i_0. Suppose the total number of points is M. (By necessity, $cum(i_{\max})$ is equal to M where i_{max} is the maximum intensity in `img_background`.) We then locate the intensity level i_{opt} where $cum(i_{opt}) \approx 0.95M$. This intensity level i_{opt} is greater than or equal to the value of 95% of the pixels in `img_background`, so it should be mapped to white. If the output intensity of 1.0 is rendered as white, then i_{opt} is the same as the scale factor.

The calculated T1/T2 map likewise has a range of values that depends on many factors, and it would be useful to be able to calculate the scale factor for displaying the T1/T2 map. The scale factor is calculated using the cumulative histogram method as above except 98% is chosen to be the percentage of values displayed.

5.3.5 Regions of Interest Selection

The Regions of Interest dialog box displays a list of currently defined regions of interest and has facilities to add and delete ROIs, as well as load ROIs from another results file. When the Regions of Interest dialog box is invoked, it displays the MRI background image with the defined ROIs, if any, highlighted on the image. To add a new

ROI, the user clicks "Add" and begins drawing the ROI on the MRI image with the mouse. If the checkbox "Square ROI" is checked, then new ROIs are squares rather than free form shapes. Clicking the button "Add entire image" creates a ROI that covers the entire image. Clicking on the "Make map" button executes the mapping algorithm.

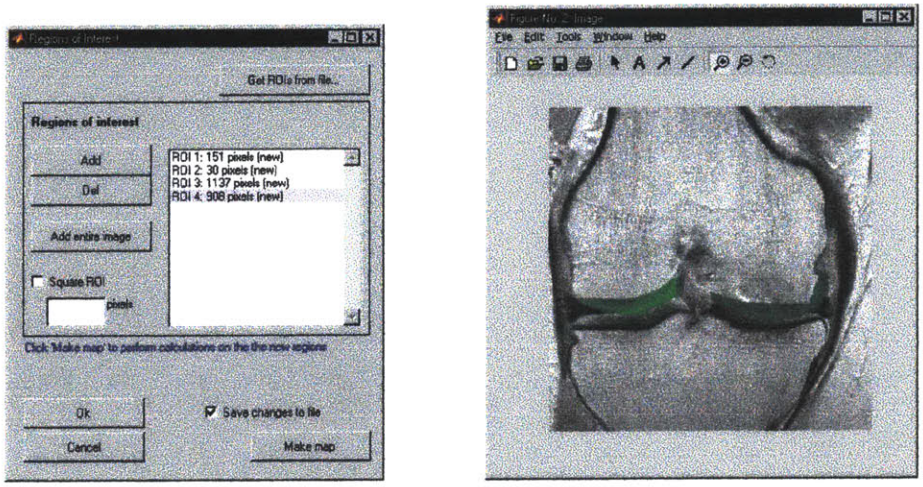


Figure 17: Region of interest selection.

This dialog box can be used to add more ROIs after the current ones have had T1/T2 maps generated. Newly added ROIs will have "(new)" next to their listing. If there are no new ROIs, whether because all current ROIs already have T1/T2 maps generated for or because the ROI list is empty, then the "Make map" button is disabled.

5.3.6 T1/T2 Map Generation

The individual regions of interest are stored in the array `ROIlist`. Regions of interest are allowed to overlap or even be entirely contained with another. However, the T1/T2 map generation module calculates each pixel only once regardless of how many ROIs it belongs to. The software also keeps track of which pixels and which regions of interest already have T1/T2 values generated for it. This is useful for when the user

decides to add additional ROIs after the original set of ROIs has had T1/T2 values calculated.

To generate a T1 or T2 map, the software first creates a list of pixels that need T1/T2 calculation. These pixels are from the ROIs, but pixels that have already had T1/T2 values generated for are excluded. The pixels are then processed one at a time. For each pixel the software estimates the T1 or T2 value using the algorithm described in section 2.4 and saves the pixels into the `theMapSave`.

5.3.7 View Results

The View Results dialog box is started from the Main window, but it is available only when a T1/T2 map has already been generated. When the View Results dialog box is started, it will display the calculated T1/T2 map in another window. A list of ROIs is on the right side, which allows the user to select which ROIs to display. The upper left panel of the dialog box displays the relaxation equation and has two buttons for displaying more detailed information about the T1/T2 estimation. The functions these buttons call will be discussed in more detail in sections 5.3.7.1 and 5.3.7.2. Below this panel is a panel for controlling the display of the T1/T2 map. Two edit boxes control background image maximum value and the T1/T2 map maximum value, useful for adjusting the brightness and color range of the display. The T1/T2 map maximum value is particularly important for maintaining consistent color scale across different MRI scans. There are also options for adding labels to the T1/T2 map figure and saving the figure as a graphics file.

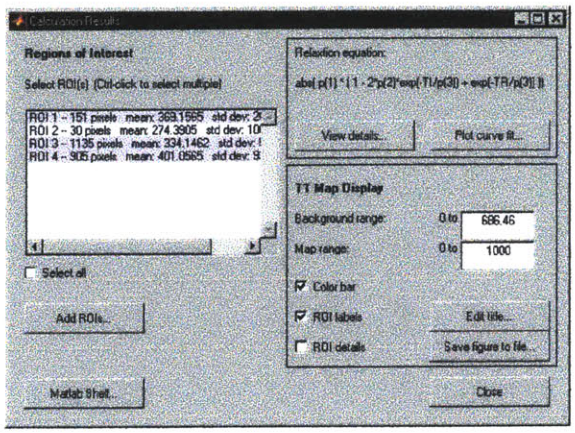


Figure 18: View Results dialog box.

The View Results dialog box also has a button for adding more ROIs. This button invokes the Regions of Interest dialog box described in section 5.3.5.

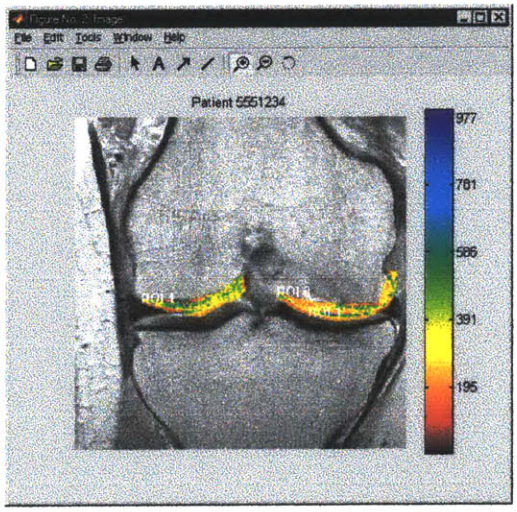


Figure 19: T1 map display window with several regions of interest

5.3.7.1 Analyze Estimated Parameters

Clicking on the button "View details" in the Results dialog box brings up the Analyze Estimated Parameters dialog box. This dialog box displays statistics on the estimated parameters of the relaxation equation. A pull down list lets the user select which parameter – T1/T2, P1, P2, or P3 – to analyze. (The T1 or T2 parameter will be

the same as one of the P parameters.) The statistics displayed are only for the ROIs that have been selected in the View Results dialog box. Clicking on one of the buttons on the left side of the dialog box brings up a figure window to visualize the values in one of three ways: as a linear plot of values, as a histogram, and as a false-color map. The user can also plot a T1/T2 profile across a user-selected line on the image.

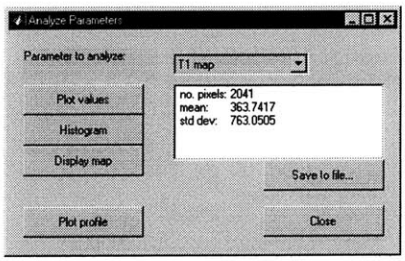


Figure 20: Analyze Parameters dialog box.

5.3.7.2 Analyze Curve Fit

Clicking on the "Plot curve fit" button in the View Results dialog box brings up the Analyze Curve Fit dialog box. This dialog box lets the user plot the relaxation curve for a particular pixel or the average for a region of interest. The plot of the relaxation curve with the data points plotted on the same graph is a visual check, and perhaps the best check, for the correctness of the T1/T2 estimation.

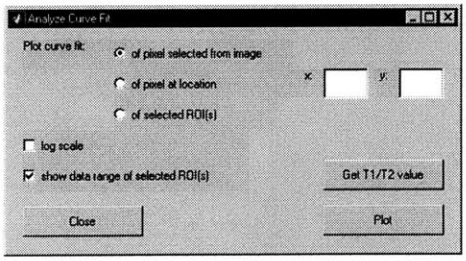


Figure 21: Analyze Curve Fit dialog box.

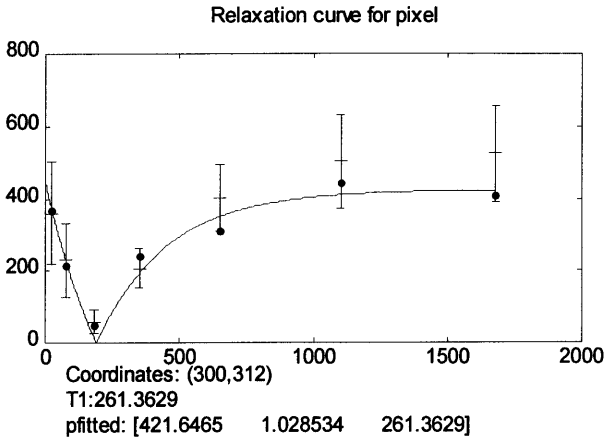


Figure 22: Calculated relaxation curve for a selected pixel. The dots are the source MRI data over different TI times. The vertical lines denote the variance of data in the selected ROI. The means of the data in the selected ROI are shown as tick marks dividing the vertical lines in half.

6 Verification

6.1 T1/T2 mapping

The mapping algorithm as described in section 4.1 minimizes the mean square error between the available data points and the calculated relaxation curve, so the software's T1 and T2 mapping algorithms are already optimal in the mean square error sense. An ad hoc method of verifying the calculated parameters is to visually compare the original data points with the estimated relaxation curve. If the parameter estimation is a good fit, then the curve will match the data points fairly well (see Figure 23). In practice, the plots of pixels of different MRI data and scan types do indeed confirm that the mapping algorithms consistently generate valid relaxation parameters when the data has reasonable quality. In one MRI data set with good SNR, a majority of the estimated relaxation curves fit well with the data points. The parameter P_2 (defined in section 4.1) had a mean of 0.98 and a standard deviation of 0.18. A discussion of more exact measures of the "goodness" of a parameter fit follows.

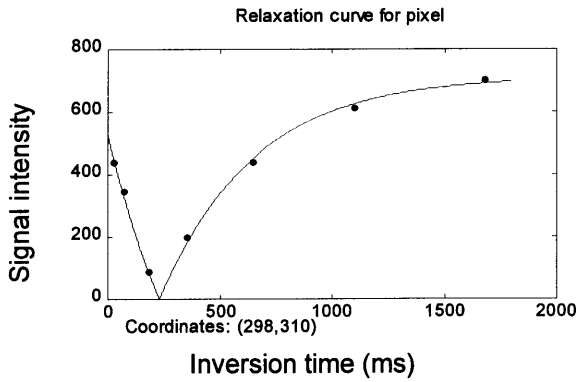


Figure 23: Plot of data points and estimated curve. The estimated curve is the IR relaxation equation using the estimated parameters..

Besides visual inspection, there are several ways of quantifying the quality of the data fit. Recall that the Nelder-Mead simplex algorithm terminates when the global

minimum MSE is reached. One characteristic of a poor fit is the increased number of iterations for the function minimization algorithm before it terminates. Another characteristic of a poor fit is that the minimum MSE the algorithm terminates with is still relatively large. (For comparing with other pixels, the minimum MSE must be normalized.)

Figure 24 shows the resulting MSE and iterations for the ROI pixels of the same data set. As expected for a data set with good quality, the majority of pixel estimations stays within a limited range of MMSE and requires similar numbers of iterations. The graphical plots of two pixels' estimated relaxation curves are shown in Figure 25, one having a good fit and one having a bad fit. The MMSE and iterations numbers for these pixels agree with what one would expect for good and bad fits.

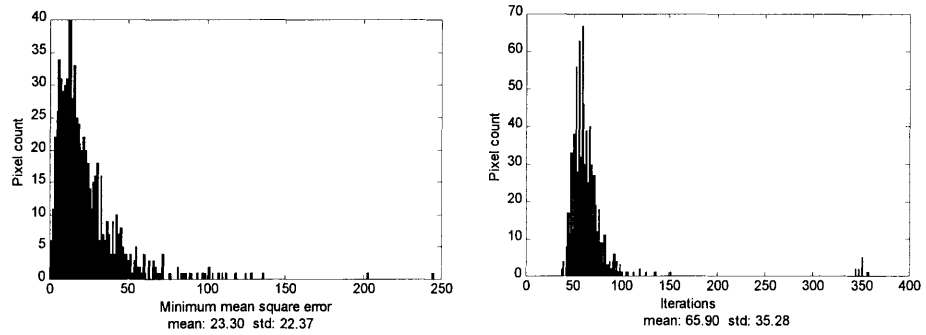


Figure 24: Performance output from Nelder-Mead simplex algorithm. a. Histogram of minimum MSE of data fit. b. Histogram of iterations required in data fit.

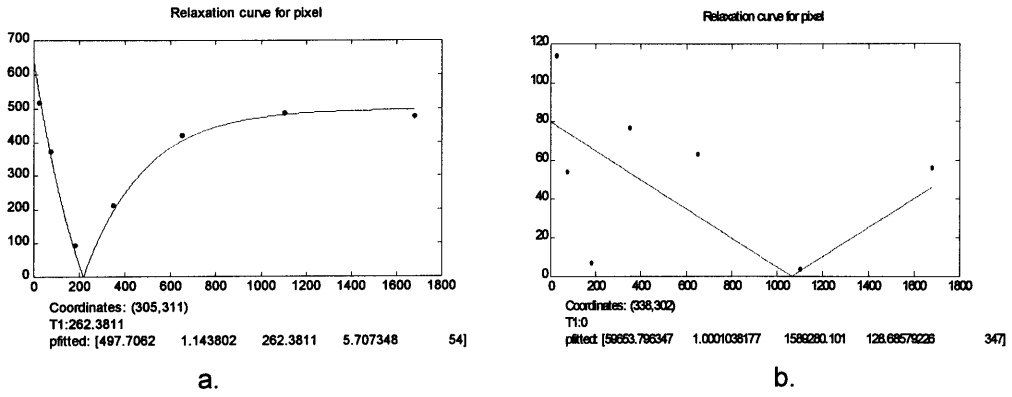


Figure 25: Plots of parameter estimation results. a. A pixel with good fit, requiring 54 iterations and normalized MMSE of 5.70. b. A pixel with poor fit, requiring 347 iterations and normalized MMSE of 128.69.

We are interested in evaluating the performance of the parameter estimator with varying degrees of data quality. Common causes of poor data quality include low signal to noise ratio and imaging in a way that does not favor T1 or T2 contrast. To test the robustness of the estimation algorithms, some of the scanning parameters can be varied. With inversion recovery, for example, one can vary the flip angle and the repetition time. In Figure 26, the 'x's show the calculated T1 when an initial 135-degree pulse is used rather than a 180-degree pulse. The 135-degree pulse has virtually no effect on the calculated T1 values, but the values of the parameter P_2 are affected, as expected, since the function minimizer perturbs P_2 in order to improve the fit of the data to the relaxation equation. Figure 26 also shows the results of using a short time to repetition (TR) of 1 second rather than 15 seconds. The shortened TR affects areas with higher T1 the most.

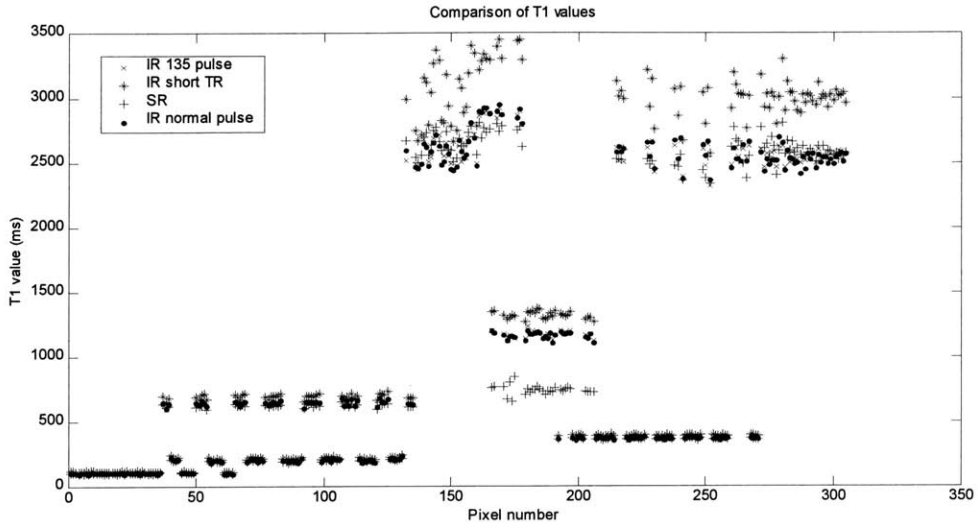


Figure 26: Comparison of calculated T1 with different pulse parameters.

6.2 Image alignment

Unsurprisingly, the success of the image alignment algorithm depends on the degree of misalignment and the signal to noise ratio in the images and varies with different image sets. In one particular inversion recovery data set with seven inversion times, the first image, used as the base image, has good contrast and little noise (Figure 27a). Most of the subsequent images also have good contrast and little noise (Figure 27b), but others have little contrast (Figure 27c). It is typical for at least one inversion time in an IR data set to have a very dark image because the M_z component passes through zero during relaxation.

a. b. c.

Figure 27: a. Base image is from TI_0. b. Image from TI_1. c. Image from TI_3.

To test the alignment algorithm, the image from Figure 27b – image 1 – and the image from Figure 27c – image 2 – were artificially translated and rotated by various amounts. With image 1, the algorithm was able to correct for translations up to 40 pixels when there was no rotation, and it was able to correct for rotations up to 18 degrees when there was no translation. When translation and rotation were simultaneously introduced in image 1, the algorithm was able to correct rotation up to 20 degrees and translation up to 15 pixels. Simultaneous rotation and translation misalignment greater than these values often caused the algorithm to fail, generating output images that were completely crooked. With the noisier image 2, the algorithm was not able to fully correct images with more severe rotation, but the algorithm's ability to correct translation was largely unaffected.

6.3 Further work

The modularity of the code in separating user interface and control functions from data processing algorithms allows new data functions to be added with relative ease. New estimation algorithms and MRI data types can be added to the software with very little modification of the existing code. In this way the software can potentially evolve into a library of data-fitting equations the user can select from.

The software currently leaves the Matlab command prompt active. Users wishing to analyze the calculated data themselves or annotate the figure displays may do so by typing commands on the command line. As the user base of this software grows, there may arise a need to add certain functionality in the software's graphical interface.

The T1/T2 mapping algorithms' performance may be improved in several ways. It may not make sense to estimate T1/T2 to a high precision, so the Nelder-Mead simplex algorithm can be instructed to perform its search more coarsely. This will lead to quicker computation of the parameters, but care must be taken not to sacrifice the accuracy of the estimations. There are a number of possible approaches to determining when pixels qualify as bad fits and addressing the bad data fits, such as masking out the pixel or interpolating values from neighboring pixels. The image alignment algorithm can also be improved to be more efficient and more robust over varying quality of MRI data.

References

1. A. Bashir, M. L. Gray, D. Burstein. Gd-DTPA^{2-} as a Measure of Cartilage Degradation. Magnetic Resonance in Medicine. 36:665-673, 1996.
2. K. P. McGee, A. Manduca, J. P. Felmlee, S. J. Riederer, R. L. Ehman. Image Metric-Based correction (Autocorrection) of Motion Effects: Analysis of Image Metrics. Journal of Magnetic Resonance Imaging. 11:174-181, 2000.
3. J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. Computer Journal, 7:308-313, 1965.
4. J. S. Lim. Two-Dimensional Signal and Image Processing. Upper Saddle River, New Jersey: Prentice Hall PTR, 1990.
5. A. S. Willsky, G. W. Wornell, J. H. Shapiro. Stochastic Processes, Detection, and Estimation: 6.432 Course Notes. Cambridge, Massachusetts: MIT Department of Electrical Engineering and Computer Science, 2000.

Appendix – Source Code

A.1 User Interface Control Files

A.1.1 User Interface Functions

```
%MRIMAPPER

% Bruce Po - igan@mit.edu

more off
global sourceParams dataParams viewParams figs theMapSave pfittedSave
run_main( 'init' );

% INCLUDE_GLOBALS
%  Define global variables are common to all the modules.

% Bruce Po - igan@mit.edu

global figs sourceParams dataParams viewParams paramPrompts img_background ...
    ROIlist theMapSave theMapi pfittedSave fh_image
global sourceParamsValid figsValid ROIlistValid theMapSaveValid ...
    saveFile doSaveFigs

function ret = run_analyzeCurve( action, varargin )
%RUN_ANALYZECURVE

% Bruce Po - igan@mit.edu

  include_globals;

  global tempMapi tempMap thisROIlist textLabels ...
      parameterType;

  persistent fh;
  persistent push_close push_getVal push_plot rad_pixelSelect ...
      rad_pixelCoordinates rad_ROI chk_logScale chk_showRange edit_x edit_y
  ret = '';

  if( isempty(action) ), action=' ';, end;
  switch( action )
   case 'init'                % init method --------------------------
    if( feval( mfilename, 'isopen') )
      figure( fh );
      return;
    end;
    fh = ui_analyzeCurve;     % start the UI
    placeFig( fh, 'top', 'right' );
    setupUI( fh, mfilename );
    ret = fh;

    % set up UI controls
    push_close = findobj( allchild(fh), 'Tag', 'push_close' );
    push_plot = findobj( allchild(fh), 'Tag', 'push_plot' );
    rad_pixelSelect = findobj( allchild(fh), 'Tag', 'rad_pixelSelect' );
    rad_pixelCoordinates = findobj(allchild(fh),'Tag','rad_pixelCoordinates');
    rad_ROI = findobj( allchild(fh), 'Tag', 'rad_ROI' );
    chk_logScale = findobj( allchild(fh), 'Tag', 'chk_logScale' );
    chk_showRange = findobj( allchild(fh), 'Tag', 'chk_showRange' );
    edit_x = findobj( allchild(fh), 'Tag', 'edit_x' );
    edit_y = findobj( allchild(fh), 'Tag', 'edit_y' );

    feval(mfilename, 'rad_pixelSelect' );

   case 'close'                       % close method -------------------------
    if( feval(mfilename,'isopen') )
      delete( fh );
      disp( [mfilename ' closed'] );
    end;

   case 'isopen'
    if( isempty(fh) ) ret = 0;                % fh = ''
    elseif( ~ishandle(fh) ) ret = 0; % fh = invalid handle
    else
      ret = (strcmp( get(fh, 'Name'), 'Analyze Curve Fit' ));
```

```
 end;

case 'raise'
 if( feval(mfilename,'isopen') )
   figure(fh);
   ret = 1;
 else
   ret = 0;
 end;

case 'rad_pixelSelect'
 set(rad_pixelSelect,'Value',1);
 set(rad_pixelCoordinates,'Value',0);
 set(rad_ROI,'Value',0);

case 'rad_pixelCoordinates'
 set(rad_pixelSelect,'Value',0);
 set(rad_pixelCoordinates,'Value',1);
 set(rad_ROI,'Value',0);

case 'rad_ROI'
 set(rad_pixelSelect,'Value',0);
 set(rad_pixelCoordinates,'Value',0);
 set(rad_ROI,'Value',1);

case { 'edit_x', 'edit_y' }
 run_analyzeCurve( 'rad_pixelCoordinates' );

case 'push_getVal'     % Just return the T1 or T2 info
 msize = dataParams.msize;

 % get x and y
 if( get(rad_ROI,'Value' ))
   a = tempMap(:);
   vals = a(tempMapi);
   msgbox( [parameterType ' mean: ' num2str(mean(vals)) ...
            ' standard deviation: ' num2str(std(vals))], ...
           [parameterType ' for Region'] );
   return;
 elseif( get(rad_pixelSelect,'Value') )
   h = dlgInfo( 'Select an image', '' );
   waitfor( 0, 'CurrentFigure' );
   figHandle = gcf;
   close( h );
   if( ~strcmp(get(gcf,'Name'),'Image') );
     return;
   end;
   figure( figHandle );
   [x,y] = ginput(1);      % user selects pixel from image now
   x = round(x); y = round(y);
   if( x<1 | y<1 | x>msize(2) | y>msize(1) )
     return;
   end;
 elseif( get(rad_pixelCoordinates,'Value'))
   x = str2num(get(edit_x,'String'));
   y = str2num(get(edit_y,'String'));
   if( isempty(x) | isempty(y) ) return; end;
   if( x<1 | y<1 | x>msize(1) | y>msize(2) ) return; end;
 end;

 val = tempMap(y,x);
 if( val~=0 )
   msgbox( [parameterType ' value at (' num2str(x) ',' num2str(y) '): ' ...
            num2str(val)], [parameterType ' Value'] );
 else
   errordlg( ['No ' parameterType ' calculated at (' num2str(x) ',' ...
              num2str(y) ')'], 'Note' )
 end;

case 'push_plot'
 if( isempty( thisROIlist )) return; end;
 msize = dataParams.msize;
 %set( push_curves, 'Enable', 'off' );

 % load source data (if needed)
 if( ~get(rad_ROI,'Value') | get(chk_showRange, 'Value') )
   if( ~figsValid )
     a = feval( ['load_' sourceParams.magnet] );
     if( a==0 )
       errordlg( 'Could not load source files', ...
                 'Unable to complete command', 'modal' );
```

```
      return;
    end;
    figsValid = 1;
  end;
end;

% get x and y (if needed)
if( get(rad_pixelSelect,'Value') )
  h = dlgInfo( 'Select an image', '' );
  waitfor( 0, 'CurrentFigure' );
  figHandle = gcf;
  close( h );
  if( ~strcmp(get(gcf,'Name'),'Image') );
    return;
  end;
  figure( figHandle );
  [x,y] = ginput(1);      % user selects pixel from image now
  x = round(x); y = round(y);
  if( x<1 | y<1 | x>msize(2) | y>msize(1) )
    %set( push_curves, 'Enable', 'on' );
    %set( push_curves, 'Value', 0 );
    return;
  end;
elseif( get(rad_pixelCoordinates,'Value'))
  x = str2num(get(edit_x,'String'));
  y = str2num(get(edit_y,'String'));
  if( isempty(x) | isempty(y) ) return; end;
  if( x<1 | y<1 | x>msize(1) | y>msize(2) ) return; end;
end;

%set( push_curves, 'Enable', 'on' );
%set( push_curves, 'Value', 0 );

% retrieve the data points
%   if( ndims(figs)==2 )
%     no_images = size(figs,2);
%   else
%     no_images = size(figs,3);
%   end;
no_images = length(dataParams.data_sequence);
dataPoints = zeros( no_images,1 );
data_mean = zeros( no_images,1 );
data_std = zeros( no_images,1 );

%selectedROIs = get( list_ROIlist, 'Value' );
selectedROIs = run_viewResults( 'getSelectedROIs' );    % call method
pixelsi = unionall(ROIlist(selectedROIs).maski);

% (1) or (2) or show data range
if( ~get(rad_ROI, 'Value') | get(chk_showRange, 'Value'))
  %for(i = 1:no_images)
   for i = 1:length(dataParams.data_sequence)
    if( ndims(figs)==2 )
      %A = reshape( figs(:,i), msize );
      A = reshape( figs(:,dataParams.data_sequence(i)), msize );
    else
      %A = figs(:,:,i);
      A = figs(:,:,dataParams.data_sequence(i));
    end;
    if( ~get(rad_ROI,'Value' ))          % (1) and (2)
      dataPoints(i) = A(y,x);
    end;
    if( get(chk_showRange, 'Value' ))    % show data range
      data = A(:);
      data_mean(i) = mean(data(pixelsi));
      data_std(i) = std(data(pixelsi));
    end;
  end;
end;

if( get(rad_ROI, 'Value' ))              % (3)
  [c ia ib] = intersect(theMapi, tempMapi);
  pfitted = pfittedSave(ia,:);
  pMean = mean(pfitted);
  pStd = std(pfitted);
  [i j V] = find( tempMap );
  Tmean = mean(V);
  Tstd = std(V);
end;

% set up new figure
```

```
    new_h = figure;
    if( ~get(rad_ROI,'Value' ))
      title( 'Relaxation curve for pixel' );

      infoStr = ['Coordinates: (' num2str(x) ',' num2str(y) ')'];
      i = getIndex( x, y );  % this just converts x,y to linear index
      ii = find(theMapi==i);
      if( isempty(ii) )
        val = '';
        p = '';
      else
        val = theMapSave(y,x);% the T1 or T2 value for that pixel
        p = pfittedSave(ii,:);% estimated parameters for that pixel
        p = p(1,:);    % it is possible to have multiple values
                       % returned, but they should be identical
         infoStr = strvcat( infoStr, ...
                          [parameterType ':' num2str(val) '  '], ...
                          ['pfitted: [' num2str(p) ']  '] );
      end; %if
      figure( new_h );
      axis off;
      text( 0, 0, infoStr );
      axes( 'position', [.1, .25, .8, .65]);
      feval( ['plot_' sourceParams.magnet], dataPoints, p, ...
            get(chk_logScale,'Value'));
    else
      infoStr = strvcat( [parameterType ' mean: ' num2str(Tmean) ...
                          '     std dev: ' num2str(Tstd) ], ...
                         ['pfitted mean:     ' num2str(pMean)], ...
                         ['pfitted std dev:  ' num2str(pStd)] );
      figure( new_h );
      title( 'Relaxation curve for region of interest' );
      axis off;
      text( 0, 0, infoStr );
      axes( 'position', [.1, .25, .8, .65]);
      feval( ['plot_' sourceParams.magnet], '', pMean, ...
            get(chk_logScale,'Value'));
    end;


    if( get(chk_showRange, 'Value' ))
      hold on;
      % make sure TI is the same for all (IR, SR, T2)
      t = gettimeaxis;
      errorbar( t(dataParams.data_sequence), data_mean, data_std, '+' );
      hold off;
    end;

    %axis( [0 ...
    %     length(dataPoints)+1 ...
    %     min(dataPoints)-range(dataPoints)*.1 ...
    %     max(dataPoints)+range(dataPoints)*.1 ] );

   case 'push_close'
    feval(mfilename, 'close' );

  end;

function i = getIndex( x, y )
% converts x,y to linear index
  global dataParams;
  msize = dataParams.msize;
  i = (x-1)*msize(1)+(y-1)+1;

function ret = run_analyzeParams( action, varargin )
%RUN_ANALYZEPARAMS

% Bruce Po - igan@mit.edu

  include_globals;

  global tempMapi tempMap thisROIlist textLabels ...
      parameterType;

  persistent fh dataUnion;
  persistent push_close pop_fieldNo edit_fieldInfo push_plotField ...
      push_plotHist push_displayMap push_saveStats push_plotProfile;
  ret = '';

  if( isempty(action) ), action=' ';, end;
  switch( action )
```

```
case 'init'                % init method -------------------------
 if( feval(mfilename,'isopen'))
   figure( fh );
   return;
 end;
 fh = ui_analyzeParams;    % start the UI
 placeFig( fh, 'top', 'center' );
 setupUI( fh, mfilename );
 ret = fh;

 % set up UI controls
 push_close = findobj( allchild(fh), 'Tag', 'push_close' );
 pop_fieldNo = findobj( allchild(fh), 'Tag', 'pop_fieldNo' );
 edit_fieldInfo = findobj( allchild(fh), 'Tag', 'edit_fieldInfo' );
 push_plotField = findobj( allchild(fh), 'Tag', 'push_plotField' );
 push_plotHist = findobj( allchild(fh), 'Tag', 'push_plotHist' );
 push_displayMap = findobj( allchild(fh), 'Tag', 'push_displayMap' );
 push_saveStats= findobj( allchild(fh), 'Tag', 'push_saveStats' );
 push_plotProfile = findobj( allchild(fh), 'Tag', 'push_plotProfile' );

 % set up field number pop up ....................................
 temp = [parameterType ' map'];
 numFields = size(pfittedSave,2);        % get number of fields
 for( i=1:numFields )
   temp = strvcat( temp, ['param ' num2str(i)] );
 end;
 set( pop_fieldNo, 'String', temp );
 set( pop_fieldNo, 'Value', 1 );

 run_analyzeParams( 'update' );

case 'isopen'
 if( isempty(fh) ) ret = 0;              % fh = ''
 elseif( ~ishandle(fh) ) ret = 0; % fh = invalid handle
 else
   ret = (strcmp( get(fh, 'Name'), 'Analyze Parameters' ));
 end;

case 'raise'
 if( feval(mfilename,'isopen') )
   figure(fh);
   ret = 1;
 else
   ret = 0;
 end;

case 'close'                        % close method -------------------------
 if( feval(mfilename,'isopen'))
   delete( fh );
   disp( [mfilename ' closed'] );
 end;

case 'update'

 if( ~feval(mfilename,'isopen')) return; end;

 % set up dataunion, a temporary array for analyzing selected pfitted data
 selectedROI = run_viewResults( 'getSelectedROIs' );
 if( isempty(selectedROI) )
   set( edit_fieldInfo, 'String', ' ' );
   %tempMap = zeros(size(img_background));
   dataUnion = '';
   return;
 end;

 whichField = get( pop_fieldNo, 'Value' );
 if( whichField==1 )
   %[i j dataUnion] = find(tempMap);
   dataUnion = tempMap(tempMapi);
   [i,j,v] = find(dataUnion);     % don't count zeros
   dataUnion = v;
 else
   % **** can rewrite this portion to be more efficient
   %temp = zeros( length(theMapSave(:)), 1 );
   %temp(theMapi) = pfittedSave(:, whichField-1);
   %temp(setdiff(theMapi,tempMapi)) = 0;  % remove the unselected ROIs
   %dataUnion = temp(tempMapi);
   [c ia ib] = intersect(theMapi, tempMapi);
   dataUnion = pfittedSave(ia,whichField-1);
 end
```

```
 % display information in UI box
 set( edit_fieldInfo, ...
      'String', strvcat( ...
          ['no. pixels: ' num2str(length(dataUnion))], ...
          ['mean:       ' num2str(mean(dataUnion))], ...
          ['std dev:    ' num2str(std(dataUnion)) ] ));

case 'push_close'
 run_analyzeParams( 'close' );

case 'pop_fieldNo'
 run_analyzeParams( 'update' );

case 'push_saveStats'
 [filen dirn] = uiputfile( '*.txt', 'Save Results Analysis' );
 if( filen~=0 )
   outputFile = fullfile( dirn,filen);
   fid = fopen( outputFile, 'w' );
   fprintstring( fid, struct2string(sourceParams) );     % write sourceParams
   fprintf( fid, '\n' );
   fprintstring( fid, struct2string(dataParams) );       % write dataParams
   fprintf( fid, '\n' );

   temp = get( pop_fieldNo, 'Value' );    % save pop_fieldNo

   ROIstr = run_viewResults( 'getROIstring' );

   fprintstring( fid, ROIstr );
   fprintf( fid, '\n' );

   fieldStr = get( pop_fieldNo, 'String' );
   for i=1:size(fieldStr,1)
     set( pop_fieldNo, 'Value', i );              % set UI pop_fieldNo
     run_analyzeParams( 'update' );               % update UI
     fprintf( fid, '%s --------------\n', fieldStr(i,:) );       % field name
     fprintstring( fid, get(edit_fieldInfo, 'String') ); % field info
     fprintf( fid, '\n' );
   end;
   fclose( fid );

   set( pop_fieldNo, 'Value', temp );     % restore pop_fieldNo
   run_analyzeParams( 'update' );
 end;

case 'push_plotField'
 if( isempty( thisROIlist )) return; end;
 % dataUnion should already be updated
 figure;
 plot( dataUnion, '.' );
 a = get( pop_fieldNo, 'Value' );
 b = get( pop_fieldNo, 'String' );
 c = b(a,:);
 title( [c ' - values'] );
 ylabel( [c ' value'] );
 xlabel( 'Pixel number' );

 %axis( [0 ...
 %      length(dataUnion)+1 ...
 %      min(dataUnion)-range(dataUnion)*.1 ...
 %      max(dataUnion)+range(dataUnion)*.1 ] );

case 'push_plotHist'
 figure;
 hist( dataUnion, 200 );
 a = get( pop_fieldNo, 'Value' );
 b = get( pop_fieldNo, 'String' );
 c = b(a,:);
 title( [c ' - histogram'] );
 ylabel( 'Pixel count' );
 xlabel( [c ' value'] );

case 'push_displayMap'
 whichField = get( pop_fieldNo, 'Value' );
 if( whichField==1 )              % display T1/T2 map
   displayMap = tempMap;
 else                       % display param field
   [c ia ib] = intersect(theMapi, tempMapi);
   displayMap = zeros(dataParams.msize);
   displayMap(c) = pfittedSave(ia,whichField-1);
 end
```

```
    new_h = run_image('init','another');
    placeFig( new_h, 'bottom', 'right' );
    run_image('display', 'handle', new_h, ...
              struct('map_max',mean(dataUnion)+ 2*std(dataUnion)), ...
              img_background, displayMap );
    overlaycolorbar;
    a = get( pop_fieldNo, 'Value' );
    b = get( pop_fieldNo, 'String' );
    c = b(a,:);
    title( [c ' - map'] );

   case 'push_plotProfile'
    msize = dataParams.msize;

    h = dlgInfo( 'Select an image and draw a line', '', [200 40 10] );
    waitfor( 0, 'CurrentFigure' );
    figHandle = gcf;
    close( h );
    if( ~strcmp(get(gcf,'Name'),'Image') );
      return;
    end;
    figure( figHandle );
    [x,y] = getline( figHandle );     % user selects line from image
    x = round(x); y = round(y);
    if( sum(x<1 | y<1 | x>msize(2) | y>msize(1)) )
      return;
    end;

    data = improfile( theMapSave, x(1:2), y(1:2) );
    figure;
    plot( data );
    title( 'Profile' );
    ylabel( 'Value' );

  end;

function ret = run_dataSource( action )
%RUN_DATASOURCE

  include_globals;
  persistent fh;
  persistent pop_magnets list_dataParams push_loadParam ...
      list_sourceParams push_sourceDir txt_message push_load push_close;
  persistent magnet_list %magnet
  ret = '';

  if( isempty(action) ), action=' ';, end;
  switch( action )
   case 'init'
    if( run_dataSource('raise'))
      return;  % if window already exists
    end;

    fh = ui_dataSource;                        % start UI
    placeFig( fh, 'center', 'left' );
    setupUI( fh, mfilename );

    definitions;                       % loads magnet definitions

    % set up UI controls
    pop_magnets = findobj( allchild(fh), 'Tag', 'pop_magnets' );
    list_dataParams = findobj( allchild(fh), 'Tag', 'lst_dataParams' );
    txt_message = findobj( allchild(fh), 'Tag', 'txt_message' );
    list_sourceParams = findobj( allchild(fh), 'Tag', 'list_sourceParams' );
    push_sourceDir = findobj( allchild(fh), 'Tag', 'push_sourceDir' );
    push_loadParam = findobj( allchild(fh), 'Tag', 'push_loadParam' );
    push_load = findobj( allchild(fh), 'Tag', 'push_load' );
    push_close = findobj( allchild(fh), 'Tag', 'push_close' );
    set( pop_magnets, 'String', strvcat(magnet_list) ); % see definitions.m
    set( list_dataParams, 'String', ' ' );
    set( txt_message, 'String', ' ' );

    %if( isempty( dataParams)),  run_dataSource( 'push_resetDataParams' );
    %else,                    run_dataSource( 'update' );
    %end;
    %dataSource_isDone = '';
    if( sourceParamsValid )
      run_dataSource( 'update' );
    else
      run_dataSource( 'push_resetDataParams' );
    end;
```

```
 ret = fh;

case 'isopen'
 if( isempty(fh) ) ret = 0;               % fh = ''
 elseif( ~ishandle(fh) ) ret = 0; % fh = invalid handle
 else
   ret = (strcmp( get(fh, 'Name'), 'Data Source' ));
 end;

case 'raise'
 if( run_dataSource('isopen') )
   figure(fh);
   ret = 1;
 else
   ret = 0;
 end;

case 'close'
 if( run_dataSource('isopen'))
   delete( fh );
   disp( 'run_dataSource closed' );
 end;
 return

case 'update'      % update UI to reflect program values
 if( ~run_dataSource('isopen')) return; end;

 set( pop_magnets, 'Value', find(strcmp(magnet_list, sourceParams.magnet)));
 set( list_dataParams, 'String', struct2string( dataParams ) );
 set( list_sourceParams, 'String', struct2string( sourceParams ) );

 % update the message box
 if( isSourceEmpty( sourceParams ))
   set( txt_message, 'String', ['Data source needs to be set.' ...
               ' Select the magnet and click ''Source files...''.'] );
   set( push_load, 'Enable', 'Off' );
   sourceParamsValid = 0;
 elseif( isempty( dir(sourceParams.sourceDir)))
   set( txt_message, 'String', ['Data source directory is invalid'] );
   set( push_load, 'Enable', 'Off' );
   %sourceParamsValid = 0;    ** Do not change valid status **
 elseif( ~sourceParamsValid )
   set( txt_message, 'String', ['Parameters have been modified.  Click' ...
               ' ''Load'' to load data.'] );
   set( push_load, 'Enable', 'On' );
 else
   set( txt_message, 'String', '[Unmodified]' );
   set( push_load, 'Enable', 'On' );
 end;

 % update other stuff in the program
 run_main( 'update' );

case 'pop_magnets'
 if( ~confirmChange ), return;, end;
 a = get( pop_magnets, 'Value' );
 temp = magnet_list{a};
 if( ~strcmp(sourceParams.magnet, temp) ) % user changed something?
   sourceParamsValid = 0;
   run_dataSource( 'push_resetDataParams' );
 end
 run_main( 'update' );

case 'list_sourceParams'
 if( ~confirmChange ), return;, end;

 a = get( list_sourceParams, 'Value' );   % which field selected?
 fields = fieldnames(sourceParams);
 fieldname = fields{a};
 switch( fieldname )
  case { 'sourceDir', 'sourceFile', 'sourceFiles' }
   temp = feval( ['def_' sourceParams.magnet], 'input', fieldname );
   if( temp==1 )                    % user changed something?
     sourceParamsValid = 0;
     run_dataSource( 'update' );
   end;

  %case { 'sourceDir', 'sourceFile' }
  % tempParams = inputFieldDlg( sourceParams, fieldname, 'confirm', ...
     %                        paramPrompts );
  % if( isstruct(tempParams ))     % user changed something?
```

```
    %sourceParams = tempParams;
    %run_dataSource( 'update' );
  % end;
  otherwise
   % ignore
 end;

case 'push_sourceDir'
 if( ~confirmChange ), return;, end;
 %try
   temp = feval( ['def_' sourceParams.magnet], 'input', 'get_sourceDir' );
 %catch
 %  errordlg( 'Invalid filename for this magnet', 'Error', 'modal' );
 %  temp = 1;
 %end;
 if( temp==1 )                    % user changed something?
   sourceParamsValid = 0;
   run_dataSource( 'update' );
 end;

case 'lst_dataParams'      % User wants to change a field
 a = get( list_dataParams, 'Value' );     % which field selected?
 fields = fieldnames(dataParams);
 fieldname = fields{a};

 % Some fields affect the outcome of the calculated results while
 %  other fields are harmless to change, such as maxImgVal and maxTval
 %if( strcmp(fieldname, 'maxImgVal') | strcmp(fieldname, 'maxTval'))
 %  importantField = 0;
 %else
 %  importantField = 1;
 %end;
 importantField = 1;

 if( importantField )      % confirmChange confirms with the user
   if( ~confirmChange ) return; end;
 end;

 % run the function to input data for this field
 temp = feval( ['def_' sourceParams.magnet], 'input', fieldname );
 if( temp==1 )                    % user changed something?

   % fix up the user input if necessary
   switch( fieldname )
    case 'msize'
     a = dataParams.msize;
     if( length(a)>1 )
       dataParams.msize = [a(1) a(2)];
     else
       dataParams.msize = [a(1) a(1)];
     end;
    case {'slice', 'no_of_slices'}
     if( dataParams.slice > dataParams.no_of_slices )
       dataParams.slice = dataParams.no_of_slices;
     end;
    case {'no_of_timeVar', 'data_sequence'}
     seq = dataParams.data_sequence;
     if( ~sum(seq<=0) & ~sum(seq>dataParams.no_of_timeVar) )
       dataParams.data_sequence = round(seq);
     else
       dataParams.data_sequence = 1:dataParams.no_of_timeVar;
     end;
   end;

   if( importantField ) sourceParamsValid = 0; end;
   run_dataSource( 'update' );
   if( ~importantField )
    if( theMapSaveValid ) % means data has already been calculated, so
                          %  save the changes
      save( saveFile, 'dataParams', '-append' );
      run_viewResults( 'update' );
    end;
   end;
 end;

case 'push_loadParam'
 if( ~confirmChange ), return;, end;
 [filen, pathn] = uigetfile('results*.mat', ['Select file to load' ...
                  ' parameters from'] );
 if( filen~=0 )                   % not a cancel?
   sourceParamsValid = 0;
```

```
      s = load( [pathn filen], 'sourceParams', 'dataParams' );
      if( ~isfield(s,'sourceParams') | ~isfield(s,'dataParams' ))
        errordlg( [filen ' does contain the required data'], ...
                  'Error', 'modal' );
        return;
      end;
      sourceParams.magnet = s.sourceParams.magnet;  % set loaded magnet
      feval( ['def_' sourceParams.magnet], 'default' );     % get magnet defaults
      sourceParams = structcombine( sourceParams, s.sourceParams, 1 );
      dataParams = structcombine( dataParams, s.dataParams, 1 );
      run_dataSource( 'update' );
    end;

   case 'push_resetDataParams'
    if( ~confirmChange ), return;, end;
    a = get( pop_magnets, 'Value' );

    sourceParams.magnet = magnet_list{a};
    feval( ['def_' sourceParams.magnet], 'default' );       % get magnet defaults
    sourceParamsValid = 0;
    set( list_dataParams, 'Value', 1 );
    set( list_sourceParams, 'Value', 1 );
    run_dataSource( 'update' );

   case 'push_load'
    if( strcmpi(get(push_load, 'Enable'), 'On') )
      set(fh,'Pointer','watch');
      a = feval( ['load_' sourceParams.magnet] );
      set(fh,'Pointer','arrow');
      if( a==0 ) return; end;
      sourceParamsValid = 1;
      figsValid = 1;
      doSaveFigs = 0;
      run_dataSource( 'close' );
      run_ROIlist( 'close' );
      run_selectData( 'close' );
      run_viewResults( 'close' );
    end;

    run_image('close');
    run_image( 'display', struct('imgmax',viewParams.maxImgVal), ...
               img_background );
    run_main( 'update' );

   case 'push_close'
    run_dataSource( 'close' );

   otherwise
    % ignore unrecognized action

  end; %switch

function ret = confirmChange
  % returns 1 if it's okay to make changes
  include_globals;
  %if( isempty(theMapSave) & isempty(figs) )
  if( theMapSaveValid | figsValid )  % is there data already loaded?
    resp = questdlg( ['This action will nullify the data already loaded!' ...
                      '  Do you wish to continue?'], ...
                     'Warning', 'No' );
    if( strcmp(resp,'Yes'))
      ret = 1;
      run_viewResults( 'close' );
    else ret = 0;
    end; %if
  else
    ret = 1;
  end; %if

  if( ret==1 )
    %sourceParamsValid = 0;
    theMapSaveValid = 0;
    ROIlistValid = 0;
    figsValid = 0;
    saveFile = '';
    run_main( 'update' );
  end;


function ret = isSourceEmpty( sourceParams )
  if( isempty( sourceParams.sourceDir) )
```

```
    ret = 1;
  else
    if( isfield( sourceParams, 'sourceFile' ))
      ret = isempty( sourceParams.sourceFile );
    elseif( isfield( sourceParams, 'sourceFiles' ))
      ret = isempty( sourceParams.sourceFiles );
    end;
  end;

  function ret = run_image( action, varargin )
% RUN_IMAGE
%
%   H = RUN_IMAGE( 'init' )                          % for primary figure
%   H = RUN_IMAGE( 'init', 'another' )
%   RET = RUN_IMAGE( ACTION, ... )                   % for primary figure
%   RET = RUN_IMAGE( ACTION, 'handle', H, ... )
%
%   RUN_IMAGE( 'display', OPTIONS, BACKGROUND_IMG, MAP_IMG ) OPTIONS is a
%   structre in the format or just [] to use defaults.  To display only
%   BACKGROUND_IMG, omit MAP_IMG.
%
%
%
%   The fields in OPTIONS are:
%   'imgmax' --   input image max intensity
%   'mapmax' --   input map max intensity
%   'colormap' -- 'hsv', 'green', or user defined N-by-3 color map

  include_globals
  persistent fh              % fh is figure handle to primary figure
  ret = '';
  h = fh;              % h is generic figure handle
  isMainFigure = 1;    % if 'handle' argument is used, then isMainFigure
                       %  will be set to 0

  totalColors = 512;   % actual displaced T1/T2 map uses this many colors
                       % (must be a multiple of 256)

  args = varargin;
  if( nargin>1 )
    if( ischar(varargin{1}))
      if( strcmp('handle', varargin{1}))
        h = varargin{2};
        args = {varargin{3:length(varargin)}};
        isMainFigure = 0;
      end;
    end;
  end;

  switch( action )

    % init closes current window if it exists
   case 'init'
    temp = 1;
    if( nargin>1 )
      if( strcmp('another',varargin{1}) )
        h = figure;
        temp = 0;
      end;
    end;
    if( temp )
      run_image('close');     % close current figure window
      h = figure;
      fh = h;
    end;

    iptsetpref( 'ImshowTrueSize', 'auto' );
    set( h, 'Name', 'Image' );
    set( h, 'DeleteFcn', ['close(' num2str(h) ');axis off;'] );
    placeFig( h, 'center', 'right' );
    ret = h;

   case 'isopen'
    if( isempty(h) ) ret = 0;         % if h = ''
    elseif( ~ishandle(h) ) ret = 0;   % if h is invalid handle
    else
      ret = (strcmp( get(h, 'Name'), 'Image' ));
    end;

   case 'raise'
    if( run_image('isopen', 'handle', h ))
```

```
  figure(h);
  ret = h;
else
  ret = 0;
end;

case 'close'
 if( run_image('isopen', 'handle', h) )
  close(h);
 end;

case 'display'
 % Display the image.  If the handle is for a figure and it doesn't exist,
 % create it.  If the handle is for an axis and it doesn't exist, then
 % return 0.

 createNewFig = 0;
 if( isvalidfigure(h) )
   if( ~run_image('raise', 'handle', h ))
    createNewFig = 1;      % if raise failed, then need new figure
   end;
 else
   if( isaxeshandle(h) )   % if the handle looks like it's for an axis
    if( isvalidaxes(h) )
      axes(h)              % ok
    else
      ret = 0;             % axis doesn't exist
      return
    end;
   else                    % the handle is for a figure
    createNewFig = 1;
   end;
 end;
 if( createNewFig )
   % the old h can be discarded
   h = run_image('init','another');
   if( isMainFigure )
    fh = h;
   end;
 end;

 set(fh,'Pointer','watch');

 % parse arguments ..........................................
 OPTIONS = args{1};
 bgImg = args{2};
 if(length(args)>2)
   mapImg = args{3};
 else
   mapImg = '';
 end

 if( isfield(OPTIONS,'imgmax'))          % set bgScale
   bgScale = OPTIONS.imgmax;
   if(strcmp(bgScale,'auto'))
    bgScale = getImgMax( bgImg );
   end;
 else
   bgScale = getImgMax( bgImg );
 end

 if( ~isempty(mapImg))                   % set mapScale
   if( isfield(OPTIONS,'mapmax') )
    mapMaxVal = OPTIONS.mapmax;
    if(strcmp(mapMaxVal,'auto'))
      mapMaxVal = getMapMax( mapImg );
    end;
   else
    mapMaxVal = getMapMax( mapImg );
   end
   mapscale = totalColors / mapMaxVal;
 end;

 if( isfield(OPTIONS,'colormap') )       % set colormap
   colormapType = OPTIONS.colormap;
   switch( colormapType )
    case 'green'
     cm = greenColormap(totalColors);
    case 'hsv'
     cm = hsvColormap(totalColors);
    otherwise
```

```
      cm = colormapType;     % user defined
    end;
  else
    cm = hsvColormap(totalColors);
  end;

  % create background image ....................................
  tempImg = bgImg/bgScale;          % rescale to [0,1]
  tempImg = min(tempImg, 1.0 );             % clip bright pixels
  dims = size(bgImg);
  clear bgImg;

  if( isempty(mapImg))
    display_img = zeros( dims(1), dims(2), 3 );
    display_img(:,:,1) = tempImg;           % it's in gray, so R=G=B
    display_img(:,:,2) = tempImg;
    display_img(:,:,3) = tempImg;
    clear tempImg;
  else
    % overlay with map ..........................................
    mapImg = round( mapImg * mapscale );   % range 1:512
    mapImg = max( mapImg, 0 );             % clip dark
    mapImg = min( mapImg, size(cm,1) );    % clip light
%     tic
%     [y,x] = find(mapImg);
%     for i = 1:length(y)
%      val = mapImg(y(i),x(i));
%      display_img(y(i),x(i),:) = cm(val,:);
%     end;
%     toc

     mapImg = mapImg(:);
     [mapi j vals] = find(mapImg);
     clear mapImg j;
     tempImg = tempImg(:);
     display_r = tempImg;
     display_r(mapi) = cm(vals,1);
     display_g = tempImg;
     display_g(mapi) = cm(vals,2);
     display_b = tempImg;
     display_b(mapi) = cm(vals,3);
     display_img = zeros( dims(1), dims(2), 3 );
     display_img(:,:,1) = reshape(display_r,dims);
     display_img(:,:,2) = reshape(display_g,dims);
     display_img(:,:,3) = reshape(display_b,dims);
     clear map_r map_g map_b display_r display_g display_b mapi;

    set( gcf, 'Userdata', 256/mapMaxVal );
  end

  if( isvalidfigure(h) )
    figure(h);
    % colormap command requires color map to be 256 in size of less
    colormap( cm(1:totalColors/256:totalColors,:));
  elseif( isvalidaxes(h) )
    axes(h);
  end;
  imshow( display_img );
  ret = 1;

  set(fh,'Pointer','arrow');

 case 'colorbar'
  figure( h );
  colorbar;

  cbScale = get( h, 'Userdata' );
  if( isempty(cbScale) )
    cbScale = 1;
  end;

  ah = colorbar;
  set( ah, 'YTicklabel', strvcat( num2str( 1/cbScale*50,3 ), ...
                                  num2str( 1/cbScale*100,3 ), ...
                                  num2str( 1/cbScale*150,3 ), ...
                                  num2str( 1/cbScale*200,3 ), ...
                                  num2str( 1/cbScale*250,3 )));
  ret = 1;

end; %switch( action )
```

```
function ret = isvalidfigure( h )
  if( ishandle(h) )
    ret = strcmp(get(h,'Type'),'figure');
  else
    ret = 0;
  end;

function ret = isvalidaxes( h )
  if( ishandle(h) )
    ret = strcmp(get(h,'Type'),'axes');
  else
    ret = 0;
  end;

function ret = isaxeshandle(h)
  % Axes handles are non-integers while figure handles are integers.
  if( ~isempty(h) )
    ret = (h~=fix(h));
  else
    ret = 0;
  end;

function cm = greenColormap( totalColors )
% This is a colormap for highlighting regions of interest
  cm = zeros(totalColors,3);
  cm(:) = .2;
  cm(:,2) = (.2:(.8/511):1.0)';

function cm = hsvColormap( totalColors )

  hsvEnd = .7;         % use for 70% of standard HSV

  % insert sections of red, yellow, and cyan into the standard HSV
  percentRed =     .10;
  percentYellow =  .0333;
  percentCyan =    .0333;

  hsvLength = round(totalColors*(1-percentRed-percentYellow-percentCyan));
  hsvSection = hsv( round(hsvLength/hsvEnd) );
  hsvSecton = imadjust(hsvSection,[0.25,1],[0,1],1);

  redLength = round(percentRed*totalColors);
  redSection = zeros( redLength,3);
  redSection(:,1) = (1/redLength:1/redLength:1)';

  yellowLength = round(percentYellow*totalColors);
  yellowSection = zeros(yellowLength, 3);
  yellowSection( :,1:2 ) = 1;

  cyanLength = round(percentCyan*totalColors);
  cyanSection = zeros( cyanLength, 3 );
  cyanSection( :,2:3 ) = 1;

  a = find( hsvSection(:,1)<hsvSection(:,2)); % find the place to insert
  a = a(1);                                   %  yellow into
  b = find( hsvSection(:,3)>hsvSection(:,2)); % find the place to insert
  b = b(1);

  cm = [ redSection ; hsvSection(1:a-1,:) ; yellowSection ; ...
         hsvSection(a:b-1,:) ; cyanSection ; hsvSection(b:hsvLength,:)];

  function ret = run_main( action )
%RUN_MAIN

  include_globals;
  persistent push_dataSource push_selectData txt_sourceDir txt_magnet ...
      push_ROIlist push_loadResults txt_ROImsg push_viewResults ...
      txt_resultsSaved
  persistent fh;
  ret = '';

  if( isempty(action) ), action=' ';, end;
  switch( action )
   case 'init'

    % set up a whole slew of variables
    sourceParamsValid = 0;
    figsValid = 0;
    ROIlistValid = 0;
    theMapSaveValid = 0;
```

```
 fh = ui_main;
 placeFig( fh, 'top', 'left' );
 setupUI( fh, mfilename );
 ret = fh;

 % Set up event callbacks.

 % set up UI controls
 childs = allchild(fh);
 push_dataSource = findobj( childs, 'Tag', 'push_dataSource' );
 push_selectData = findobj( childs, 'Tag', 'push_selectData' );
 txt_sourceDir = findobj( childs, 'Tag', 'txt_sourceDir' );
 txt_magnet = findobj( childs, 'Tag', 'txt_magnet' );
 push_ROIlist = findobj( childs, 'Tag', 'push_ROIlist' );
 push_loadResults = findobj( childs, 'Tag', 'push_loadResults' );
 txt_ROImsg = findobj( childs, 'Tag', 'txt_ROImsg' );
 push_viewResults = findobj( childs, 'Tag', 'push_viewResults' );
 txt_resultsSaved = findobj( childs, 'Tag', 'txt_resultsSaved' );

 run_main( 'update' );

case 'close'
 run_dataSource( 'close' );
 run_ROIlist( 'close' );
 run_selectData( 'close' );
 run_viewResults( 'close' );
 if( run_main( 'isopen' ));
   delete( fh );
   disp( 'run_main closed' );     % for debugging
 end;

case 'isopen'
 if( isempty(fh) ) ret = 0;               % fh = ''
 elseif( ~ishandle(fh) ) ret = 0; % fh = invalid handle
 else
   ret = (strcmp( get(fh, 'Name'), 'Main' ));
 end;

case 'raise'
 if( run_main('isopen') )
   figure(fh);
   ret = 1;
 else
   ret = 0;
 end;

case 'update'

 if( ~sourceParamsValid ) saveFile = ''; end;

 % Data Source panel ..................................
 if( sourceParamsValid )
   temp1 = ['Data type: ' sourceParams.magnet];
   % to help word wrapping in text box:
   temp = strrep( sourceParams.sourceDir,'\','\ ' );
   temp = strrep( temp,'/','/ ' );
   temp2 = strvcat( 'Loaded from:', temp );
   set( push_selectData, 'Enable', 'On' );
 else
   temp1 = 'No source data set';
   temp2 = '';
   set( push_selectData, 'Enable', 'Off' );
 end;
 if( ~figsValid )
   run_selectData( 'warning' );
 end;

 set( txt_magnet, 'String', temp1 );
 set( txt_sourceDir, 'String', temp2);

 % Region of Interest panel ..............................
 if( sourceParamsValid )
   set( push_ROIlist, 'Enable', 'On' );
 else
   set( push_ROIlist, 'Enable', 'Off' );
 end;
 if( ROIlistValid )
   if( sum([ROIlist.modified])>0 )
     set( txt_ROImsg, 'String', ['Parameter map has' ...
                ' not been generated for some regions of interest'] );
```

```
  else
    set( txt_ROImsg, 'String', ['Parameter map has been generated for' ...
                  ' all regions of interest'] );
  end; %if
 else
   set( txt_ROImsg, 'String', 'No regions of interest defined' );
   run_ROIlist( 'warning' );      % if the window is open, then send warning
 end; %if

 % View Results panel ..............................
 %if( isempty(theMapSave) )
 if( theMapSaveValid )
   set( push_viewResults, 'Enable', 'On' )
 else
   set( push_viewResults, 'Enable', 'Off' )
 end;
 if( run_viewResults( 'isopen' ))
   if( theMapSaveValid==0 )
     run_viewResults( 'close' );
   elseif( ROIlistValid==2 | theMapSaveValid==2 )
     run_viewResults( 'warning' );
   end;
 end;
 if( ROIlistValid==2 ) ROIlistValid = 1; end;
 if( theMapSaveValid==2 ) theMapSaveValid = 1; end;

 if( isempty(saveFile) )
   set( txt_resultsSaved, 'String', 'No output has been generated' );
 else
   temp = strrep( saveFile,'\','\ ' ); % to help word wrapping in text box
   temp = strrep( temp,'/','/ ' );     % to help word wrapping in text box
   temp = strvcat( 'Output file:', temp );
   set( txt_resultsSaved, 'String', temp );
 end;

case 'push_dataSource'
 run_dataSource( 'init' );

case 'push_selectData'
 run_selectData( 'init' );

case 'push_ROIlist'
 %if( ~isempty(figs) )
 if( sourceParamsValid )
   run_ROIlist( 'init' );
 end;

case 'push_loadResults'
 %if( ~isempty(figs))
 if( sourceParamsValid )
   temp = questdlg( ['This action will replace the data already loaded!' ...
                     '  Do you wish to continue?'], ...
                    'Warning', 'Yes', 'Cancel', 'Cancel' );
   if( ~strcmp(temp,'Yes')), return;, end;
 end; %if
 [filen dirn] = uigetfile( 'results*.mat', 'Open Calculation Results' );
 if( filen~=0 )
   saveFile = fullfile(dirn,filen);              % global variable

   % check to make sure this file is valid
   s = whos( '-file', saveFile );
   vars = {s.name};
   if( isempty(strmatch('sourceParams',vars)) | ...
       isempty(strmatch('dataParams',vars)) | ...
       isempty(strmatch('viewParams',vars)) | ...
       isempty(strmatch('ROIlist',vars)) | ...
       isempty(strmatch('theMapSave',vars)) | ...
       isempty(strmatch('theMapi',vars)) | ...
       isempty(strmatch('pfittedSave',vars)) | ...
       isempty(strmatch('img_background',vars)) )
     errordlg( [saveFile ' does contain the required data'], ...
               'Error', 'modal' );
     return;
   end;

   run_dataSource( 'close' );
   run_selectData( 'close' );
   run_ROIlist( 'close' );
   run_viewResults( 'close' );

   set(fh,'Pointer','watch');
```

```
      load( saveFile, 'sourceParams', 'dataParams', 'viewParams', ...
            'ROIlist', 'theMapSave', 'theMapi', 'pfittedSave', ...
            'img_background' );
      sourceParamsValid = 1;
      ROIlistValid = 2;
      theMapSaveValid = 2;

      % load figs file if it exists
      figsValid = 0;
      [pathn,name,ext] = fileparts(saveFile);
      figsFile = [pathn '/' name '-figs.mat'];
      if( ~isempty(dir(figsFile)) )
        s = whos( '-file', figsFile );
        vars = {s.name};
        if( ~isempty(strmatch('figs',vars)))
          load( figsFile, 'figs' );
          figsValid = 1;
        end;
      end;

      run_image('close');
      run_image( 'display', struct('imgmax',viewParams.maxImgVal), ...
                 img_background );

      run_main( 'update' );
      set(fh,'Pointer','arrow');

    end;

   case 'push_viewResults'
    %if( ~isempty(theMapSave) )
    if( theMapSaveValid )
      run_viewResults( 'init' );
    end;

   otherwise
    % ignore unrecognized actions

  end; %switch

function ret = run_ROIlist( action, varargin )
%RUN_ROILIST

  include_globals;
  persistent fh itemCount newROIlist newDataParams;
  persistent push_addROI push_delROI push_addEntire list_ROIlist ...
      push_loadFile push_exe push_close txt_message chk_square edit_square ...
      chk_saveChanges
  persistent baseMaxImgVal state_saveChanges;
  ret = '';

  if( isempty(action) ), action=' ';, end;
  switch( action )

   case 'init'
    if( run_ROIlist('isopen'))
      figure( fh );
      ret = fh;
      return;
    end;

    fh = ui_ROIlist;
    placeFig( fh, 'center', 'left' );
    setupUI( fh, mfilename );
    ret = fh;

    if( ROIlistValid )
      newROIlist = ROIlist;          % newROIlist is working copy
    else
      newROIlist = '';
    end;
    newDataParams = dataParams;
    itemCount = length(newROIlist);
    if( strcmp(viewParams.maxImgVal, 'auto' ))
      viewParams.maxImgVal = getImgMax(img_background);
    end;

    % set up UI controls
    push_addROI = findobj( allchild(fh), 'Tag', 'push_addROI' );
    push_delROI = findobj( allchild(fh), 'Tag', 'push_delOI' );
```

```
 push_addEntire = findobj( allchild(fh), 'Tag', 'push_addEntire' );
 list_ROIlist = findobj( allchild(fh), 'Tag', 'list_ROIlist' );
 push_loadFile = findobj( allchild(fh), 'Tag', 'push_loadFile' );
 push_exe = findobj( allchild(fh), 'Tag', 'push_exe' );
 push_close = findobj( allchild(fh), 'Tag', 'push_close' );
 txt_message = findobj( allchild(fh), 'Tag', 'txt_message' );
 chk_square = findobj( allchild(fh), 'Tag', 'chk_square' );
 edit_square = findobj( allchild(fh), 'Tag', 'edit_square' );
 chk_saveChanges = findobj( allchild(fh), 'Tag', 'chk_saveChanges' );

 baseMaxImgVal = viewParams.maxImgVal;

 run_ROIlist( 'update' );

case 'isopen'
 if( isempty(fh) ) ret = 0;              % fh = ''
 elseif( ~ishandle(fh) ) ret = 0; % fh = invalid handle
 else
   ret = strcmp( get(fh, 'Name'), 'Regions of Interest' );
 end;

case 'raise'
 if( run_ROIlist('isopen') )
   figure(fh);
   ret = 1;
 else
   ret = 0;
 end;

case 'update'
 if( ~run_ROIlist('isopen') ) return; end;
 %if( ~sourceParamsValid )
 %  set( push_exe, 'Enable', 'Off' );
 %  for i = 1:length(newROIlist)
    %newROIlist(i).modified = 1;
   %end;
 %   set(txt_message, 'String', ['Warning: Source data has changed.' ...
    %           ' Existing ROIs may be invalid.'] );
 % else
 if( isempty( newROIlist ))
   set( push_exe, 'Enable', 'Off' );
   set(txt_message, 'String', 'Click ''Add'' to add regions of interest');
 else
   a = find([newROIlist.modified]);
   if( isempty(a))
     set( push_exe, 'Enable', 'Off' );
     set( txt_message, 'String', ['Map has been generated for all' ...
                 ' regions'] );
   else
     set( push_exe, 'Enable', 'On' );
     set(txt_message, 'String', ['Click ''Make map'' to perform' ...
                 ' calculations on the the new regions']);
   end; %if
 end; %if

 if( isempty(saveFile) )
   set( chk_saveChanges, 'Value', 1);
   set( chk_saveChanges, 'Enable', 'off' );
   state_saveChanges = 1;
 else
   if( isempty(state_saveChanges)) state_saveChanges = 1; end;
   set( chk_saveChanges, 'Value', state_saveChanges );
   set( chk_saveChanges, 'Enable', 'on' );
 end;

 %end; %if
 run_ROIlist( 'update_ROIlist' ); % this also updates the image
 if( nargin>1 )
   if( strcmp(varargin{1}, 'thisonly' ))
     return;
   else
     run_main( 'update' );
   end; %if
 end; %if

case 'update_ROIlist'
 set(fh,'Pointer','watch');
 if( isempty(newROIlist) )
   set( list_ROIlist, 'String', ' '  );
 else
   tempStr = '';
```

```
    for i = 1:length(newROIlist)
      b = length(newROIlist(i).maski);
      if( newROIlist(i).modified ), stat = '(new)';
      else, stat = '';
      end; %if
      tempStr = strvcat( tempStr, ...
                         ['ROI ' num2str(i) ': ' num2str(b) ' pixels ' ...
                         stat]);
    end; %for
    set( list_ROIlist, 'String', tempStr );
  end;
  showROIImage(newROIlist, list_ROIlist);
  set(fh,'Pointer','arrow');

case 'list_ROIlist'
  set(fh,'Pointer','watch');
  showROIImage(newROIlist, list_ROIlist);
  set(fh,'Pointer','arrow');

case 'warning' % This means ROIlist is potentially dirty due to
               % modification elsewhere.
  if( run_ROIlist('isopen') )
    newDataParams = dataParams;
    if( ~isempty( newROIlist ))
      for i = 1:length(newROIlist)
        newROIlist(i).modified = 1;
      end;
    end;
    run_ROIlist( 'update', 'thisonly' );
    set(txt_message, 'String', ['Warning: Source may have changed.' ...
                    ' Existing ROIs and image data may be invalid.'] );
  end;

case 'close'               % does not update anything (same as cancel)
  if( run_ROIlist('isopen') )
    delete( fh );
    disp( 'run_ROIlist closed' );
    run_main( 'update' );
  end;

case 'push_addROI'
  set(fh,'Pointer','watch');

  if( ~run_image('isopen') )
    showROIImage(newROIlist, list_ROIlist);
  else
    run_image('raise');
  end;

  if( get(chk_square,'Value') )
    sqSize = round(max(1, str2num(get(edit_square,'String')))/2);
    [x,y] = ginput(1);
    x = round(x); y = round(y);
    msize = newDataParams.msize;
    mask = zeros(msize);
    xrange = max(x-sqSize,1):min(x+sqSize-1,msize(2));
    yrange = max(y-sqSize,1):min(y+sqSize-1,msize(1));
    mask(yrange,xrange) = 1;
  else
    mask = roipoly;
  end;

  maski = find(mask(:));    % indices of non-zero's in mask
  %maskInfo = ['ROI ' num2str(itemCount) ' -- ' ...
  %          num2str(length(maski)) ' pixels'];
  if( ~isempty(maski) )
    newROIlist = [newROIlist ; ...
                  struct( 'maski', maski, ...
                          'modified', 1)];
    run_ROIlist( 'update' );
  end;
  figure( fh );
  set(fh,'Pointer','arrow');

case 'push_addEntire'
  set(fh,'Pointer','watch');
  mask = ones( newDataParams.msize );
  maski = find(mask(:));    % indices of non-zero's in mask
  newROIlist = [newROIlist ; ...
                struct( 'maski', maski, ...
                        'modified', 1)];
```

```
 run_ROIlist( 'update' );
 figure( fh );
 set(fh,'Pointer','arrow');

case 'push_delROI'
 if( isempty(newROIlist)), return;, end;
 a = get( list_ROIlist, 'Value' );
 b = size( newROIlist,1 );
 newROIlist = [newROIlist(1:a-1,:);
           newROIlist(a+1:b,:)];

 temp = max( size(newROIlist,1), 1 );            % lower bound
 set( list_ROIlist, 'Value', temp );
 run_ROIlist( 'update' );

case 'push_loadFile'
 [filen dirn] = uigetfile( 'result*.mat', ['Select file to load ROI' ...
                ' set from' ] );
 if( filen~= 0 )
   s = load( [dirn filen], 'ROIlist', 'dataParams' );
   if( ~isfield(s, 'ROIlist') | ~isfield(s, 'dataParams' ))
    % load didn't work
    errordlg( [filen ' does not contain a ROI set'], 'Error', 'modal' );
    return;
   end;
   if( ~prod(s.dataParams.msize==dataParams.msize ))
    errordlg( ['Matrix size in ' filen ' is incompatible with' ...
               ' current data'], 'Error' );
   else
    % loaded okay
    newDataParams = dataParams;
    % add the new ROI list now ..................................
    for i = 1:length(s.ROIlist)
      s.ROIlist(i).modified = 1;
    end;
    if( isempty(newROIlist) )
      newROIlist = s.ROIlist;
    else
      resp = questdlg( ['Do you want to replace or append to' ...
                        ' current ROI list?'], 'Question', ...
                       'Append', 'Replace', 'Append' );
      if( strcmp(resp, 'Replace' ))
        newROIlist = s.ROIlist;
      else
        newROIlist = [newROIlist; s.ROIlist];
      end;
    end;

    % (don't nullify existing T1/T2 map data because it may still be
    % useful)

    run_ROIlist( 'update' );
    itemCount = size( newROIlist, 1 );
   end
 end

case 'chk_square'
 if( get(chk_square,'Value'))
   set( edit_square, 'String', '16' );     % some default value
 else
   set( edit_square, 'String', '' );
 end;

case 'edit_square'
 if( ~get(chk_square,'Value'))
   set( chk_square,'Value',1 );
 end;
 val = str2num(get(edit_square,'String'));
 if( isempty(val) )
   set( edit_square,'String', '16' );     % some default value
 else
   if( val<=0 )
     set( edit_square,'String', '16' );   % some default value
   end;
 end;

case 'chk_saveChanges'
 state_saveChanges = get( chk_saveChanges, 'Value' );
 if( strcmp(get(chk_saveChanges,'Enable'),'on'))
   if( state_saveChanges==1 )
     set( txt_message, ...
```

```
        'String', strvcat('Changes will be saved to file', saveFile ));
  else
    set( txt_message, 'String', 'Changes will not be saved to file' );
  end;
end;

case 'push_exe'
 % reset results data
 if( isempty(theMapSave) | ~theMapSaveValid)
   theMapSave = zeros(newDataParams.msize);
   theMapi = '';
   pfittedSave = '';
 end;

 if( isempty(newROIlist)), return;, end;
 if( ~figsValid )
   a = feval( ['load_' sourceParams.magnet] );
   if( a==0 )
     errordlg( 'Unable to open source files', 'Error', 'modal' );
     return;
   end;
   figsValid = 1;
 end;
 disp( 'execute now' );
 if( isempty(saveFile) )          % saveFile is a global variable
   saveFile = getSaveFile;
   state_saveChanges = 1;
 end;

 dataParams = newDataParams;
 ROIlist = newROIlist;            % commit newROIlist to actual ROIlist

 whichOnes = find([newROIlist.modified]); % find the new ROIs
 if( isempty(whichOnes) ), return;, end;

 maski = unionall(ROIlist(whichOnes).maski);     % make union of the masks
 mask = zeros(size(img_background));
 mask(maski) = 1;

 removei = intersect(maski,theMapi);             % remove from mask the pixels
 mask(removei) = 0;                       %  already calculated

 % run calculations now ------------------------------------------
 [newMap, newMapi, newPfitted] = ...
     feval( ['exe_' sourceParams.magnet], mask, dataParams );

 if( ~isempty(newMapi) )
   % filter out-of-range pixels
   [i j v] = find(newMap(:));             %  consider non-zeros only
   if( ~isempty(v) )
     med = median( v );
     i = find( newMap(:)>100*med );
     newMap(i) = 0;
   end;

   % combine new ROIs with existing T1/T2-mapped ROIs
   theMapi = [theMapi ; newMapi];         % new theMapi -- indices
   pfittedSave = [pfittedSave ; newPfitted];
   theMapSave(newMapi) = newMap(newMapi); % 64x64 (e.g.) grid
   if( strcmp(viewParams.maxTval, 'auto') )
     viewParams.maxTval = getMapMax(theMapSave);
   end;

 end; %if ~isempty(newMapi)

 for i = 1:length(whichOnes)
   ROIlist(whichOnes(i)).modified = 0;           % unset modified flag
 end;
 if( state_saveChanges )
   disp( [' Saving to ' saveFile] );
   save( saveFile, 'sourceParams', 'dataParams', 'viewParams', ...
         'ROIlist', 'theMapSave', 'theMapi', 'pfittedSave', ...
         'img_background' );
 end;

 % save figs if needed (i.e. figs data has been aligned)
 if( doSaveFigs )
   [pathn,name,ext] = fileparts(saveFile);
   figsFile = [pathn '/' name '-figs.mat'];
   disp( [' Saving figs to ' figsFile] );
   save( figsFile, 'figs' );
```

```
      doSaveFigs = 0;
    end;

    ROIlistValid = 2;         % valid & modified flag
    theMapSaveValid = 2;              % valid & modified flag
    run_ROIlist( 'close' );
    run_viewResults( 'init' );
    run_main( 'update' );

   case 'push_close'
    % Pushing the close button updates things.
    dataParams = newDataParams;
    ROIlist = newROIlist;             % commit newROIlist to actual ROIlist
    if( ~isempty(ROIlist) )
      ROIlistValid = 2;               % valid & modified flag
      if( ~isempty(saveFile) & state_saveChanges )
        disp( [ ' Saving to ' saveFile] );
        save( saveFile, 'sourceParams', 'dataParams', 'viewParams', ...
              'ROIlist', 'theMapSave', 'theMapi', 'pfittedSave', ...
              'img_background' );
      end;
    else
      ROIlistValid = 0;
    end;
    run_ROIlist( 'close' );
    run_viewResults( 'warning' );
    run_main( 'update' );

   case 'push_cancel'
    run_ROIlist( 'close' );

   otherwise
    % ignore unrecognized (unrecognised) action

  end; %switch


function filen = getSaveFile

  filen = 0;
  while( filen==0 )
    [filen dirn] = uiputfile( 'results*.mat', 'Save Calculation Results To' );
    if( filen==0 )
      h = errordlg( 'A save file must be specified', 'Error', 'modal' );
      waitfor( h );
    else
      filen = [dirn filen];
    end
  end



function showROIImage(newROIlist, list_ROIlist)
  include_globals;

  if( ~isempty( newROIlist ))
    unionMaski = unionall( newROIlist(:).maski, [] );
    unionMask = zeros(dataParams.msize);
    unionMask(unionMaski) = img_background(unionMaski);

    % highlight the selected ROI
    selectedROI = get( list_ROIlist, 'Value' );
    if( ~isempty(selectedROI))
      unionMaski = unionall( newROIlist(selectedROI).maski, [] );
      unionMask(unionMaski) = img_background(unionMaski)*3;
    end;
  else
    unionMask = '';
  end;
  run_image( 'display', ...
             struct( 'imgmax', viewParams.maxImgVal, ...
                     'colormap', 'green' ), ...
             img_background, unionMask );
function ret = run_selectData( action, varargin )
%RUN_SELECTDATA

% Bruce Po - igan@mit.edu

  include_globals;
  %global isDone itemCount;
  persistent fh itemCount newDataParams newViewParams dataParamsModified ...
```

```
    viewParamsModified figsModified;
persistent sld_brightness push_autoBrightness txt_brightness ...
    edit_dataSequence push_viewPoints push_viewImg push_align ...
    push_close push_cancel;
persistent sliderbase_imgVal

global newFigs

ret = '';
if( isempty(action) ), action=' ';, end;
switch( action )

 case 'init'
  if( run_selectData('isopen') )
    ret = run_selectData('raise');
    return;
  end;

  fh = ui_selectData;
  placeFig( fh, 'center', 'left' );
  setupUI( fh, mfilename );
  ret = fh;

  newDataParams = dataParams;
  newViewParams = viewParams;
  newFigs = '';
  dataParamsModified = 0;
  viewParamsModified = 0;
  figsModified = 0;

  % set up UI controls
  push_viewImg = findobj( allchild(fh), 'Tag', 'push_viewImg' );
  push_align = findobj( allchild(fh), 'Tag', 'push_align' );
  push_viewPoints = findobj( allchild(fh), 'Tag', 'push_viewPoints' );
  sld_brightness = findobj( allchild(fh), 'Tag', 'sld_brightness' );
  edit_dataSequence = findobj( allchild(fh), 'Tag', 'edit_dataSequence' );
  push_autoBrightness=findobj(allchild(fh),'Tag','push_autoBrightness');
  txt_brightness = findobj( allchild(fh), 'Tag', 'txt_brightness' );
  push_close = findobj( allchild(fh), 'Tag', 'push_close' );
  push_cancel = findobj( allchild(fh), 'Tag', 'push_cancel' );

  % image background value
  if( strcmp(newViewParams.maxImgVal, 'auto' ))
    newViewParams.maxImgVal = getImgMax(img_background);
  end;
  sliderbase_imgVal = newViewParams.maxImgVal;

  %set( edit_dataSequence, 'String', num2str(newDataParams.data_sequence) );
  % this will be set in 'update' method
  set( sld_brightness, 'Value', 0.5 );
  set( txt_brightness, 'String', num2str(newViewParams.maxImgVal,6) );

  run_selectData( 'update' );

 case 'isopen'
  if( isempty(fh) ) ret = 0;               % fh = ''
  elseif( ~ishandle(fh) ) ret = 0; % fh = invalid handle
  else
    ret = strcmp( get(fh, 'Name'), 'Select Data' );
  end;

 case 'raise'
  if( run_selectData('isopen') )
    figure(fh);
    ret = 1;
  else
    ret = 0;
  end;


 case 'update'
  if( ~run_selectData('isopen')) return; end;
  set( edit_dataSequence, 'String', num2str(newDataParams.data_sequence) );

 case 'warning' % This means data was modified, potentially
                % nullifying any data being worked on in this
                % window.  So cancel.

  if( run_selectData('isopen') )
    run_selectData( 'push_cancel' );
  end;
```

```
case 'edit_dataSequence'   % this affects all the calculated results
 if( theMapSaveValid )
   a = questdlg( ['Modifying this value will nullify existing results!' ...
                  '  Do you wish to continue?'], 'Warning', 'Yes', 'No', ...
                 'No' );
   if( strcmp(a,'No'))
     run_selectData( 'update' );
     return;
   end;
 end;

 a = str2num(get( edit_dataSequence, 'String' ));
 if( ~sum(a<=0) & ~sum(a>newDataParams.no_of_timeVar) )
   newDataParams.data_sequence = round(a);
   dataParamsModified = 1;
 end;

 run_selectData( 'update' );

case 'push_setBackground'
 imageList = 1:newDataParams.no_of_timeVar;
 imageList = cellstr(num2str(imageList'));
 [imgNum,ok] = listdlg( 'PromptString', ...
                        'Select image no.:', ...
                        'Name', 'Set background image', ...
                        'SelectionMode', 'single', ...
                        'ListString', imageList );
 if( ok )
   viewParamsModified = 1;
   if( ~figsValid )        % first make sure data is loaded
     a = feval( ['load_' sourceParams.magnet] );
     if( a==0 )
       errordlg( 'Could not load source files', ...
                 'Unable to complete command', 'modal' );
       return;
     end;
     figsValid = 1;
   end;
   if( ndims(figs)==3 )    % now set the new background
     img_background = figs(:,:,imgNum);
   else
     img_background = reshape(figs(:,imgNum),newDataParams.msize);
   end;

   run_selectData( 'push_autoBrightness' );
 end;

case 'push_viewPoints'
 if( ~sourceParamsValid ) return; end;
 no_images = newDataParams.no_of_timeVar;
 msize = newDataParams.msize;
 run_image('display', struct('imgmax',newViewParams.maxImgVal), ...
           img_background );


 h = dlgInfo( 'Select a pixel from the image', '', [200 40 10] );
 figure( fh_image );
 [x,y] = ginput(1);% user selects pixel from image now
 close( h );
 x = round(x); y = round(y);
 if( x<1 | y<1 | x>msize(2) | y>msize(1) )
   return;
 end;

 % Viewing data points requires source data to be loaded
 if( ~figsValid )
   a = feval( ['load_' sourceParams.magnet] );
   if( a==0 )
     errordlg( 'Could not load source files', ...
               'Unable to complete command', 'modal' );
     return;
   end;
   figsValid = 1;
 end;

 % sample pixel from each image
 %dataPoints = zeros(no_images,1);
 clear dataPoints;
 %for(i = 1:no_images)
 a = 1;
```

```
 for i = newDataParams.data_sequence
   if( ndims(figs)==2 )
    A = reshape( figs(:,i), msize );
   else
    A = figs(:,:,i);
   end;
   dataPoints(a) = A(y,x);
   a = a+1;
 end;
 figure;
 t = gettimeaxis;
 plot( t(newDataParams.data_sequence), dataPoints, '.' );
 xlabel( ['Coordinates: (' num2str(x) ',' num2str(y) ')'] );

case 'push_viewImg'        % user wants to view image sequence
 if( ~sourceParamsValid ), return;, end;

 % if the source images are not loaded already, then load them now
 if( ~figsValid )
   a = feval( ['load_' sourceParams.magnet] );
   if( a==0 )
    errordlg('Could not load source files', 'Unable to complete command', ...
             'modal');
    return;
   end;
   figsValid = 1;
 end;

 run_viewFigs( 'init', newDataParams );

case 'push_align'
 ds = newDataParams.data_sequence;
 todo = inputdlg( strvcat(['Note: Alignment should be done only' ...
                ' if images appear crooked.'], 'Images to align:'), ...
               'Input', 1, {num2str(ds)} );
 todo = str2num(char(todo));
 if(isempty(todo)) return; end;
 if(length(todo)==1 | sum(todo<=0)) return; end;
 run_viewFigs( 'close' );

 % memory hungry part!
 if( ~figsValid )          % load figs if not already loaded
   a = feval( ['load_' sourceParams.magnet] );
   if( a==0 )
    errordlg( 'Could not load source files', ...
              'Unable to complete command', 'modal' );
    return;
   end;
   figsValid = 1;
 end;
 newFigs = figs;

 % align 2nd image (using 1st image as base image)
 h = waitbar( 0, 'Aligning images...' );
 waitbar(0.01,h);
 newFigs(:,:,todo(2)) = alignData( figs(:,:,todo(1)), figs(:,:,todo(2)));

 % align the rest (using 1st image as base image)
 for i=3:length(todo)
   if( ~ishandle(h) )
    h = waitbar( 0, 'Aligning images...' );
    waitbar(0.01,h);
   end;
   waitbar((i-2)/(length(todo)-1),h);
   newFigs(:,:,todo(i)) = alignData( [], figs(:,:,todo(i)), 'reuse');
 end;
 clear alignData;
 close(h);

 figsModified = 1;

case 'sld_brightness'
 if( ~sourceParamsValid ), return;, end;
 viewParamsModified = 1;
 a = get( sld_brightness, 'Value' );
 newViewParams.maxImgVal = 2*a*sliderbase_imgVal;
 set( txt_brightness, 'String', num2str(newViewParams.maxImgVal,6) );

 run_image('display', struct('imgmax',newViewParams.maxImgVal), ...
           img_background );
```

```
    case 'push_autoBrightness'
     if( ~sourceParamsValid ), return;, end;
     viewParamsModified = 1;
     set( sld_brightness, 'Value', 0.5 );
     newViewParams.maxImgVal = getImgMax(img_background);
     set( txt_brightness, 'String', num2str(newViewParams.maxImgVal,6) );
     run_image('display', struct('imgmax',newViewParams.maxImgVal), ...
               img_background );

    case 'push_close'                  % save changes that were made
     if( run_selectData('isopen') )

       if( dataParamsModified )
         dataParams = newDataParams;
         theMapSaveValid = 0;
         for i = 1:length(ROIlist)
           ROIlist(i).modified = 1;
         end;
       end;

       if( viewParamsModified )
         viewParams = newViewParams;
         run_viewResults( 'update' );
       end;

       if( figsModified )
         figs = newFigs;
         doSaveFigs = 1;
       end;

       if( (dataParamsModified|viewParamsModified) & ~isempty(saveFile) )
         disp( [ 'Saving to ' saveFile] );
         save( saveFile, 'viewParams', 'dataParams', 'img_background', ...
               '-append' );
       end;
       run_selectData( 'close' );
       run_main( 'update' );
     end;

    case 'push_cancel'
     run_selectData( 'close' );

    case 'close'
     run_viewFigs( 'close' );
     clear newFigs;
     if( run_selectData('isopen'))
       if( viewParamsModified )              % reset display if needed
         run_image('display', struct('imgmax',viewParams.maxImgVal), ...
                   img_background );
       end;

       delete( fh );
       disp( 'run_selectData closed' );
     end;

    otherwise
     % ignore unrecognized (unrecognised?) action

   end; %switch


% function showImage
%   include_globals;

%   saveAxis = '';

%   % display the picture now
%   if( run_image('isopen') )
%     figure( fh_image );
%     saveAxis = axis;
%     iptsetpref( 'ImshowTrueSize', 'manual' );
%   else
%     fh_image = newImageFig;
%   end;

%   overlaymap( img_background, '', viewParams.maxImgVal );

%   % restore zoom (if possible)
%   if( ~isempty( saveAxis ) )
%     zoom( 1 );
```

```
%     axis( saveAxis );
%   end;

function ret = run_viewFigs( action, varargin )
%RUN_VIEWFIGS Views the figs array as a movie.
%
%   Creates a movie sequence out of the MRI data stored in the global
%   variable figs.  However, if the global variable newFigs has data in
%   it, then use that instead.

  global figs newFigs viewParams img_background
  persistent fh ah_mov mov sld_mov push_play dataParams rad_old ...
      rad_new push_undo
  ret = '';

  switch( action )
   case 'init'

    if( feval(mfilename,'isopen') ) return; end;
    dataParams = varargin{1};

    fh = figure( 'Name', 'Image Sequence', ...
                 'NumberTitle', 'off' );
    ret = fh;

    if( isempty(mov) ) run_viewFigs( 'makeMovie' ); end;


    % set up the rest of the movie window
    num_frames = length(dataParams.data_sequence);
    title( ['Data ' num2str(dataParams.data_sequence(1))] );
    ah_mov = gca;
    sld_mov = uicontrol( 'Parent', fh, ...
                        'Position',[10 10 100 15], ...
                        'Style','slider', ...
                        'Tag','sld_mov', ...
                        'Max', num_frames, ...
                        'Min', 1, ...
                        'Value', 1, ...
                        'SliderStep', [1/(num_frames-1) 4/(num_frames-1)], ...
                        'Callback', 'run_viewFigs(''frame'');' );

    push_play = uicontrol( 'Parent', fh, ...
                          'Position', [110 10 40 15], ...
                          'String', 'Play', ...
                          'Callback', 'run_viewFigs(''play'');' );

    if( ~isempty(newFigs) )
      push_undo = uicontrol( 'Parent', fh, ...
                            'Position', [ 170 10 100 15], ...
                            'String', 'Undo align', ...
                            'Callback', 'run_viewFigs( ''undoAlign'');');
    end;


    %run_viewFigs( 'frame' );

   case 'isopen'
    if( isempty(fh) ) ret = 0;                % if h = ''
    elseif( ~ishandle(fh) ) ret = 0; % if h is invalid handle
    else
      ret = (strcmp( get(fh, 'Name'), 'Image Sequence' ));
    end;

   case 'makeMovie'

    imgMax = viewParams.maxImgVal;
    if(strcmp(imgMax,'auto'))
      imgMax = getImgMax(img_background);
    end;
    if( isempty(newFigs))
      X(:,:,1,:) = uint8(figs(:,:,dataParams.data_sequence) / ...
                         imgMax*256);
    else
      X(:,:,1,:) = uint8(newFigs(:,:,dataParams.data_sequence) / ...
                         imgMax*256);
    end;
    set( fh, 'DeleteFcn', 'run_viewFigs(''close'');' );
    figure( fh );
    mov = immovie(X,gray(256));
    clear X;
```

```
    ret = 1;

   case 'play'
    set( sld_mov, 'Enable', 'off' );
    set( push_play, 'Enable', 'off' );
    title( '' );
    movie(ah_mov,mov,1,6);
    set( sld_mov, 'Enable', 'on' );
    set( push_play, 'Enable', 'on' );
    run_viewFigs( 'frame' );
    ret = 1;

   case 'frame'
    frame_no = round(get(sld_mov,'Value'));
    set(sld_mov,'Value',frame_no);
    movie(ah_mov,mov,[0 frame_no]);
    title( ['Data ' num2str(dataParams.data_sequence(frame_no))] );
    ret = 1;

   case 'undoAlign'
    frame_no = round(get(sld_mov,'Value'));
    resp = questdlg( 'Undo align of current image?', 'Confirm', ...
                     'Yes', 'No', 'Yes' );
    ret = 0;
    if( strcmp(resp,'Yes') )
      i = dataParams.data_sequence(frame_no);
      newFigs(:,:,i) = figs(:,:,i);
      %run_viewFigs( 'makeMovie' );
      msgbox( ['Image Sequence display will not be updated until' ...
               ' it is closed and reopened'], 'Note', 'modal' )
      ret = 1;
    end;

   case 'close'
    if( feval(mfilename,'isopen' ))
      delete( fh );
    end;
    clear run_viewFigs;
  end;

function ret = run_viewResults( action, varargin )
%RUN_VIEWRESULTS  View results of calculation.

% Bruce Po - igan@mit.edu

  include_globals;
  persistent list_ROIlist chk_selectAll ...
      push_ROIs push_saveImg push_prompt ...
      edit_bgMax edit_tMax push_figTitle ...
      txt_eqn chk_colorbar chk_labels chk_ROIdetails...
      push_analyzeParams push_analyzeCurve
  persistent fh txt_figROI
  global tempMapi tempMap thisROIlist textLabels ...
      parameterType;

  % tempMap - same size as MRI image, contains calculated
  %     T1 or T2 map for the ROI(s) selected, for displaying

  ret = '';

  if( isempty(action) ), action=' ';, end;
  switch( action )
   case 'init'
    if( run_viewResults('isopen'))
      figure( fh );
      ret = fh;
      run_viewResults( 'updateFig', 'preserveZoom' );
      return;
    end;
    fh = ui_viewResults;      % start the UI
    setupUI( fh, mfilename );
    set( fh, 'Name', 'Calculation Results' );
    placeFig( fh, 'center', 'left' );
    ret = fh;


    % only ROIs with maps generated already matter to this function
    thisROIlist = ROIlist(find([ROIlist.modified]==0));

    % set up UI controls
    list_ROIlist = findobj( allchild(fh), 'Tag', 'list_ROIlist' );
```

```
chk_selectAll = findobj( allchild(fh), 'Tag', 'chk_selectAll' );
push_ROIs = findobj( allchild(fh), 'Tag', 'push_ROIs' );
edit_bgMax = findobj( allchild(fh), 'Tag', 'edit_bgMax' );
edit_tMax = findobj( allchild(fh), 'Tag', 'edit_tMax' );
push_figTitle = findobj( allchild(fh), 'Tag', 'push_figTitle' );
chk_colorbar = findobj( allchild(fh), 'Tag', 'chk_colorbar' );
chk_labels = findobj( allchild(fh), 'Tag', 'chk_labels' );
chk_ROIdetails = findobj( allchild(fh), 'Tag', 'chk_ROIdetails' );
push_saveImg = findobj( allchild(fh), 'Tag', 'push_saveImg' );
push_prompt = findobj( allchild(fh), 'Tag', 'push_prompt' );
txt_eqn = findobj( allchild(fh), 'Tag', 'txt_eqn' );
push_analyzeCurve = findobj( allchild(fh), 'Tag', 'push_analyzeCurve' );


% figure out T1 or T2 ..........................................
a = length( sourceParams.magnet );
tag = sourceParams.magnet(a-1:a);
switch( tag )
 case { 'IR', 'SR' }
  parameterType = 'T1';
 case 'T2'
  parameterType = 'T2';
 otherwise
  parameterType = 'Paramater';
end;
set(findobj(allchild(fh), 'Tag', 'txt_mapSettingsBox'), 'String', ...
                  [parameterType ' Map Display'] );

% set up ROI list box ..........................................
run_viewResults( 'setup_list_ROIlist' );

run_viewResults( 'update' );
run_viewResults( 'updateFig', 'redraw' );

case 'isopen'
 if( isempty(fh) ) ret = 0;              % fh = ''
 elseif( ~ishandle(fh) ) ret = 0; % fh = invalid handle
 else
   ret = (strcmp( get(fh, 'Name'), 'Calculation Results' ));
 end;

case 'raise'
 if( run_viewResults('isopen') )
   figure(fh);
   ret = 1;
 else
   ret = 0;
 end;

case 'update'              % does not re-display the figure
 if( ~run_viewResults('isopen')) return; end;

 eqn = feval( ['def_', sourceParams.magnet], 'showEqn');
 set( txt_eqn, 'String', eqn );

 set( chk_colorbar, 'Value', viewParams.figColorbar );
 set( chk_labels, 'Value', viewParams.figROIlabels );
 set( chk_ROIdetails, 'Value', viewParams.ROIdetails );

 if( strcmp( viewParams.maxImgVal, 'auto' ))
   viewParams.maxImgVal = getImgMax( img_background );
 end;
 if( strcmp( viewParams.maxTval, 'auto' ))
   viewParams.maxTval = getMapMax( tempMap );
 end;
 set( edit_bgMax, 'String', num2str(viewParams.maxImgVal) );
 set( edit_tMax, 'String', num2str(viewParams.maxTval) );
 %set( edit_tMin, 'String', num2str(viewParams.minTval) );

 selectedROI = get( list_ROIlist, 'Value' );

 tempMapi = unionall( thisROIlist(selectedROI).maski, [] );

 tempMap = theMapSave(:);
 tempMap(setdiff(theMapi,tempMapi)) = 0; % unselected ROIs' data
 tempMap = reshape( tempMap, dataParams.msize );

 run_analyzeParams( 'update' );

case 'warning'      % another process updated ROIlist while this window
                    %  is still open
```

```
 if( ~run_viewResults('isopen')) return; end;

 % use only the ROIs that have a T1/T2 map generated already
 if( ~isempty(ROIlist) )
   thisROIlist = ROIlist(find([ROIlist.modified]==0));
 else
   thisROIlist = '';
 end;

 % re-setup ROI list box
 run_viewResults( 'setup_list_ROIlist' );

 run_viewResults( 'update' );
 run_viewResults( 'updateFig' );

case 'close'
 run_analyzeCurve( 'close' );
 run_analyzeParams( 'close' );
 if( run_viewResults('isopen'))
   delete( fh );
   disp( 'run_viewResults closed' );      % for debugging
 end;

case 'updateFig'
 isNewFig = bringUpTheImage;

 % optional: save the current zoom settings ......................
 prevXLim = ''; prevYLim = '';
 if( nargin>1 )
   if( strmatch('preserveZoom', varargin) & ~isNewFig )
     prevXLim = xlim; %get( ah, 'XLim' );
     prevYLim = ylim; %get( ah, 'YLim' );
   end;
 end;

 % optional: redraw the figure window ............................
 redraw = 0;
 if( nargin>2 )
   if( strmatch( 'redraw', varargin ))
     redraw = 1;
     run_image('close');
     run_image('init');
   end;
 end;

 % ROI details text ..............................................
 if( get( chk_ROIdetails, 'Value' ) )
   needNew = 0;
   if( isempty(txt_figROI) )
     needNew = 1;
   else
     needNew = ~ishandle(txt_figROI);
   end;
   if( needNew )
     axis off;
     txt_figROI = text( 0, 0, '' );
   end;
   i = get(list_ROIlist,'Value');
   if( ~isempty(i) )
     temp = get(list_ROIlist,'String');
     set( txt_figROI, 'String', temp(i,:) );
   end;
   if( redraw )
     axes( 'position', [.1, .25, .8, .65]);
   end;
 end;

 % display the image .............................................
 run_image('display', struct('imgmax',viewParams.maxImgVal, ...
                             'mapmax',viewParams.maxTval), ...
           img_background, tempMap );

 % apply colorbar ................................................
 if( get( chk_colorbar, 'Value' )), overlaycolorbar;, end;

 % ROI labels on the figure ......................................
 selectedROIs = get( list_ROIlist, 'Value' );
 textLabels = zeros(length(thisROIlist),1);
 for n = selectedROIs
   i = ROIlist(n).maski(1);
   [y,x] = getyx(i);
```

```
  textLabels(n) = text('String', ['ROI ' num2str(n)],'Position', ...
                       [x,y], 'Color', 'white', 'FontWeight', 'bold');
end;
if( get(chk_labels, 'Value')==1 )
  set( textLabels,'Visible','on' );
else
  set( textLabels,'Visible','off' );
end;

% figure title and labels ....................................
title( viewParams.figTitle );
xlabel( viewParams.figxlabel );
ylabel( viewParams.figylabel );

% optional: restore the zoom level ..........................
if( nargin>1 )
  %zoom reset;
  if( strmatch( 'preserveZoom', varargin) & ~isNewFig )
    run_image('raise');
    zoom(1);
    xlim( prevXLim );
    ylim( prevYLim );

    %set( ah, 'XLim', prevXLim );
    %set( ah, 'YLim', prevYLim );
    %axes( ah );
  end;
end;

case 'setup_list_ROIlist'          % method
 tempStr = '';
 theMapLin = theMapSave(:);
 for i = 1:length(thisROIlist)
   %b = length(thisROIlist(i).maski);
   map = theMapLin(thisROIlist(i).maski);
   [a b map] = find(map);  % non-zeros
   tempStr = strvcat( ...
       tempStr, ...
       ['ROI ' num2str(i) ' -- ' num2str(length(map)) ' pixels' ...
        '    mean: ' num2str(mean(map)) '   std dev: ' num2str(std(map))] );
 end
 set( list_ROIlist, 'String', tempStr );
 set( list_ROIlist, 'Value', 1:length(thisROIlist) ); % select all ROIs
 set( chk_selectAll, 'Value', 1 );

case 'getSelectedROIs'             % method .....................
 ret = get( list_ROIlist, 'Value' );

case 'getROIString'                % method .....................
 temp= get( list_ROIlist, 'String' );
 ret = temp(get(list_ROIlist, 'Value'),:);

case 'list_ROIlist'
 selected = get(list_ROIlist,'Value');
 if( length(selected)==length(ROIlist) )
   set( chk_selectAll, 'Value', 1 );
 else
   set( chk_selectAll, 'Value', 0 );
 end;
 run_viewResults( 'update' );
 run_viewResults( 'updateFig', 'preserveZoom' );

case 'chk_selectAll'
 if( get(chk_selectAll,'Value')==1 )
   numROIs = length(thisROIlist);
   set( list_ROIlist, 'Value', 1:numROIs );      % select all ROIs
   run_viewResults( 'update' );
   run_viewResults( 'updateFig' );
 end;

case 'push_ROIs'           % call run_ROIlist
 prevXLim = ''; prevYLim = '';           % save the zoom level
 if( run_image('isopen') )               %  if needed
   run_image('raise');
   %ah = gca;
   prevXLim = xlim; %get( ah, 'XLim' );
   prevYLim = ylim; %get( ah, 'YLim' );
   %prevXLim = get( ah, 'XLim' );
   %prevYLim = get( ah, 'YLim' );
 end;
 run_ROIlist( 'init' );
```

```
 if( ~isempty( prevXLim ))              % restore the zoom level
   run_image( 'raise' );
   zoom(1);
   xlim( prevXLim );
   ylim( prevYLim );
   %set( ah, 'XLim', prevXLim );
   %set( ah, 'YLim', prevYLim );
   %axes( ah );
 end;

case 'edit_bgMax'
 a = get( edit_bgMax, 'String' );
 if( strcmp(a,'auto'))
   viewParams.maxImgVal = getImgMax( img_background );
 elseif( str2num(a)>0 )
   viewParams.maxImgVal = str2num(a);
 end;
 set( edit_bgMax, 'String', num2str(viewParams.maxImgVal) );
 run_viewResults( 'updateFig', 'preserveZoom' );

case 'edit_tMax'
 a = get( edit_tMax, 'String' );
 if( strcmp(a,'auto') )
   viewParams.maxTval = getMapMax( tempMap );
 elseif( str2num(a)>0 )
   viewParams.maxTval = str2num(a);
 end;
 set( edit_tMax, 'String', num2str(viewParams.maxTval) );
 run_viewResults( 'updateFig', 'preserveZoom' );

case 'push_figTitle'
 a = inputdlg( {'Figure title:', 'x label:', 'y label:'}, ...
               'Input', 1, {viewParams.figTitle, viewParams.figxlabel, ...
               viewParams.figylabel} );
 if( ~isempty(a) )
   viewParams.figTitle = a{1};
   viewParams.figxlabel = a{2};
   viewParams.figylabel = a{3};
   isNewImg = bringUpTheImage;

   % apply title
   %if( ~isNewImg ) title( a ); end;
   title( viewParams.figTitle );
   xlabel( viewParams.figxlabel );
   ylabel( viewParams.figylabel );
 end;

case 'chk_colorbar'
 viewParams.figColorbar = get( chk_colorbar, 'Value' );
 run_viewResults( 'updateFig', 'preserveZoom' );

case 'chk_ROIdetails'
 viewParams.ROIdetails = get( chk_ROIdetails, 'Value' );
 run_viewResults( 'updateFig', 'preserveZoom', 'redraw' );
 % adding ROI details text to bottom of figure requires
 %  redrawing the figure

case 'chk_labels'
 viewParams.figROIlabels = get( chk_labels, 'Value' );
 try
   if( get(chk_labels, 'Value') )
    set( textLabels,'Visible','on' );
   else
    set( textLabels,'Visible','off' );
   end;
 catch
   run_viewResults( 'updateFig' );
 end;

case 'push_saveImg'
 h = run_image( 'raise' );
 exportFig( h );

case 'push_prompt'
 disp( 'Type ''return'' to end the command line session.' );
 disp( '  The variables of interest are:' );
 disp( '  sourceParams, dataParams, theMapSave, pfittedSave' );
 keyboard

case 'push_analyzeParams'
 run_analyzeParams( 'init' );
```

```
   case 'push_analyzeCurve'
    run_analyzeCurve( 'init' );

   case 'push_close'          % save view parameter changes first
    set( fh, 'Pointer', 'watch' );
    try
      save( saveFile, 'viewParams', '-append' );
    end;
    set( fh, 'Pointer', 'arrow' );
    run_viewResults( 'close' );

  end %switch


% **** perhaps remove this function altogether once zoom is working
function ret = bringUpTheImage
  % returns 1 if new figure window
% returns 0 if figure already exists

  % display the picture now
  if( run_image('isopen'))
    run_image( 'raise' );
    ret = 0;
  else
    run_image( 'init' );
    ret = 1;
  end;
```

A.1.2 User Interface Definition Files

```
function fig = ui_analyzeCurve()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.


h0 = figure('Units','points', ...
        'Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        ...
        'FileName','E:\bruce\work3\src\ui_analyzeCurve.m', ...
        'MenuBar','none', ...
        'Name','Analyze Curve Fit', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[527.25 378 336.75 165], ...
        'Tag','Fig3', ...
        'Visible','on');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 135 67.5 22.5], ...
        'String','Plot curve fit:', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[78.75 135 135 22.5], ...
        'String','of pixel selected from image', ...
        'Style','radiobutton', ...
```

```
        'Tag','rad_pixelSelect');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[78.75 112.5 135 22.5], ...
        'String','of pixel at location', ...
        'Style','radiobutton', ...
        'Tag','rad_pixelCoordinates');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[78.75 90 123.75 22.5], ...
        'String','of selected ROI(s)', ...
        'Style','radiobutton', ...
        'Tag','rad_ROI');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[236.25 112.5 33.75 22.5], ...
        'Style','edit', ...
        'Tag','edit_x');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[292.5 112.5 33.75 22.5], ...
        'Style','edit', ...
        'Tag','edit_y');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 67.5 168.75 22.5], ...
        'String','log scale', ...
        'Style','checkbox', ...
        'Tag','chk_logScale');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 45 168.75 22.5], ...
        'String','show data range of selected ROI(s)', ...
        'Style','checkbox', ...
        'Tag','chk_showRange');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[236.25 45 90 22.5], ...
        'String','Get T1/T2 value', ...
        'Tag','push_getVal');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[236.25 11.25 90 22.5], ...
        'String','Plot', ...
        'Tag','push_plot');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 11.25 90 22.5], ...
        'String','Close', ...
        'Tag','push_close');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[270 112.5 22.5 22.5], ...
        'String','y:', ...
        'Style','text', ...
        'Tag','StaticText7');
h1 = uicontrol('Parent',h0, ...
```

```
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[213.75 112.5 22.5 22.5], ...
        'String','x:', ...
        'Style','text', ...
        'Tag','StaticText7');
if nargout > 0, fig = h0; end
function fig = ui_analyzeParams()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.


h0 = figure('Units','points', ...
        'Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        ...
        'FileName','e:\bruce\work3\src\ui_analyzeParams.m', ...
        'MenuBar','none', ...
        'Name','Analyze Parameters', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[252.75 364.5 291 163.5], ...
        'Tag','Fig2', ...
        'Visible','on');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Min',1, ...
        'Position',[123.75 123.75 101.25 22.5], ...
        'String',' ', ...
        'Style','popupmenu', ...
        'Tag','pop_fieldNo', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[191.25 45 90 22.5], ...
        'String','Save to file...', ...
        'Tag','push_saveStats');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 127.5 101.25 22.5], ...
        'String','Parameter to analyze:', ...
        'Style','text', ...
        'Tag','StaticText5');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 56.25 90 22.5], ...
        'String','Display map', ...
        'Tag','push_displayMap');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 78.75 90 22.5], ...
        'String','Histogram', ...
        'Tag','push_plotHist');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
```

```
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 101.25 90 22.5], ...
        'String','Plot values', ...
        'Tag','push_plotField');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Max',2, ...
        'Position',[123.75 67.5 157.5 56.25], ...
        'Style','edit', ...
        'Tag','edit_fieldInfo');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[191.25 11.25 90 22.5], ...
        'String','Close', ...
        'Tag','push_close');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 11.25 90 22.5], ...
        'String','Plot profile', ...
        'Tag','push_plotProfile');
if nargout > 0, fig = h0; end
function fig = ui_dataSource()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.


h0 = figure('Units','points', ...
        'Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        ...
        'FileName','e:/bruce/work3/src/ui_dataSource.m', ...
        'MenuBar','none', ...
        'Name','Data Source', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[42.75 102.75 282 368.25], ...
        'Tag','Fig3', ...
        'Visible','on');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[5.25 36.75 270 298.5], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 303.75 101.25 22.5], ...
        'String','Magnet type:', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Min',1, ...
        'Position',[112.5 303.75 157.5 22.5], ...
```

```
        'String',' ', ...
        'Style','popupmenu', ...
        'Tag','pop_magnets', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 213.75 258.75 22.5], ...
        'String','Click parameter below to edit', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ForegroundColor',[0 0 0.627450980392157], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 45 258.75 45], ...
        'Style','text', ...
        'Tag','txt_message');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[157.5 90 112.5 22.5], ...
        'String','Reset parameters', ...
        'Tag','push_resetDataParams');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[174 10.5 101.25 22.5], ...
        'String','Load', ...
        'Tag','push_load');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[140.25 341.25 135 18.75], ...
        'String','Get params from file...', ...
        'Tag','push_loadParam');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'HorizontalAlignment','left', ...
        'Position',[112.5 236.25 157.5 67.5], ...
        'String',' ', ...
        'Style','listbox', ...
        'Tag','list_sourceParams', ...
        'TooltipString','Click to edit source directory and file(s)', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 281.25 101.25 22.5], ...
        'String','Source files...', ...
        'Tag','push_sourceDir');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[6 10.5 101.25 22.5], ...
        'String','Close', ...
        'Tag','push_close');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'FontName','courier', ...
        'Position',[11.25 112.5 258.75 101.25], ...
        'String',' ', ...
        'Style','listbox', ...
        'Tag','lst_dataParams', ...
        'TooltipString','Click on a parameter to edit it', ...
        'Value',1);
if nargout > 0, fig = h0; end
function fig = ui_main()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
```

```
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.


h0 = figure('Units','points', ...
        'Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        ...
        'FileName','e:/bruce/work3/src/ui_main.m', ...
        'MenuBar','none', ...
        'Name','Main', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576.0000000000001 432.0000000000002], ...
        'PaperUnits','points', ...
        'Position',[17.25 288 413.25 153.75], ...
        'Tag','Fig1', ...
        'Visible','on');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ListboxTop',0, ...
        'Position',[277.5 7.5 131.25 142.5], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ListboxTop',0, ...
        'Position',[7.5 7.5 131.25 142.5], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ListboxTop',0, ...
        'Position',[142.5 7.5 131.25 142.5], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ListboxTop',0, ...
        'Position',[11.25 101.25 123.75 22.5], ...
        'String','Load Data Files...', ...
        'Tag','push_dataSource', ...
        'TooltipString','Load MRI data files and specify parameters');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ForegroundColor',[0 0 0.627450980392157], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 45 123.75 22.5], ...
        'Style','text', ...
        'Tag','txt_magnet');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ForegroundColor',[0 0 0.627450980392157], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 11.25 123.75 33.75], ...
        'Style','text', ...
        'Tag','txt_sourceDir');
h1 = uicontrol('Parent',h0, ...
```

```
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ListboxTop',0, ...
        'Position',[146.25 101.25 123.75 22.5], ...
        'String','Set Regions of Interest...', ...
        'Tag','push_ROIlist', ...
        'TooltipString','Specifiy regions of interest and run calculations');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ForegroundColor',[0 0 0.627450980392157], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[146.25 11.25 123.75 56.25], ...
        'Style','text', ...
        'Tag','txt_ROImsg');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ListboxTop',0, ...
        'Position',[281.25 11.25 123.75 22.5], ...
        'String','Load Results from File...', ...
        'Tag','push_loadResults', ...
        'TooltipString','Load previously saved calculation results');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 123.75 123.75 22.5], ...
        'String','Data Source', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[146.25 123.75 123.75 22.5], ...
        'String','Generate T1 Map', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'Enable','off', ...
        'ListboxTop',0, ...
        'Position',[281.25 101.25 123.75 22.5], ...
        'String','View Results...', ...
        'Tag','push_viewResults', ...
        'TooltipString','View calculation results');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[281.25 123.75 123.75 22.5], ...
        'String','View T1 Map', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ForegroundColor',[0 0 0.627450980392157], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[281.25 33.75 123.75 33.75], ...
        'Style','text', ...
        'Tag','txt_resultsSaved');
h1 = uicontrol('Parent',h0, ...
```

```
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'Callback','include_globals; activeFig=''main''; uiresume(fh_main);', ...
        'ListboxTop',0, ...
        'Position',[11.25 75 123.75 22.5], ...
        'String','Select Data...', ...
        'Tag','push_selectData', ...
        'TooltipString','Select and align data for calculation; set background');
if nargout > 0, fig = h0; end
function fig = ui_ROIlist()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.


h0 = figure('Units','points', ...
        'Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        ...
        'FileName','e:/bruce/work3/src/ui_ROIlist.m', ...
        'MenuBar','none', ...
        'Name','Regions of Interest', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[43.5 120 276.75 318.75], ...
        'Tag','fig_main', ...
        'Visible','on');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[10.5 105 258.75 174], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[16.5 246.75 101.25 22.5], ...
        'String','Regions of interest', ...
        'Style','text', ...
        'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[16.5 224.25 90 22.5], ...
        'String','Add', ...
        'Tag','push_addROI');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'Position',[117.75 110.25 146.25 135], ...
        'String','', ...
        'Style','listbox', ...
        'Tag','list_ROIlist', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[16.5 201.75 90 22.5], ...
        'String','Del', ...
        'Tag','push_delROI');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
```

```
        'ListboxTop',0, ...
        'Position',[156.75 283.5 112.5 22.5], ...
        'String','Get ROIs from file...', ...
        'Tag','push_loadFile');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[180 11.25 90 22.5], ...
        'String','Make map', ...
        'Tag','push_exe');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ForegroundColor',[0 0 0.501960784313725], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 68.25 258.75 33.75], ...
        'Style','text', ...
        'Tag','txt_message');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 36.75 90 22.5], ...
        'String','Ok', ...
        'Tag','push_close');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 11.25 90 22.5], ...
        'String','Cancel', ...
        'Tag','push_cancel');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[16.5 134.25 101.25 22.5], ...
        'String','Square ROI', ...
        'Style','checkbox', ...
        'Tag','chk_square');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[27.75 111.75 45 22.5], ...
        'Style','edit', ...
        'Tag','edit_square');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[72.75 111.75 45 22.5], ...
        'String','pixels', ...
        'Style','text', ...
        'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[16.5 168 90 22.5], ...
        'String','Add entire image', ...
        'Tag','push_addEntire');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[146.25 36.75 123.75 22.5], ...
        'String','Save changes to file', ...
        'Style','checkbox', ...
        'Tag','chk_saveChanges');
if nargout > 0, fig = h0; end
function fig = ui_selectData()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
```

```
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.


h0 = figure('Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        ...
        'FileName','e:/bruce/work3/src/ui_selectData.m', ...
        'MenuBar','none', ...
        'Name','Select Data', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[60 345 406 256], ...
        'Tag','Fig3', ...
        'Visible','on');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[5.25 42 292.5 56.25], ...
        'Style','frame', ...
        'Tag','Frame2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[5.25 104.25 292.5 78.75], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 45 78.75 22.5], ...
        'String','Select...', ...
        'Tag','push_setBackground');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 133.5 168.75 22.5], ...
        'Style','edit', ...
        'Tag','edit_dataSequence', ...
        'TooltipString','Enter the data time-points to be used in calculation');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 156 135 22.5], ...
        'String','Data sequence:', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 111 90 22.5], ...
        'String','Plot data points', ...
        'Tag','push_viewPoints', ...
        'TooltipString','Plot data value vs. time for a selected pixel');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[191.25 133.5 101.25 22.5], ...
        'String','View image seq.', ...
        'Tag','push_viewImg', ...
        'TooltipString','View data sequence as a series of images');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
```

```
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[180 45.75 33.75 22.5], ...
        'Style','text', ...
        'Tag','txt_brightness');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[236.25 45 56.25 22.5], ...
        'String','Auto', ...
        'Tag','push_autoBrightness');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[123.75 67.5 56.25 22.5], ...
        'String','Brightness', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[180 67.5 112.5 16.5], ...
        'Style','slider', ...
        'Tag','sld_brightness', ...
        'TooltipString','Change brightness of background image');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[219 5.25 78.75 22.5], ...
        'String','Ok', ...
        'Tag','push_close');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[6 5.25 78.75 22.5], ...
        'String','Cancel', ...
        'Tag','push_cancel');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 67.5 101.25 22.5], ...
        'String','Background image', ...
        'Style','text', ...
        'Tag','StaticText2', ...
        'TooltipString','Select image to use as background');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[191.25 111 101.25 22.5], ...
        'String','Align images...', ...
        'Tag','push_align');
if nargout > 0, fig = h0; end
function fig = ui_viewResults()
% This is the machine-generated representation of a Handle Graphics object
% and its children.  Note that handle values may change when these objects
% are re-created. This may cause problems with any callbacks written to
% depend on the value of the handle at the time the object was saved.
% This problem is solved by saving the output as a FIG-file.
%
% To reopen this object, just type the name of the M-file at the MATLAB
% prompt. The M-file and its associated MAT-file must be on your path.
%
% NOTE: certain newer features in MATLAB may not have been saved in this
% M-file due to limitations of this format, which has been superseded by
% FIG-files.  Figures which have been annotated using the plot editor tools
% are incompatible with the M-file/MAT-file format, and should be saved as
% FIG-files.


h0 = figure('Units','points', ...
```

```
        'Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        ...
        'FileName','e:/bruce/work3/src/ui_viewResults.m', ...
        'MenuBar','none', ...
        'Name','Results', ...
        'NumberTitle','off', ...
        'PaperPosition',[18 180 576 432], ...
        'PaperUnits','points', ...
        'Position',[44.25 315.75 417 293.25], ...
        'Tag','Fig1', ...
        'Visible','on');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[207.75 41.25 202.5 151.5], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[207.75 197.25 202.5 90], ...
        'Style','frame', ...
        'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 259.5 101.25 22.5], ...
        'String','Regions of Interest', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'HorizontalAlignment','left', ...
        'Max',2, ...
        'Position',[11.25 123.75 191.25 112.5], ...
        'String',' ', ...
        'Style','listbox', ...
        'Tag','list_ROIlist', ...
        'Value',1);
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 101.25 90 22.5], ...
        'String','Select all', ...
        'Style','checkbox', ...
        'Tag','chk_selectAll');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 11.25 90 22.5], ...
        'String','Matlab Shell...', ...
        'Tag','push_prompt');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'FontWeight','bold', ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[213.75 157.5 150 22.5], ...
        'String','Image map display', ...
        'Style','text', ...
        'Tag','txt_mapSettingsBox');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[315 45 90 22.5], ...
        'String','Save figure to file...', ...
        'Tag','push_saveImg');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
```

```
        'ListboxTop',0, ...
        'Position',[213.75 90 90 22.5], ...
        'String','Color bar', ...
        'Style','checkbox', ...
        'Tag','chk_colorbar');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[213.75 258.75 123.75 22.5], ...
        'String','Relaxtion equation:', ...
        'Style','text', ...
        'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[213.75 112.5 101.25 22.5], ...
        'String','Map range:', ...
        'Style','text', ...
        'Tag','StaticText6');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[213.75 135 112.5 22.5], ...
        'String','Background range:', ...
        'Style','text', ...
        'Tag','StaticText6');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','right', ...
        'ListboxTop',0, ...
        'Position',[303.75 112.5 33.75 22.5], ...
        'String','0 to ', ...
        'Style','text', ...
        'Tag','StaticText8');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[337.5 135 56.25 22.5], ...
        'Style','edit', ...
        'Tag','edit_bgMax');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[1 1 1], ...
        'ListboxTop',0, ...
        'Position',[337.5 112.5 56.25 22.5], ...
        'Style','edit', ...
        'Tag','edit_tMax');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','right', ...
        'ListboxTop',0, ...
        'Position',[303.75 135 33.75 22.5], ...
        'String','0 to ', ...
        'Style','text', ...
        'Tag','StaticText8');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[315 11.25 90 22.5], ...
        'String','Close', ...
        'Tag','push_close');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[11.25 236.25 191.25 22.5], ...
        'String','Select ROI(s)  (Ctrl-click to select multiple)', ...
        'Style','text', ...
        'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
```

```
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[213.75 67.5 90 22.5], ...
        'String','ROI labels', ...
        'Style','checkbox', ...
        'Tag','chk_labels');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[11.25 67.5 90 22.5], ...
        'String','Add ROIs...', ...
        'Tag','push_ROIs');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[213.75 225 191.25 33.75], ...
        'Style','text', ...
        'Tag','txt_eqn');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[315 202.5 90 22.5], ...
        'String','Plot curve fit...', ...
        'Tag','push_analyzeCurve');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[213.75 202.5 90 22.5], ...
        'String','View details...', ...
        'Tag','push_analyzeParams');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'ListboxTop',0, ...
        'Position',[315 67.5 90 22.5], ...
        'String','Edit title...', ...
        'Tag','push_figTitle');
h1 = uicontrol('Parent',h0, ...
        'Units','points', ...
        'BackgroundColor',[0.752941176470588 0.752941176470588 0.752941176470588], ...
        'HorizontalAlignment','left', ...
        'ListboxTop',0, ...
        'Position',[213.75 45 90 22.5], ...
        'String','ROI details', ...
        'Style','checkbox', ...
        'Tag','chk_ROIdetails');
if nargout > 0, fig = h0; end
```

A.2 Data Processing Files

A.2.1 Definition functions

```
% DEFINITIONS Global definitions files for MRI mapper program

% Bruce Po - igan@mit.edu

% Put list of magnets in here
magnet_list = { 'brukerIR' ; 'brukerSR' ; 'brukerT2'; 'siemensIR' ; ...
                'geIR' };

% For each magnet listed, the program expects the following files
% (replace 'magnet' with the name listed above):
%
%   def_magnet.m       Definitions file
%   load_magnet.m      Loads MRI
%   exe_magnet.m       Runs the MRI analysis
function ret = def_brukerIR( action, varargin )
%DEF_BRUKERIR Definitions file.

  include_globals;
```

```
ret = '';

switch( action )

 case 'default'      % reset dataParams and cm to default values
  % inputs:  none
  % modifies:       dataParams, cm
  dataParams = struct( ...
      'dataScale', 1/5e6, ...
      'msize', [64 64], ...
      'noise_level', 1, ...
      'no_of_timeVar', 11, ...
      'no_of_slices', 1, ...
      'slice', 1, ...
      'TR', 15, ...
      'TI', [.025 .075 .175 .3 .45 .6 .75 1.0 3.0 7.0 15.0], ...
      'data_sequence', 1:11, ...
      'initial_guess_t1', 500 );
  sourceParams = struct( ...
      'magnet', 'brukerIR', ...
      'sourceDir', '', ...
      'sourceFile', '2DSEQ' );
  paramPrompts = struct ( ...
      'TR', 'TR time (seconds):', ...
      'TI', 'TI times (seconds):', ...
      'msize', 'Image size [h,w]:', ...
      'data_sequence', ['Which data points to use (can be edited' ...
                  ' after loading):'], ...
      'initial_guess_t1', 'Initial guess for T1 (ms):' );
  viewParams = struct( ...
      'maxImgVal', 'auto', ...
      'maxTval', 'auto', ...
      'minTval', 0, ...
      'figTitle', '', ...
      'figxlabel', '', ...
      'figylabel', '', ...
      'figColorbar', 1, ...
      'figROIlabels', 0, ...
      'ROIdetails', 0 );

 case 'input'                % inputs:      fieldname
                      % modifies:    dataParams
  fieldname = varargin{1};

  switch( fieldname )
   case { 'sourceDir', 'sourceFile', 'get_sourceDir' }
    if( strcmp(computer, 'PCWIN' )), mask = '*.*';
    else mask = '*';
    end;
    [filen pathn] = uigetfile( mask, 'Select data file' );
    if( pathn~=0 )
      sourceParams.sourceDir = pathn;
      sourceParams.sourceFile = filen;
      ret = 1;       % modified flag
    else
      ret = 0;
    end;

   case 'no_of_timeVar'
    msgbox( ['no_of_timeVar is updated automatically and cannot be' ...
             ' manually edited'], 'Note');
    ret = 0;

   otherwise
    % use the default dialog box for modifying fields
    tempParams = inputFieldDlg( dataParams, fieldname, '', paramPrompts );
    if( isstruct(tempParams) )
      dataParams = tempParams;
      ret = 1;
    else
      ret = 0;
    end;

    % fix up the user input if necessary
    if( strcmp(fieldname,'TI') )          % for inversion recovery
      % reset some other parameters (to be safe)
      dataParams.no_of_timeVar = length(dataParams.TI);
      dataParams.data_sequence = 1:dataParams.no_of_timeVar;
    end;

  end %switch( fieldname )
```

```
    case 'showEqn'
     ret = 'abs( p(1) * (1 - 2*p(2)*exp(-TI/p(3)) + exp(-TR/p(3))))';

  end %switch( action )

function ret = def_brukerSR( action, varargin )
% DEF_BRUKERSR Definitions file.
%    Relies def_brukerIR.m for some parts.

  include_globals;
  ret = '';

  switch( action )

    case 'default'      % reset dataParams and cm to default values
     dataParams = struct( ...
         'dataScale', 1/5e6, ...
         'msize', [128 128], ...
         'noise_level', 4, ...
         'no_of_timeVar', 6, ...
         'no_of_slices', 1, ...
         'slice', 1, ...
         'TR', [0.04 0.1 0.25 0.5 0.8 1.5], ...
         'data_sequence', 1:6, ...
         'initial_guess', [30 3] );
     sourceParams = struct( ...
         'magnet', 'brukerSR', ...
         'sourceDir', '', ...
         'sourceFile', '2DSEQ' );
     paramPrompts = struct ( ...
         'no_of_timeVar', 'Number of inversion times:', ...
         'TR', 'Enter the TR times (seconds):', ...
         'msize', 'Enter image size [h,w]:', ...
         'data_sequence', ['Which data points to use (can be edited' ...
                  ' after loading):'], ...
         'initial_guess', 'Initial guesses for M0 and T1 (seconds):' );
     viewParams = struct( ...
         'maxImgVal', 'auto', ...
         'maxTval', 'auto', ...
         'minTval', 0, ...
         'figTitle', '', ...
         'figxlabel', '', ...
         'figylabel', '', ...
         'figColorbar', 1, ...
         'figROIlabels', 0, ...
         'ROIdetails', 0 );

    case 'showEqn'
     ret = 'p(1) * (1 - exp(-TR/p(2)))';

    case 'no_of_timeVar'
     ret = def_brukerSR( 'input', 'TR' );

    otherwise
     ret = def_brukerIR( action, varargin{:} );

     % fix up the user input if necessary
     fieldname = varargin{1};
     if( strcmp(fieldname,'TR') )            % for saturation recovery
       % reset some other parameters (to be safe)
       dataParams.no_of_timeVar = length(dataParams.TR);
       dataParams.data_sequence = 1:dataParams.no_of_timeVar;
     end;

  end; %switch( action )

function ret = def_brukerT2( action, varargin )
% DEF_BRUKERT2 Definitions file.
%
%   Relies def_brukerIR.m for some parts.

  include_globals;
  ret = '';

  switch( action )

    case 'default'      % reset dataParams and cm to default values
     % inputs:  none
     % modifies:       dataParams, cm
     dataParams = struct( ...
```

```
        'dataScale', 1/5e6, ...
        'msize', [128 128], ...
        'no_of_slices', 1, ...
        'slice', 1, ...
        'noise_level', 1, ... % use this?
        'initial_guess', [50 300], ...
        'no_of_timeVar', 32, ...
        'TR', 10, ...
        'TE', 20:20:640, ...
        'data_sequence', 2:32 );
     sourceParams = struct( ...
        'magnet', 'brukerT2', ...
        'sourceDir', '', ...
        'sourceFile', '2DSEQ' );
     paramPrompts = struct ( ...
        'no_of_timeVar', 'Number of echo times:', ...
        'TR', 'TR time (seconds):', ...
        'TE', 'TE times (ms):', ...
        'msize', 'Image size [h,w]:', ...
        'data_sequence', ['Which data points to use (can be edited' ...
                    ' after loading):'], ...
        'initial_guess', 'Initial guesses for M0 and T2 (ms):' );
     viewParams = struct( ...
        'maxImgVal', 'auto', ...
        'maxTval', 'auto', ...
        'minTval', 0, ...
        'figTitle', '', ...
        'figxlabel', '', ...
        'figylabel', '', ...
        'figColorbar', 1, ...
        'figROIlabels', 0, ...
        'ROIdetails', 0 );


    case 'showEqn'
     ret = 'abs( p(1)*exp(-TE/p(2) )';

    case 'no_of_timeVar'
     ret = def_brukerT2( 'input', 'TE' );

    otherwise
     ret = def_brukerIR( action, varargin{:} );

     % fix up the user input if necessary
     fieldname = varargin{1};
     if( strcmp(fieldname,'TE') )             % for T2
       % reset some other parameters (to be safe)
       dataParams.no_of_timeVar = length(dataParams.TE);
       dataParams.data_sequence = 2:dataParams.no_of_timeVar;
     end;

   end %switch( action )


function ret = def_ge( action, varargin )
%DEF_GEIR Definitions file for struct dataParams.

   include_globals;
   ret = '';

   switch( action )

    case 'default'     % reset dataParams and cm to default values
     % inputs:  none
     % modifies:       dataParams, cm
     sourceParams = struct( ...
        'magnet', 'geIR', ...
        'sourceDir', '', ...
        'sourceFiles', '' );
     dataParams = struct( ...
        'dataScale', 1, ...
        'msize', [512 512], ...
        'no_of_timeVar', 7, ...
        'TR', 1800, ...
        'TI', [50 75 180 325 650 1100 1650], ...
        'data_sequence', 1:7, ...
        'noise_level', 1, ...
        'median_filter', 'no' );
     %'imgScale', 1/8, ...     % for displaying
     %'filen_base', '', ...
     %'filen_begin', '001', ...
```

```
    paramPrompts = struct ( ...
        'no_of_timeVar', 'Number of inversion times:', ...
        'TI', 'TI times (miliseconds):', ...
        'TR', 'TR time (miliseconds):', ...
        'msize', 'Enter image size [h,w]:', ...
        'data_sequence', ['Which data points to use (can be edited' ...
                    ' after loading):'], ...
        'median_filter', 'Apply median filter to data? [yes/no]');
    viewParams = struct( ...
        'maxImgVal', 'auto', ...
        'maxTval', 'auto', ...
        'minTval', 0, ...
        'figTitle', '', ...
        'figxlabel', '', ...
        'figylabel', '', ...
        'figColorbar', 1, ...
        'figROIlabels', 0, ...
        'ROIdetails', 0 );


   case 'input'                % inputs:      fieldname
                          % modifies:    dataParams
    fieldname = varargin{1};

    switch( fieldname )

     case 'get_sourceDir'     % this is not really a field in the
                                % struct sourceParams
       ret = def_siemensIR( 'input', 'get_sourceDir', '*' );
%       [filen pathn] = uigetfile( '*.*', 'Select GE data file' );
%       if( pathn~=0 )
%       % get the parts of the file name (base name, begin num, and end num)
%       [ppath,ffile,ext,ver] = fileparts(filen);
%       sourceParams.sourceDir = pathn;
%       filen_base = ffile;
%       try
%         filen_begin = ext(2:4);
%         filen_end = findGELast( pathn, filen_base );
%       catch
%         errordlg( ['Unable to parse directory.  You may need to' ...
%                    'manually edit the file list.'], 'Warning', 'modal' );
%         sourceParams.sourceFiles = dirFiles( pathn );
%         ret = 1;     % flag that things have been modified
%         return;
%       end;

%       % create filename list
%       sourceParams.sourceDir = pathn;
%       sourceParams.sourceFiles = '';
%       dataParams.no_of_timeVar = str2num(filen_end) - str2num(filen_begin ) + 1;
%       for n=1:dataParams.no_of_timeVar
%         fname = [filen_base '.' num2str(n, '%03d')];
%         sourceParams.sourceFiles = strvcat( sourceParams.sourceFiles, ...
%                                             fname );
%       end; %for
%       ret = 1;       % flag that things have been modified
%       else %if
%       ret = 0;
%       end;

     case { 'sourceDir', 'sourceFiles' }
%       a = questdlg( [ 'Pushing the ''Source files'' button will ' ...
%                    'automatically set this field.  Are you ' ...
%                    'sure you wish to manually modify this field ' ...
%                    'instead?'], ...
%                  'Confirm', 'Yes', 'Cancel', 'Cancel' );
%       if( strcmp(a, 'Cancel') )    ret = 0; return; end;
       a = inputdlg( {'Enter the source directory:', ...
                      'Enter the source file names (one line each):'}, ...
                    'Source files', [1; 4], ...
                    {sourceParams.sourceDir, sourceParams.sourceFiles} );
       if( isempty( a ))       ret = 0; return;
       else
         sourceParams.sourceDir = a{1};
         sourceParams.sourceFiles = a{2};
         dataParams.no_of_timeVar = size(a{2},1);
         ret = 1;
       end; %if

     case {'no_of_timeVar'}
      tempParams = inputFieldDlg( dataParams, fieldname, 'confirm', ...
```

```
                                paramPrompts );
      if( isstruct(tempParams) )
        dataParams = tempParams;
        ret = 1;
      else
        ret = 0;
      end;

     otherwise
      % use the default dialog box for changing the field value
      tempParams = inputFieldDlg( dataParams, fieldname, '', paramPrompts );
      if( isstruct(tempParams) )
        dataParams = tempParams;
        ret = 1;
      else
        ret = 0;
      end;

      % fix up the user input if necessary
      if( strcmp(fieldname,'median_filter' ))
        if( ~strcmp(dataParams.median_filter,'yes' ))
          dataParams.median_filter = 'no';
        end;
      elseif( strcmp(fieldname,'msize') )
        a = dataParams.msize;
        if( length(a)>1 )
          dataParams.msize = [a(1) a(2)];
        else
          dataParams.msize = [a(1) a(1)];
        end;
      elseif( strcmp(fieldname,'TI') )
        % reset some other parameters (to be safe)
        dataParams.no_of_timeVar = length(dataParams.TI);
        dataParams.data_sequence = 1:dataParams.no_of_timeVar;
      end;
    end; %switch( fieldname )

   case 'showEqn'
    ret = 'abs( p(1) * ( 1 - 2*p(2)*exp(-TI/p(3)) + exp(-TR/p(3)) ))';

  end %switch( action )

function extension = findGELast( pathn, filen_base )
% Finds the extension of the last file in the series.
%  For example, if filen_base = 'I'
%  and the directory contains files I.001 I.002 I.003, then the function
%  returns the string '003'.

  fileList = dir( [fullfile(pathn, filen_base) '.*' ] );
  fileList = strvcat( fileList.name );
  for i = 1:length(fileList);
    [pathn,filen,ext,ver] = fileparts(fileList(i,:));
    extList(i,:) = ext(2:4);
  end

  extList2 = str2num(extList);
  [val i] = max(extList2);
  extension = extList(i,:);


function ret = def_siemensIR( action, varargin )
%DEF_SIEMENSIR Definitions file for struct dataParams.

  include_globals;
  ret = '';

  switch( action )

   case 'default'     % reset dataParams and cm to default values
    sourceParams = struct( ...
        'magnet', 'siemensIR', ...
        'sourceDir', '', ...
        'sourceFiles', '' );
    dataParams = struct( ...
        'dataScale', 1, ...
        'msize', [512 512], ...
        'no_of_timeVar', 7, ...
        'TR', 1800, ...
        'TI', [1680 1100 650 350 180 75 25], ...
        'data_sequence', 1:7, ...
        'noise_level', 1, ...
```

```
    'median_filter', 'no' );
    %'t1mapScale', 1, ...
    %'imgScale', 1/6, ... % for displaying
    %'patID', '', ...
    %'stuBegin', 0, ...
    %'stuEnd', 0, ...
    %'imgBegin', 0, ...
    %'imgEnd', 0, ...
    %'imgStep', 0 );
 paramPrompts = struct ( ...
    'no_of_timeVar', 'Number of inversion times:', ...
    'TI', 'TI times (miliseconds):', ...
    'TR', 'TR time (miliseconds):', ...
    'msize', 'Enter image size [h,w]:', ...
    'data_sequence', ['Which data points to use (can be edited' ...
            ' after loading):'], ...
    'median_filter', 'Apply median filter to data? [yes/no]');
 viewParams = struct( ...
    'maxImgVal', 'auto', ...
    'maxTval', 'auto', ...
    'minTval', 0, ...
    'figTitle', '', ...
    'figxlabel', '', ...
    'figylabel', '', ...
    'figColorbar', 1, ...
    'figROIlabels', 0, ...
    'ROIdetails', 0 );

 ret = '';

case 'input'                % inputs:      command
                  % modifies:    sourceParams or dataParams
 command = varargin{1};

 switch( command ) % command or fieldname

  case 'get_sourceDir' % this command is for automatically
                       % setting the source directory and files
   %[filen pathn] = uigetfile( '*.ima', ['Select the first data' ...
   %     ' file in series'] );
   if( strmatch('*', varargin))
    if( strcmp(computer, 'PCWIN' )), mask = '*.*';
    else mask = '*';
    end;
   else
    mask = '*.ima';
   end;
   [filen pathn] = uigetfile( mask, ['Select a data file'] );

   if( pathn~=0 )
    [fileNames fileInfo] = dirFiles( pathn );
    [sel,ok] = listdlg( ...
        'PromptString', strvcat('Select data files:', ...
                ['(Ctrl-click or Shift-click for' ...
              ' multiple)']), ...
        'SelectionMode', 'multiple', ...
        'ListSize', [200 200], ...
        'ListString', fileInfo );
    if( ok )
      dataParams.no_of_timeVar = length(sel);
      sourceParams.sourceDir = pathn;
      sourceParams.sourceFiles = fileNames(sel,:);
      ret = 1;
    else
      ret = 0;
    end;

    % get the parts of the file name
    %a = setSiemensParams( pathn, filen );
    %dataParams = setSiemensParams( dataParams, pathn, filen );
    %ret = 1;      % flag that things have been modified
    %if( a==0 )
    %  errordlg( ['Unable to parse directory.  You may need to' ...
    %          'manually edit the file list.'], 'Warning', 'modal' );
    %sourceParams.sourceFiles = dirFiles( pathn );
    %  return;
    %end;
   else %if
    ret = 0;
   end; %if
```

```
    case { 'sourceDir', 'sourceFiles' } % these fields are for
                                        % manually setting the source
     fieldname = command;
%      a = questdlg( [ 'Pushing the ''Source files'' button will ' ...
%                    'automatically set this field.  Are you ' ...
%                    'sure you wish to manually modify this field ' ...
%                    'instead?'], ...
%                  'Confirm', 'Yes', 'Cancel', 'Cancel' );
%      if( strcmp(a, 'Cancel') )    ret = 0; return; end;
     a = inputdlg( {'Enter the source directory:', ...
                    'Enter the source file names (one line each):'}, ...
                   'Source files', [1; 4], ...
                   {sourceParams.sourceDir, sourceParams.sourceFiles} );
     if( isempty( a ))       ret = 0; return;
     else
       sourceParams.sourceDir = a{1};
       sourceParams.sourceFiles = '';
       fileNames = a{2};
       for( i=1:size(fileNames,1) )
         if( ~prod(isspace(fileNames(i,:))) )       % non-empty items only
           sourceParams.sourceFiles = strvcat(sourceParams.sourceFiles, ...
                                              fileNames(i,:));
         end;
       end;
       dataParams.no_of_timeVar = size(sourceParams.sourceFiles,1);
       ret = 1;
     end; %if

    % these fields are used by the program to parse the file directory
    case {'patID', 'stuBegin', 'stuEnd', 'imgBegin', 'imgEnd', 'imgStep', ...
          'no_of_timeVar'}
     fieldname = command;
     tempParams = inputFieldDlg( dataParams, fieldname, 'confirm', ...
                                 paramPrompts );
     if( isstruct(tempParams) )
       dataParams = tempParams;
       ret = 1;
     else
       ret = 0;
     end;

    otherwise
     % use the default dialog box for changing the field value
     fieldname = command;
     tempParams = inputFieldDlg( dataParams, fieldname, '', paramPrompts );
     if( isstruct(tempParams) )
       dataParams = tempParams;
       ret = 1;
     else
       ret = 0;
     end;

     % fix up the user input if necessary
     if( strcmp(fieldname,'median_filter' ))
       if( ~strcmp(dataParams.median_filter,'yes' ))
         dataParams.median_filter = 'no';
       end;
     elseif( strcmp(fieldname,'msize') )
       a = dataParams.msize;
       if( length(a)>1 )
         dataParams.msize = [a(1) a(2)];
       else
         dataParams.msize = [a(1) a(1)];
       end;
     elseif( strcmp(fieldname,'TI') )
       % reset some other parameters (to be safe)
       dataParams.no_of_timeVar = length(dataParams.TI);
       dataParams.data_sequence = 1:dataParams.no_of_timeVar;
     end;
   end; %switch( command )

  case 'showEqn'
   ret = 'abs( p(1) * ( 1 - 2*p(2)*exp(-TI/p(3)) + exp(-TR/p(3)) ))';
 end %switch( action )


function ret = setSiemensParams( pathn, filen )
 global dataParams sourceParams;

% Sets up a whole bunch of file name parameters in dataParams based on
% the current directory and the file the user selected.  Specifically,
```

```
% patient ID, study numbers, and image numbers are set.

  sourceParams.sourceDir = pathn;

  dashLoc = findstr( filen, '-' );
  dotLoc = findstr( filen, '.' );
  if( length(dashLoc)<2 )
    ret = 0;
    return;
  end
  ret = 1;

  % set and patID
  patID = filen(1:dashLoc(1)-1);

  % set stuBegin and imgBegin
  stuBegin = filen(dashLoc(1)+1:dashLoc(2)-1);
  imgBegin = filen(dashLoc(2)+1:dotLoc-1);

  % find the list of image numbers for the study that was selected
  fileMask = [patID '-' stuBegin '-*.ima'];
  list = listSiemensNums( pathn, fileMask, 'imageNum' );

  % set imgStep and figure out which image in the study was selected
  list = strjust( list, 'left' );
  imagePos = strmatch( imgBegin, list );
  imgStep = size( list, 1 );

  % find stuEnd
  fileMask = [patID '-*-*.ima'];
  list = listSiemensNums( pathn, fileMask, 'studyNum' );
  stuEnd = list(size(list,1),:);     % pick the last one

  % find imgEnd
  fileMask = [patID '-' stuEnd '-*.ima'];
  list = listSiemensNums( pathn, fileMask, 'imageNum' );
  imgEnd = list(imagePos,:);

  stuBegin = str2num( stuBegin );
  stuEnd = str2num( stuEnd );
  imgEnd = str2num( imgEnd );
  imgBegin = str2num( imgBegin );

  dataParams.no_of_timeVar = stuEnd - stuBegin + 1;
  stu_num = stuBegin:stuEnd;
  ima_num = imgBegin:imgStep:imgEnd;

  % create filename list
  sourceParams.sourceFiles = '';
  for n=1:dataParams.no_of_timeVar,
    fname = [patID '-' int2str(stu_num(n)) '-' ...
             int2str(ima_num(n)) '.ima'];
    sourceParams.sourceFiles = strvcat( sourceParams.sourceFiles, ...
                                        fname );
  end; %for

  %ret = dataParams;
  %ret.dataDir = pathn;
  %ret.patID = patID;
  %ret.stuBegin = str2num(stuBegin);
  %ret.stuEnd = str2num(stuEnd);
  %ret.imgBegin = str2num(imgBegin);
  %ret.imgEnd = str2num(imgEnd);
  %ret.imgStep = imgStep;
```

A.2.2 Data loading functions

```
function ret = load_brukerIR
%LOAD_BRUKERIR

  include_globals;
  dp = dataParams;     % saves typing

  h = dlgInfo( 'Loading files...', '' );

  filen = fullfile(sourceParams.sourceDir, sourceParams.sourceFile);
  data = fopen(filen, 'r', 'b');
  if( data==-1 )
```

```
    errordlg( ['Could not open ' filen], 'Error', 'modal' );
    ret = 0;
    close( h );
    return;
  end;

  %figs = zeros(dp.msize(1)*dp.Msize(2), dp.no_of_timeVar);
  figs = zeros(dp.msize(1),dp.msize(2), dp.no_of_timeVar);
  for tinum = 1:dp.no_of_timeVar
    temp = fread(data,[dp.msize(1)*dp.msize(2) dp.no_of_slices], 'long');
    ima = reshape(temp(:,dp.slice), dp.msize(2), dp.msize(1));
    %ima = ima'*dp.dataScale;
    %figs(:,tinum) = ima(:);
    ima = ima'*dp.dataScale;
    figs(:,:,tinum) = ima;
  end;
  img_background = ima;

  fclose(data);

  %imas = imas.*dp.dataScale;

  %figs = '';
  %for n = 1:dp.no_of_timeVar,
  %  ima = reshape(imas(:,n),dp.msize(1),dp.msize(2));
  %  ima = flipud(rot90(ima));
  %  figs(:,n) = ima(:);
  %end

  %img_background = ima;                    % use last image for background
  if( ishandle(h) ), close(h);, end;

  ret = 1;


function ret = load_brukerSR
% LOAD_BRUKERSR

  ret = load_brukerIR;

function ret = load_brukerT2
% LOAD_BRUKERT2

  include_globals;

  ret = load_brukerIR;
  img_background = figs(:,:,1);

function ret = load_geIR
% LOAD_GEIR
  ret = load_siemensIR( 'ge' );

function ret = load_siemensIR( varargin )
% LOAD_SIEMENSIR

  include_globals;

  % GE magnet uses nearly the same data format
  GEmode = 0;
  if( nargin>0 )
    if( strcmp( varargin{1}, 'ge' ))
      GEmode = 1;
    end;
  end;

  figs=zeros(dataParams.msize(1),dataParams.msize(2),dataParams.no_of_timeVar);

  h = waitbar( 0, ['Loading files...'] );
  for n=1:dataParams.no_of_timeVar

    fname = fullfile(sourceParams.sourceDir, ...
                     deblank( sourceParams.sourceFiles(n,:)));

    fid=fopen(fname,'r','b');
    if( fid==-1 )
      errordlg( ['Could not open ' fname], 'Error', 'modal' );
      close( h );
      ret = 0;
      return;
    end;
```

```
    % is this a DICOM format image? ..............................
    %   DICM format seems to require a different byte ordering for data
    fseek( fid, 128, -1 );
    str = fread( fid, 4, 'schar' );
    str = char(str');
    if( strcmp(str, 'DICM'))
      fclose( fid );
      fid=fopen(fname,'r');
    end;

    % here's a more robust way to find the offset ...................
    finfo = dir( fname );
    offset = finfo.bytes - (dataParams.msize(1)*dataParams.msize(2)*2);
    fseek(fid,offset,-1);

    %figs(:,:,n)=fread(fid,[dataParams.msize(1),dataParams.msize(2)],'short');
    ima=fread(fid,[dataParams.msize(2),dataParams.msize(1)],'uint16');
    fclose(fid);
    waitbar( n/dataParams.no_of_timeVar, h );
    ima = ima';
    if( strcmp(dataParams.median_filter,'yes'))
      ima = medfilt2(ima,[2 2]);
    end;
    figs(:,:,n) = ima;
    waitbar( n/dataParams.no_of_timeVar, h )
  end
  close(h);
  figs = figs .* dataParams.dataScale;
  img_background = figs(:,:,1);

  ret = 1;
```

A.2.3 T1/T2 Mapping functions

```
function f = b_t1_fun_ir(p,SI,TI,TR)
% B_T1_FUN_IR Inversion recovery relaxation equation.

err = SI - abs( p(1) * (1 - 2*p(2)*exp(-TI/p(3)) + exp(-TR/p(3))));
f = sum(err.^2);
function f = b_t2_fun(p,SI,TE)
% B_T2_FUN T2 relaxation equation.

  err = SI - abs( p(1)*exp(-TE/p(2) ));
  % p(1) = amplitude
  % p(2) = T2 time

  f = sum(err.^2);

function f = excised_t1_fun_sr(p,SI,TR)
% EXCISED_T1_FUN_SR  Saturation recovery relaxation equation.

err = SI - ( p(1) * (1 - exp(-TR/p(2))));
f = sum(err.^2);
function f = s_t1_fun_ir(p,SI,TI,TR)
% S_T1_FUN_IR Inversion recovery relaxation equation.

err = SI - abs( p(1) * ( 1 - 2*p(2)*exp(-TI/p(3)) + exp(-TR/p(3)) ));
f = sum(err.^2);

function [t1map, t1mapi, pfitted] = exe_brukerIR( mask, dp )
%EXE_BRUKERIR

  global figs

  % pick out regions that are masked and above the noise theshold
  %mask = double(mask(:));
  mask = double(mask);
  %roind = find( figs(:,dp.no_of_timeVar).*mask > dp.noise_level);
  a = dp.data_sequence(length(dp.data_sequence));
  %roind = find( figs(:,a).*mask > dp.noise_level);
  roind = find( figs(:,:,a).*mask > dp.noise_level);
  if( isempty(roind) )
    t1map = ''; t1mapi = ''; pfitted = '';
    return;                    % abort because nothing to do
  end;

  % set up data to be processed array
  nn = 1;
```

```
  %for n = 1:dp.no_of_timeVar
  for n = dp.data_sequence
    %temp = figs(:,n);
    temp = figs(:,:,n);
    temp = temp(:);
    Data(:,nn) = temp(roind);
    nn = nn+1;
  end;

  %OPTIONS = 0;         % set relevant optimization params (for fmins)
  %OPTIONS = zeros(1,18);
  %OPTIONS = foptions;
  %OPTIONS(2) = 1e-2;    % Termination tolerance for x
  %OPTIONS(3) = 1e-2;    % Termination tolerance on F
  OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );
  TI = dp.TI(dp.data_sequence) * 1000;
  TR = dp.TR * 1000;

  s = size(Data,1);
  pfitted = zeros(s,5);
  initial_guess = [1 1 dp.initial_guess_t1];

  h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
  for n = 1:s,
      SI = Data(n,:);
      initial_guess(1) = max(SI);
      [sim,simi] = min(SI);   % sim is minimum value, simi is index of min.
      initial_guess(3) = TI(simi)/0.6;
      %pfitted(n,:) = fmins( 'b_t1_fun_ir', initial_guess, OPTIONS, ...
      %           [], SI, TI, TR );
      [pfitted(n,1:3) fval exitflag output] = ...
          fminsearch( 'b_t1_fun_ir', initial_guess, ...
                    OPTIONS, SI, TI, TR );
      pfitted(n,4) = fval/mean(SI);  % MSE normalized
      pfitted(n,5) = output.iterations;
      if( mod(n,10)==0 )
        waitbar( n/s, h );
      end;
    end
    close(h);
    clear Data;

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

%     t1_upper_bound = 2000;
%     t1_lower_bound = 25;
%     m_upper_bound = 1000;
%     m_lower_bound = 0;

%     pfitted(:,3)=pfitted(:,3).*(   pfitted(:,3) < t1_upper_bound  & ...
%                                    pfitted(:,3) > t1_lower_bound  & ...
%                                    pfitted(:,1) < m_upper_bound   & ...
%                                    pfitted(:,1) > m_lower_bound);

    t1_values = pfitted(:,3);
    %t1_values = t1_values./10;             % don't need to scale anymore?

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

    % save the data

    t1map = zeros(dp.msize(1)*dp.msize(2),1);
    t1map(roind) = t1_values;

    t1map = reshape(t1map, dp.msize(1), dp.msize(2));

    t1mapi = roind;    % index: pixels in the image which correspond to
                       % t1map and pfitted data

function [t1map, t1mapi, pfitted] = exe_brukerSR( mask, dp )
%EXE_BRUKERSR

  % dp is data parameters
  global figs;

    %mask = double(mask(:));
    mask = double(mask);

    % mask out the pixels that are < 4
    %temp1 = figs(:,dp.no_of_timeVar).*mask;
    a = dp.data_sequence(length(dp.data_sequence));
```

```
%temp1 = figs(:,a).*mask;
temp1 = figs(:,:,a).*mask;
[Im Jm Vm] = find(temp1 < dp.noise_level);

%for n = 1:size(Im,1),
%  mask(Im(n)) = 0;
%end
mask( Im ) = 0;

%for n = 1:dp.no_of_timeVar,
nn = 1;
for n = dp.data_sequence
  %Imasked(:,nn) = figs(:,n).*mask;
  temp = figs(:,:,n).*mask;
  Imasked(:,nn) = temp(:);
  nn = nn+1;
end

clear mask
%clear I ima imas mask scale siz n

[I, J, V] = find(Imasked(:,size(Imasked,2)));
if( isempty(I) )
  t1map = ''; t1mapi = ''; pfitted = '';
  return;                 % abort because nothing to do
end;


clear Data;
%for n = 1:dp.no_of_timeVar,
for n = 1:length(dp.data_sequence)
  Data(:,n) = Imasked(I,n);
end

clear Imasked J V

%OPTIONS = 0;         % set relevant optimization params (for fmins)
%OPTIONS = zeros(1,18);
%OPTIONS(2) = 1e-2;    % Termination tolerance for x
%OPTIONS(3) = 1e-2;    % Termination tolerance on F
OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );

s = size(Data,1);
pfitted = zeros(s,4);
TR = dp.TR(dp.data_sequence);

h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
for n = 1:s,
  SI = Data(n,:);
  %pfitted(n,:) = fmins( 'excised_t1_fun_sr', dp.initial_guess, OPTIONS,...
  %[], SI, TR);
  [pfitted(n,1:2) fval exitflag output] = ...
      fminsearch( 'excised_t1_fun_sr', dp.initial_guess, ...
                  OPTIONS, SI, TR);
  pfitted(n,3) = fval/mean(SI);  % MSE normalized
  pfitted(n,4) = output.iterations;
  if( mod(n,10)==0 )
    waitbar( n/s, h );
  end;
end
close(h);
%pfitted_backup = pfitted;       % this is never used

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

%    clear test1 spots
%    t1_upper_bound = 5;
%    spots = find(pfitted(:,2) > t1_upper_bound);
%    test1 = isempty(spots);

%    if test1 == 0;
%      for n = 1:size(spots)
%     pfitted(spots(n),3)=0;
%      end
%    end

%    clear test1 spots
%    t1_lower_bound = 0.05;
%    spots = find(pfitted(:,2) < t1_lower_bound);
%    test1 = isempty(spots);
```

```
%     if test1 == 0;
%       for n = 1:size(spots)
%      pfitted(spots(n),2)=0;
%       end
%     end

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

%     clear test1 spots
%     m_upper_bound = 100;
%     spots = find(pfitted(:,1) > m_upper_bound);
%     test1 = isempty(spots);

%     if test1 == 0;
%       for n = 1:size(spots)
%      pfitted(spots(n),2)=0;
%       end
%     end

%     clear test1 spots
%     m_lower_bound = 0;
%     spots = find(pfitted(:,1) < m_lower_bound);
%     test1 = isempty(spots);

%     if test1 == 0;
%       for n = 1:size(spots)
%      pfitted(spots(n),2)=0;
%       end
%     end


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

    % save the data

    t1map = zeros(dp.msize(1)*dp.msize(2),1);

    t1_values = pfitted(:,2);
    t1map(I) = t1_values;

    t1map = reshape(t1map, dp.msize(1), dp.msize(2));

    t1mapi = I;               % index: pixels in the image which correspond to
                        % t1map and pfitted data

function [t2map, t2mapi, pfitted] = exe_brukerT2( mask, dp )
%EXE_BRUKERT2

  global figs

  % pick out regions that are masked and above the noise theshold
  %mask = double(mask(:));
  mask = double(mask);
  %roind = find( figs(:,dp.no_of_timeVar).*mask > dp.noise_level);
  a = dp.data_sequence(length(dp.data_sequence));
  %roind = find( figs(:,a).*mask);
  roind = find( figs(:,:,a).*mask);
  if( isempty(roind) )
    t2map = ''; t2mapi = ''; pfitted = '';
    return;                   % abort because nothing to do
  end;

  % set up data to be processed array
  %for n = 1:dp.no_of_timeVar
  nn = 1;
  for n = dp.data_sequence
    %temp = figs(:,n);
    temp = figs(:,:,n);
    temp = temp(:);
    Data(:,nn) = temp(roind);
    nn = nn+1;
  end;

  %OPTIONS = foptions;   % set relevant optimization params (for fmins)
  %OPTIONS(2) = 1e-2;    % Termination tolerance for x
  %OPTIONS(3) = 1e-2;    % Termination tolerance on F
  OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );

  TE = dp.TE(dp.data_sequence);
```

```
  s = size(Data,1);
  pfitted = zeros(s,4);
  initial_guess = dp.initial_guess;

  h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
  for n = 1:s,
    SI = Data(n,:);
    %pfitted(n,:) = fmins( 'b_t2_fun', initial_guess, OPTIONS,  [], SI, TE );
    [pfitted(n,1:2) fval exitflag output] = ...
       fminsearch( 'b_t2_fun', initial_guess, OPTIONS,  SI, TE );
    pfitted(n,3) = fval/mean(SI);    % MSE normalized
    pfitted(n,4) = output.iterations;
    initial_guess = pfitted(n,:);    % use last value as initial guess
    if( mod(n,10)==0 )
      waitbar( n/s, h );
    end;
  end
  close(h);
  clear Data;

  t2_values = pfitted(:,2);

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

    % save the data

    t2map = zeros(dp.msize(1)*dp.msize(2),1);
    t2map(roind) = t2_values;

    t2map = reshape(t2map, dp.msize(1), dp.msize(2));

    t2mapi = roind;    % index: pixels in the image which correspond to
                       % t2map and pfitted data

function [t1map, t1mapi, pfitted] = exe_geIR( mask, dp )
%EXE_GEIR

  [t1map, t1mapi, pfitted] = exe_siemensIR( mask, dp );

function [t1map, t1mapi, pfitted] = exe_siemensIR( mask, dp )
%EXE_SIEMENSIR

  global figs

  %stu_length = dp.no_of_timeVar;
  %OPTIONS = foptions;   % set relevant optimization params (for fmins)
  %OPTIONS(2) = 1e-2;    % Termination tolerance for x
  %OPTIONS(3) = 1e-2;    % Termination tolerance on F
  OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );

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

    mask = double(mask);
    img = figs(:,:,1);
    temp1 = img.*mask;
    roind = find(temp1 > dp.noise_level);
    if( isempty(roind) )
      t1map = ''; t1mapi = ''; pfitted = '';
      return;                % abort because nothing to do
    end;

    clear img mask temp1;
    %Select data within the ROI for each image file and save it in Data
    clear Data;
    %for n = 1:stu_length,
    nn = 1;
    for n = dp.data_sequence
      clear temp
      temp = figs(:,:,n);
      temp = temp(:);
      Data(:,nn) = temp(roind);
      nn = nn + 1;
    end
    clear temp

    %Initial values and curve fitting
    s = size(Data,1);
    pfitted = zeros(s,5);
    iniguess = [130 1 500];   % see below iniguess
    TR = dp.TR;
    TI = dp.TI(dp.data_sequence);
```

```
    %str1=[' of total' int2str(s) ' pixels'];
    h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
    a = length(dp.data_sequence);
    for n = 1:s,              % index through pixels
      SI = Data(n,:);  % for a pixel, get from all images
      iniguess(1) = max(SI(1),SI(a));
      [sim,simi] = min(SI);   % sim is minimum value, simi is index of min.
      iniguess(3) = dp.TI(simi)/0.6;
      %pfitted(n,:) = fmins( 's_t1_fun_ir', iniguess, OPTIONS,[],SI,TI,TR );
      [pfitted(n,1:3) fval exitflag output] = ...
         fminsearch( 's_t1_fun_ir',iniguess,OPTIONS,SI,TI,TR );
      pfitted(n,4) = fval/mean(SI);  % MSE normalized
      pfitted(n,5) = output.iterations;
      if( mod(n,10)==0 )
       waitbar( n/s, h );
      end
    end
    close( h );

%     t1_upper_bound = 3000;
%     t1_lower_bound = 50;
%     a_upper_bound = 2;
%     a_lower_bound = 0.2;
%     m_upper_bound = mean(pfitted(:,1))+100;
%     m_lower_bound = 0;


%     pfitted(:,3)=pfitted(:,3).*(    pfitted(:,3) < t1_upper_bound  & ...
%                                     pfitted(:,3) < t1_upper_bound  & ...
%                                     pfitted(:,2) < a_upper_bound  & ...
%                                     pfitted(:,2) > a_lower_bound  & ...
%                                     pfitted(:,1) < m_upper_bound   & ...
%                                     pfitted(:,1) > m_lower_bound);


%     clear t1_values

    t1_values = pfitted(:,3);

    %scale down t1 values to fit 256 matrix
    %t1_values = t1_values./10;

    %create image of these values
    t1_ima = zeros(dp.msize(1)*dp.msize(2),1);
    t1_ima(roind) = t1_values;

    t1_ima = reshape(t1_ima,dp.msize(1),dp.msize(2));

    %processed = zeros(dp.msize,dp.msize,2);

    t1map = t1_ima;
    t1mapi = roind;
function plot_brukerIR( dataPoints, p, varargin )
  include_globals

  if( nargin>2 )
    logScale = varargin{1};
  else
    logScale = 0;
  end;

  TI = dataParams.TI(dataParams.data_sequence);
  timeIndex = 0:.01:max(TI);

  if( isempty(p) )
    plotIt( logScale, TI, dataPoints, '.' );
  else
    % generate the fitted curve
    %timeIndex = 0:.01:dataParams.TR;% [seconds]
    curve = brukerIR_fn( timeIndex, p );
    if( isempty(dataPoints) )
      plotIt( logScale, timeIndex, curve, '-' );
    else
      plotIt( logScale, TI, dataPoints, '.', ...
              timeIndex, curve, '-' );
    end;
  end; %if

% see b_t1_fun_ir.m
function ret = brukerIR_fn(t, p)
```

```
  % t is time, p is parameters
  global dataParams;

  TR = dataParams.TR*1000;           % convert to miliseconds
  t = t*1000;
  %ret = abs(A   * (1 - 2 *    exp(-t/t1)   + exp(-TR/t1)));
  ret = abs(p(1) * (1 - 2*p(2)*exp(-t/p(3)) + exp(-TR/p(3))));

function plot_brukerSR( dataPoints, p, varargin )
  include_globals

  if( nargin>2 )
    logScale = varargin{1};
  else
    logScale = 0;
  end;

  TR = dataParams.TR(dataParams.data_sequence);

  if( isempty(p) )
    plotIt( logScale, TR, dataPoints, '.' );
  else
    % generate the fitted curve
    timeIndex = 0:.01:TR(length(TR)); % [seconds]
    curve = brukerSR_fn( timeIndex, p );
    if( isempty(dataPoints) )
      plotIt( logScale, timeIndex, curve, '-' );
    else
      plotIt( logScale, TR, dataPoints, '.', timeIndex, curve, '-' );
    end;
  end; %if

% see excised_t1_fun_sr.m
function ret = brukerSR_fn(t, p)
  % t is time, p is parameters
  global dataParams;

  ret = p(1) * (1 - exp(-t/p(2)));


function plot_brukerT2( dataPoints, p, varargin )
  include_globals

  if( nargin>2 )
    logScale = varargin{1};
  else
    logScale = 0;
  end;

  TE = dataParams.TE(dataParams.data_sequence);

  if( isempty(p) )
    plotIt( logScale, TE, dataPoints, '.' );

  else
    % generate the fitted curve
    timeIndex = 0:10:(max(TE));      % [milliseconds]
    curve = brukerT2_fn( timeIndex, p );
    if( isempty(dataPoints) )
      plotIt( logScale, timeIndex, curve, '-' );
    else
      plotIt( logScale, TE, dataPoints, '.', timeIndex, curve, '-' );
    end;
  end; %if

% see b_t1_fun_ir.m
function ret = brukerT2_fn(t, p)
  % t is time, p is parameters
  global dataParams;

  %ret = 'abs( p(1)*exp(-TE/p(2) )';
  ret =   abs( p(1)*exp(-t/p(2)) );

function plot_geIR( dataPoints, p, varargin )
  plot_siemensIR( dataPoints, p, varargin{:} );

function plot_siemensIR( dataPoints, p, varargin )
  include_globals

  if( nargin>2 )
    logScale = varargin{1};
```

```
  else
    logScale = 0;
  end;

  TI = dataParams.TI(dataParams.data_sequence);
  timeIndex = 0:.01:max(TI);

  if( isempty(p) )
    plotIt( logScale, TI, dataPoints, '.' );

  else
    % generate the fitted curve
    curve = siemensIR_fn( timeIndex, p );
    if( isempty(dataPoints) )
      plotIt( logScale, timeIndex, curve, '-' );
    else
      plotIt( logScale, TI, dataPoints, '.', timeIndex, curve, '-' );
    end;
  end; %if


% see s_t1_fun_ir.m
function ret = siemensIR_fn(t, p)
  % t is time, p is parameters
  global dataParams;

  TR = dataParams.TR;
  %ret = abs(A *   (1 - 2*     exp(-t/t1) +   exp(-TR/t1)));
  ret = abs(p(1) * (1 - 2*p(2)*exp(-t/p(3)) + exp(-TR/p(3))));
```

A.2.4 Image Alignment Functions

```
%ALIGNDATA  Align image matrix data.
%  IMG = ALIGNDATA( IMGBASE, IMG ) uses IMGBASE as a base to correct
%  the shift and rotation of IMG1.
%
%  IMG = ALIGNDATA( [], IMG, 'reuse' ) uses the the IMGBASE data from the
%  previous call of ALIGNDATA.  This is particularly useful when a sequence
%  of images needs to be aligned to the same base image.
%
%  After using ALIGNDATA on a set of images, it is a good idea to do 'clear
%  alignData' to clear the function's persistent variables.

% Bruce Po - igan@mit.edu            4/7/01
function imgi = alignData( img0, img1, varargin )
% ALIGNDATA

% Bruce Po - igan@mit.edu            4/7/01

  persistent blockSize isSmallImg xloc yloc HF HRF

  thStep = pi/256;

  disp( 'Begin image alignment -------------' );

  % If the 'reuse' option is NOT specified, then need to do pre-calculate
  % base image info.
  if( strmatch( 'reuse', varargin, 'exact' ))
    % do nothing
  else

    blockSize = 128;
    isSmallImg = 0;
    if( size(img0,1)<blockSize*3 )
      blockSize = size(img0,1)/4;
      isSmallImg = 1;
    elseif( size(img0,2)<blockSize*3 )
      blockSize = size(img0,2)/4;
      isSmallImg = 1;
    end;

    if( ~isSmallImg )
      tic
      % locate a good test block ------------------------------------------
      %  xloc, yloc :  top-left origin of test block
      %[xloc,yloc] = findInteresting2( img0, blockSize );
      [xloc,yloc] = findInteresting2( justEdges(img0), blockSize );
      disp( ['  locate a good test block: ' ...
```

```
          'xloc=' num2str(xloc) ' yloc=' num2str(yloc) ...
          ' [' num2str(toc) 's]'] );

    % for estimating translation ....................................
    block0 = img0(yloc:yloc+blockSize-1, xloc:xloc+blockSize-1);
  else
    block0 = img0;
  end;
  %block0 = fixHist(block0);
  H = block0 - mean(block0(:));           % zero mean of known signal
  H = rot90(H,2);                  % matched filter
  HF = fft2(H);

  tic
  % for estimating rotation ....................................
  if( ~isSmallImg )
    block00 = img0((yloc-blockSize):(yloc+2*blockSize-1), ...
                   (xloc-blockSize):(xloc+2*blockSize-1));
  else
    block00 = img0;
  end;
  %block00 = fixHist(block00);
  block0r = makePolarImg( block00, thStep );
  HR = block0r - mean(block0r(:));
  HR = rot90(HR,2);
  HRF = fft2(HR);
  disp( ['  polar matrix for estimating rotation: [' num2str(toc) 's]'] );

end; % reuse

% process new image ------------------------------------------------

imgi = img1;
th = 10000;  % non-zero dummy value

for iterations = 1:2

  if( th~=0 )
    tic
    % estimate translation of test block region ....................
    %  This will be the axis for estimating rotation.  (xs,ys)
    if( ~isSmallImg )
      blocki = imgi(yloc:yloc+blockSize-1, xloc:xloc+blockSize-1);
    else
      blocki = imgi;
    end;

    J = blocki - mean(blocki(:));          % zero mean
    JF = fft2(J);                          %
    VF = HF .* JF;                   %
    V = fftshift(ifft2(VF));               %
    [ys,xs] = max2(V);                     % peak of output is shift
    xs = xs - ceil(size(V,2)/2);
    ys = ys - ceil(size(V,1)/2);
    imgi = translateImg( imgi, -xs, -ys );
    disp( ['  estimate test block translation: ' ...
           '  xs=' num2str(xs) ' ys=' num2str(ys) ' [' num2str(toc) 's]'] );

    % estimate rotation ..........................................
    tic
    if( ~isSmallImg )
      blockii = imgi((yloc-blockSize):(yloc+2*blockSize-1), ...
                     (xloc-blockSize):(xloc+2*blockSize-1));
    else
      blockii = imgi;
    end;
    blockir = makePolarImg( blockii, thStep );
    J = blockir - mean(blockir(:));
    JF = fft2(J);
    VF = HRF .* JF;
    V = fftshift(ifft2(VF));
    [r,th] = max2(V);                      % peak of output is rotation
    th = th - ceil(size(V,2)/2);
    r = r - ceil(size(V,1)/2);
    if( r>3 | r<-3 )% sanity check
      r = 0;
      th = 0;
    end;
    disp( ['  estimate rotation: ' ...
           'th=' num2str(th) ' r=' num2str(r) ' [' num2str(toc) 's]'] );
```

```
      % align real image now ....................................
      if( th~=0 )
        tic
        if( ~isSmallImg )
          imgi = rotateImg( imgi, th*thStep, [xloc yloc] );
        else
          imgi = rotateImg( imgi, th*thStep );
        end;
        disp( ['  rotate image i: [' num2str(toc) 's]'] );
        % do a second pass of translation estimation ..................

        tic
        % estimate translation ....................................
        if( ~isSmallImg )
          blocki = imgi(yloc:yloc+blockSize-1, xloc:xloc+blockSize-1);
        else
          blocki = imgi;
        end;
        J = blocki - mean(blocki(:));        % zero mean
        JF = fft2(J);                        %
        VF = HF .* JF;                 %
        V = fftshift(ifft2(VF));             %
        [ys,xs] = max2(V);                   % peak of output is shift
        xs = xs - ceil(size(V,2)/2);
        ys = ys - ceil(size(V,1)/2);
        imgi = translateImg( imgi, -xs, -ys );
        disp( ['  estimate translation 2: ' ...
               'xs=' num2str(xs) ' ys=' num2str(ys) ' [' num2str(toc) 's]'] );

      end; %if th~=0
    end; %if th~=0

  end; %for

  if( strmatch('debug', varargin, 'exact' ))
    keyboard
  end;


function RI = makePolarImg( I, thStep )
% matrix using polar indexing

  xmin = 1;
  xmax = size(I,2);
  ymin = 1;
  ymax = size(I,1);
  x0 = round(xmax/2);
  y0 = round(ymax/2);

  thmin = -pi;
  %thstep = pi/256;    % .7031 deg resolution for 512 theta units
  thmax = +pi - thStep;
  rmin = 0;
  rstep = min(xmax,ymax)/2/32;        % 32 r units
  rmax = min(xmax,ymax)/2-1;

  RI = zeros( length(rmin:rstep:rmax), length(thmin:thStep:thmax) );
  xindices = zeros( length(rmin:rstep:rmax), length(thmin:thStep:thmax) );
  yindices = zeros( length(rmin:rstep:rmax), length(thmin:thStep:thmax) );

  ri = 1;              % r index
  for r = rmin:rstep:rmax
    thi = 1;           % theta index
    for th = thmin:thStep:thmax
      x = r * cos(th);
      y = r * sin(th);
      %try
        RI(ri,thi) = I(round(y)+y0,round(x)+x0);
      %catch
        %lasterr
        %keyboard
      %end;
      xindices(ri,thi) = x;
      yindices(ri,thi) = y;
      thi = thi + 1;
    end;
    ri = ri + 1;
  end;

function [i,j,v] = max2( M, varargin )
%MAX2 Find maximum in 2-D matrix.
```

```
%  Note: M should be all positive valued.
%
%  [I,J,V] = MAX2( M ) finds the element of 2-D matrix M with the
%  largest value.  I is the row, J is the column, and V is the
%  element's value.
%
%  [I,J,V] = MAX2( M, SEARCHRANGE ) searches only within a square region
%  centered about the origin whose size is specified by the scalar
%  SEARCHRANGE (+/-SEARCHRANGE from the origin).

% Bruce Po - igan@mit.edu - 4/7/01

  if( nargin>1 )
    minx = max( 0, round(size(M,2)/2)-varargin{1} );
    maxx = min( size(M,2)+1, round(size(M,2)/2)+varargin{1} );
    miny = max( 0, round(size(M,1)/2)-varargin{1} );
    maxy = min( size(M,1)+1, round(size(M,1)/2)+varargin{1} );
  else
    minx = 0;
    maxx = size(M,2)+1;
    miny = 0;
    maxy = size(M,1)+1;
  end;

  M = abs(M);
  M(:,1:minx) = 0;
  M(:,maxx:size(M,2)) = 0;
  M(1:miny,:) = 0;
  M(maxy:size(M,1),:) = 0;

  [vals ii] = max(M);
  [v j] = max(vals);
  i = ii(j);

function newImg = rotateImg( img, th, origin )
%ROTATEIMG
%
%  NEWIMG = ROTATEIMG( IMG, TH ) rotates image IMG an angle of TH
%  (radians) about the center of IMG.
%
%  NEWIMG = ROTATEIMG( IMG, TH, [X0 Y0] ) rotates IMG about the point
%  (x0,y0).

%  Bruce Po - igan@mit.edu - 4/7/01


  newImg = zeros(size(img));

  if( nargin==2 )
    x0 = round(size(newImg,2)/2);
    y0 = round(size(newImg,1)/2);
  else
    x0 = origin(1);
    y0 = origin(2);
  end;
  xsize = size(newImg,2);
  ysize = size(newImg,1);

  sin_th = sin(th);
  cos_th = cos(th);

  xmin = -x0;
  ymin = -y0;
  xmax = xsize-x0-1;
  ymax = ysize-y0-1;
  xrange = xmin:xmax;
  yrange = ymin:ymax;
  %xrange = xmin:(xmax-size(newImg,2)/2);
  %yrange = ymin:(ymax-size(newImg,1)/2);

  % set up LUTs so that we can remove error checking from loop
   xlimOffset = xsize;
   ylimOffset = ysize;
   a = ones(1,xlimOffset);
   b = a*xsize;
   c = ones(1,ylimOffset);
   d = c*ysize;
   xlimLUT = [a 1:xsize b];
   ylimLUT = [c 1:ysize d];

   xc1 = x0+1;
```

```
   yc1 = y0+1;
   xc2 = 1+x0+xlimOffset;
   yc2 = 1+y0+ylimOffset;

  for xb = xrange
    a = xb*cos_th;
    b = xb*sin_th;
    for yb = yrange
      xa = round(a - yb*sin_th);
      ya = round(b + yb*cos_th);
      x1 = xlimLUT(xa +xc2);
      y1 = ylimLUT(ya +yc2);
      x4 = xlimLUT(-xa +xc2);
      y4 = ylimLUT(-ya +yc2);
      newImg(yb +yc1, xb +xc1) = img(y1,x1);
      %newImg(-yb +y0, -xb +x0) = img(y4-1,x4-1);
    end;
  end;


function newImg = translateImg( img, tx, ty )
%TRANSLATEIMG Translates an image.
%
%   NEWIMG = TRANSLATEIMG( IMG, TX, TY ) translate the image IMG by TX
%   and TY, padding new space with 0's.

%  Bruce Po - igan@mit.edu - 4/7/01

  [ height, width ] = size(img);
  tempImg = zeros( height+abs(ty*2), width+abs(tx*2) );
  tempImg( abs(ty)+1+ty:abs(ty)+1+ty+height-1, ...
         abs(tx)+1+tx:abs(tx)+1+tx+width-1 ) = ...
      img;
  newImg = tempImg( abs(ty)+1:abs(ty)+1+height-1, ...
                  abs(tx)+1:abs(tx)+1+width-1 );

function V = justEdges( img )
%JUSTEDGES Simple edge enhancement.
%   IMG2 = JUSTEDGES( IMG )

% Bruce Po - igan@mit.edu - 4/7/01

  H1 = [-1 0 1 ; -2 0 2 ; -1 0 1];
  V1 = conv2( img, H1, 'same' );
  H2 = [1 2 1 ; 0 0 0 ; -1 -2 -1];
  V2 = conv2( img, H2, 'same' );
  V = sqrt(V1.*V1 + V2.*V2);

function [xloc,yloc,A] = findInteresting2( V, blockSize )
% FINDINTERSTING2 Returns the block with overall highest intensity pixels.
%
%   [XLOC,YLOC,A] = FINDINTERESTING2( V, BLOCKSIZE )
%
%   To find interesting featues, V should be an edges-only image.
%   xloc,yloc  the top-left origin of the interesting block
%   A          the matrix that represents the testing metric

  %if( size(V,2)<blockSize*3 | size(V,1)<blockSize*3 )
  %  % image is kind of small, so don't bother calculating blocks
  %  xloc = round(size(V,2)/2)-blockSize/2;
  %  yloc = round(size(V,1)/2)-blockSize/2;
  %  A = 1;
  %  return;
  %end;

  %m_index = 1:(fix(size(V,2)/blockSize));
  %n_index = 1:(fix(size(V,1)/blockSize));
  m_index = 2:(fix(size(V,2)/blockSize)-1);  % don't test the border
  n_index = 2:(fix(size(V,1)/blockSize)-1);  %  blocks
  A = zeros( length(n_index)+2, length(m_index)+2 );

  %x = m_index(1);
  for m = m_index
    %y = n_index(1);
    xrange = ((m-1)*blockSize+1):m*blockSize;
    for n = n_index
      %block = V(y:y+blockSize-1,x:x+blockSize-1);
      yrange = ((n-1)*blockSize+1):n*blockSize;
      block = V(yrange,xrange);
      A(n,m) = sum(block(:));
      %y = y+blockSize;
```

```
    end;
    %x = x+blockSize;
  end;

  [i,j] = max2(A);

  xloc = (j-1)*blockSize+1;
  yloc = (i-1)*blockSize+1;
```

A.3 Helper Functions

```
function [fileName,fileInfo] = dirFiles( pathn )
% DIRFILES Returns a character array listing of files and file sizes.
%
% [FILENAME,FILEINFO] = DIRFILES( PATHN )

  fileList = dir(pathn);
  fileName = '';
  fileInfo = '';
  for i = 1:length(fileList)
    if( ~fileList(i).isdir)
      fileName = strvcat( fileName, fileList(i).name );
      fileInfo = strvcat( fileInfo, [fileList(i).name '  ' ...
                  num2str(round(fileList(i).bytes/1024)) 'K'] );
    end;
  end;

function fig = dlgInfo( message, title, varargin )
%DLGINFO Info dialog box.
%  A dialog box with no buttons is displayed.  The calling program is
%  responsible for closing the dialog box.
%
%  H = DLGINFO( MESSAGE, TITLE ) displays a dialog box with width=140
%  pixels and height=40 pixels.
%
%  H = DLGINFO( MESSAGE, TITLE, [W H M] ) displays a dialog box with
%  width=w, height=h, and margin=m.

  if( nargin==2 )
    width = 140;
    height = 40;
    marg = 0;
  else
    a = varargin{1};
    width = a(1);
    height = a(2);
    marg = a(3);
  end

  a = get( 0, 'ScreenSize' );
  sc_width = a(3);
  sc_height = a(4);
  x = sc_width/2 - width/2;
  %y = sc_height-height-100;
  y = sc_height/2 - height/2;

  h0 = figure('Color',[0.8 0.8 0.8], ...
              'Units','pixels', ...
              'MenuBar','none', ...
              'Name', title, ...
              'NumberTitle', 'off', ...
              'Position',[x y width height], ...
              'Tag','Fig1', ...
              'Resize', 'off', ...
              'Visible','on');
  h2 = uicontrol('Parent',h0, ...
                 'Units','pixels', ...
                 'BackgroundColor', [0.8 0.8 0.8], ...
                 'ForegroundColor', [0.8 0.8 0.8], ...
                 'ListboxTop',0, ...
                 'Position',[0 0 width height], ...
                 'Style','frame', ...
                 'Tag','frm_1');
  h1 = uicontrol('Parent',h0, ...
                 'Units','pixels', ...
                 'BackgroundColor', [0.8 0.8 0.8], ...
                 'ListboxTop',0, ...
```

```
                'Position', [marg marg width-2*marg height-2*marg], ...
                'Style','text', ...
                'Tag','txt_info');

  set( h1, 'String', message );

  if nargout > 0, fig = h0; end

function ret = exportFig( h )
% EXPORTFIG Dialog box prompting user what format to save figure window to.
%
%   RET = EXPORTFIG( H ) first brings up a dialog box asking the which image
%   file format to use and then brings up standard file save dialog box.  H
%   is the figure handle of the window to be saved.  RET = 0 if the user
%   pushes Cancel, otherwise it's one.

  formatSwitch = { '-djpeg75', ...
                   '-djpeg90', ...
                   '-dtiff', ...
                   '-dpng', ...
                   '-dps', ...
                   '-dpsc', ...
                   '-dps2', ...
                   '-dpsc2', ...
                   '-deps', ...
                   '-depsc', ...
                   '-deps2', ...
                   '-depsc2', ...
                   '-dmfile' };
  formatMask = { '*.jpg;*.JPG', ...
                 '*.jpg;*.JPG', ...
                 '*.tif;*.tiff;*.TIF;*.TIFF', ...
                 '*.png;*.PNG', ...
                 '*.ps;*.PS', ...
                 '*.ps;*.PS', ...
                 '*.ps;*.PS', ...
                 '*.ps;*.PS', ...
                 '*.eps;*.EPS', ...
                 '*.eps;*.EPS', ...
                 '*.eps;*.EPS', ...
                 '*.eps;*.EPS', ...
                 '*.m;*.mat;*.M;*.MAT' };


  str = { 'JPEG 75% quality', ...
          'JPEG 90% quality', ...
          'TIFF', ...
          'PNG', ...
          'PostScript BW', ...
          'PostScript color', ...
          'Level 2 PostScript BW', ...
          'Level 2 PostScript color', ...
          'Encapsulated PostScript', ...
          'Encapsulated color PostScript',...
          'Encapsulated Level 2 PostScript', ...
          'Encapsulated Level 2 color PostScript', ...
          'Matlab figure' };

  [ selection ok ] = listdlg( 'PromptString','Select a file format:',...
                              'SelectionMode','single',...
                              'ListSize', [250, 200], ...
                              'Name', 'Save figure', ...
                              'ListString',str);



  if( ok )
    [filen, pathn] = uiputfile( formatMask{selection}, ...
                                ['Save ' str{selection}] );
    if( filen~=0 )
      saveFile = fullfile(pathn, filen);
      print( h, formatSwitch{selection}, saveFile );
    else
      ok = 0;
    end;
  end;

  ret = ok;

function fixResultsFile( filen )
```

```
  s = whos( '-file',filen );
  load( filen );

  % convert biospec to bruker
  magnet = sourceParams.magnet;
  if( strcmp(magnet(1:7),'biospec' ))
    sourceParams.magnet = ['bruker' magnet(8:length(magnet))]
  end;

  save( filen, s.name );

function count = fprintstring( fid, ss )
%FPRINTSTRING Writes a multi-line string, to a file.
%
%  FPRINTSTRING( FID, MSTRING ) writes the multi-line string to the file
%  associated with the specifier FID.

% Bruce Po - igan@mit.edu

  count = 0;
  for i = 1:size(ss,1)
    count = count + fprintf( fid, '%s\n', ss(i,:));
    %sprintf( '%s\n', ss(i,:));
  end;

function maxVal = getImgMax( img, varargin )
%GETIMGMAX Calculates a maximum value image displaying.
%
%   MAXVAL = GETIMGMAX( IMG ) returns a maximum intensity value of the image
%   IMG.  GETIMGMAX integrates the IMG's histogram to determine the value
%   MAXVAL where 95% of the pixels are less than or equal to MAXVAL.
%   Using MAXVAL for the maximum pixel value (see OVERLAYMAP) ensures
%   display with good contrast.  IMG is a matrix of non-negative pixel
%   values.
%
%   MAXVAL = GETIMGMAX( IMG, MAXPERCENT ) uses MAXPERCENT for the cutoff.
%   MAXPERCENT can range between .1 and 1.
%
%   See also OVERLAYMAP.

  if( nargin<2 )
    maxPercent = .95;
  else
    maxPercent = varargin{1};
  end;

  [i j v]= find(img);                % non-zero values only
  his = hist(v,100);                 % make 100 bin histogram
  binSize = max(v(:))/100;
  hisInt = cumsum(his);

  % find value where 95% of values are less than or equal to
  cutoff = (hisInt(100)-hisInt(1))*maxPercent+hisInt(1);
  a = find(hisInt<=round(cutoff));
  a = a(length(a));                  % index of bin where cutoff is
  maxVal = binSize*a;

function maxVal = getMapMax( theMap, varargin )
%GETMAPMAX Calculate a maximum value for T1 ot T2 map displaying.
%
%   MAXVAL = GETMAPMAX( THEMAP ) returns a maximum intensity value of the T1
%   or T2 map suitable for displaying using OVERLAYMAP.
%
%   See also OVERLAYMAP.

%maxVal = round(1.2 * max(theMap(:)));       % naive method

  % this section is the same as getImgMax.m ........................
  if( nargin<2 )
    maxPercent = .98;                % default cut-off value
  else
    maxPercent = varargin{1};
  end;

  [i j v]= find(theMap);             % non-zero values only
  his = hist(v,100);                 % make 100 bin histogram
  binSize = max(v(:))/100;
  hisInt = cumsum(his);
  cutoff = hisInt(100)*maxPercent; % find value where maxPercent of values
                                      % are less than or equal to
  a = find(hisInt<=round(cutoff));
```

```
  a = a(length(a));                  % index of bin where cutoff is
  maxVal = binSize*a;
  % ..................................................................

  maxVal = maxVal * 1.2;             % add an upper cushion

function t = gettimeaxis
% GETTIMEAXIS Returns an array with the acquisition time parameter.

  global sourceParams dataParams

  switch( sourceParams.magnet )
   case { 'brukerIR', 'siemensIR', 'geIR' }
    t = dataParams.TI;
   case 'brukerSR'
    t = dataParams.TR;
   case 'brukerT2'
    t = dataParams.TR;
  end;
function [y,x] = getyx( i )
% GETYX Converts linear index to row,col
%
%   [Y,X] = GETYX(I)

  global dataParams;

  msize = dataParams.msize;
  y = mod( i, msize(1));
  if( y==0 ) y = msize(1); end;
  x = floor((i-1)/msize(1))+1;
function ret = inputFieldDlg( theStruct, fieldname, varargin )
% INPUTFIELDDLG Get user input for a structure's field.

% Bruce Po - igan@mit.edu


  % check options ---------------------------------------------------
  if( nargin>2 )

    % confrimation option
    if( strcmp(varargin{1}, 'confirm') )
      a = questdlg( [fieldname '''s value is automatically set. ' ...
                     ' Are you sure you want to manually modify' ...
                     ' this field?'], ...
                    'Confirm', 'Yes', 'Cancel', 'Cancel' );
      if( strcmp(a, 'Cancel') )
        ret = 0;                     % flag to indicate canceled
        return;
      end;
    end;
  end;
  if( nargin>3 )
    prompts = varargin{2};
  else
    prompts = '';
  end;

  % get user input for new field value ------------------------------
  value = getfield( theStruct, char(fieldname) );
  value_info = whos( 'value' );
  value = num2str(value);
  if( isfield( prompts, fieldname ))
    prompt = getfield( prompts, fieldname );
  else
    prompt = ['Enter value for ' fieldname ':'];
  end;
  newValue = inputdlg( prompt, fieldname, 1, {value}, 'off' );
  if( ~isempty(newValue) )
    newValue = newValue{1};
    switch( value_info.class )
     case 'char'
      theStruct = setfield( theStruct, char(fieldname), newValue );
     case 'double'
      if( strcmp( newValue, 'auto' ))
        theStruct = setfield( theStruct, char(fieldname), ...
                                         newValue );
      else
        theStruct = setfield( theStruct, char(fieldname), ...
                                         str2num(newValue) );
      end;
    end
```

```
    ret = theStruct;
  else ret = 0;                         % flag to indicate canceled
  end


function ret = listSiemensNums( directory, fileMask, whichPart )
% Lists file directory and use that to extract some info.  For Siemens
% magnet,format is ????-??-??.ima.  The number of digits doesn't have to be
% fixed.

  % Make sure that directy ends with '/'!

  if nargin==0,
    directory = '.';
  end
  dirList = dir( [directory fileMask ] );
  if isempty(dirList), ret='';
  end

  newList = '';
  % Go through list of files and pick out info
  for ii=1:size(dirList,1);   % number of rows
    temp = '';
    dashLoc = findstr( dirList(ii).name, '-' );
    if strcmp( whichPart, 'patientID' ),
      dashLoc = findstr( dirList(ii).name, '-' );
      dashLoc = dashLoc(1);
      temp = dirList(ii).name(1:(dashLoc-1));       % find patient ID
    elseif strcmp( whichPart, 'studyNum' ),
      beginLoc = dashLoc( 1 );
      endLoc = dashLoc( 2 );
      temp = dirList(ii).name(beginLoc+1:endLoc-1);
    elseif strcmp( whichPart, 'imageNum' ),
      beginLoc = dashLoc( 2 );
      a = findstr( dirList(ii).name,'.' );
      endLoc = a(1);
      temp = dirList(ii).name(beginLoc+1:endLoc-1);
    end
    newList = strvcat( newList, temp );
  end

  % sort the list

  newList = strjust( newList, 'right' );
  newList = sortrows( newList );

  % remove redundant entries and set up return lists
  if length(newList)==0 ,
    newList2 = '';
  else
    newList2 = newList(1,:);
    if length(newList)>1 ,
      for ii=2:size( newList,1 ),     % number of rows
        if ~ strcmp( newList(ii-1,:), newList(ii,:) ),
          newList2 = [ newList2 ; newList(ii,:) ];
        end
      end
    end
  end

  ret = newList2;

function overlaycolorbar( varargin )
%OVERLAYCOLORBAR puts a color bar on a T1 map overlay image, using the
%appropriate T1 map scale factor.
%
%   OVERLAYCOLORBAR puts a colorbar on the current figure.
%
%   OVERLAYCOLORBAR(H) puts a colorbar on the specifed figure.
%
%   See also OVERLAYMAP.

%  Bruce Po - igan@mit.edu

  if( nargin==0 )
    h = gcf;
  else
    h = varargin{1};
  end;
  figure( h );
  colorbar;
```

```
  cbScale = get( h, 'Userdata' );
  if( isempty(cbScale) )
    cbScale = 1;
  end;

  ah = colorbar;
  set( ah, 'YTicklabel', strvcat( num2str( 1/cbScale*50,3 ), ...
                                  num2str( 1/cbScale*100,3 ), ...
                                  num2str( 1/cbScale*150,3 ), ...
                                  num2str( 1/cbScale*200,3 ), ...
                                  num2str( 1/cbScale*250,3 )));
                                      function placeFig( h, wherey, wherex )
% PLACEFIG Places an existing figure somewhere on the screen.
%
%    PLACEFIG( H, WHEREY, WHEREX ) places figure H at location specifed by
%    WHEREY and WHEREX.  Possible values for WHEREY are 'top', 'bottom',
%    'center'.  Possible values for WHEREX are 'left', 'right', 'center'.
%

  % get figure's dimensions
  saveUnits = get( h, 'Units' );
  set( h, 'Units', 'Pixels' );
  figPos = get( h, 'Position' );
  width = figPos(3);
  height = figPos(4);

  % get screen dimensions
  a = get( 0, 'ScreenSize' );
  sc_width = a(3);
  sc_height = a(4);

  switch( wherex )
   case 'left'
    x = 20;
   case 'right'
    x = sc_width - width-20;
   case 'center'
    x = sc_width/2 - width/2;
  end

  switch( wherey )
   case 'center'
    y = sc_height/2 - height/2;
   case 'top'
    y = sc_height - height - 40;
   case 'bottom'
    y = 60;
  end;

  set( h, 'Position', [x y width height] );
  set( h, 'Units', saveUnits );

function plotIt( logScale, varargin )
%PLOTIT Plots a graph in regular or log scale.
%  PLOTIT( 0, ... ) does PLOT( ... ).
%  PLOTIT( 1, ... ) does SEMILOGY( ... ).

  if( logScale )
    semilogy( varargin{:} );
  else
    plot( varargin{:} );
  end;
function setupUI( fh, callerFun )
% SETUPUI Sets up the callbacks in the UI figure.
%
%   SETUPUI( FH, CALLER_FUN ) sets up the callback function for each UI
%   control in figure specified by figure handle FH.  The callback function
%   is CALLER_FUN(tag); where tag is the tag of the UI control.


  set( fh, 'CloseRequestFcn', [callerFun '(''close'');']);
  objList = get( fh, 'Children' );
  for i=1:length(objList)
    obj = objList(i);
    if( ~strcmp(get(obj,'Type'),'axes') )
      set( obj, 'Callback',...
               [callerFun '(''' get(obj,'Tag') ''');'] );
    end;
  end;
```

```
function ret = struct2string( theStruct )
%STRUCT2STRING Converts structure to character matrix suitable for Matlab
%   X = STRUCT2STRING( S ) converts the structure S into a string array
%   suitable for displaying.  Each line in X contains a field name, a
%   colon (:), and the value of that field.

% Bruce Po - igan@mit.edu


  fields = strvcat(fieldnames(theStruct));
  vals = '';
  for i=1:size(fields,1)
    value = getfield( theStruct, char(fields(i,:)) );
    value_info = whos( 'value' );     % figure out data type of this
                                      % field
    if( isempty( value ) ), value = ' '; end;
    switch( value_info.class )
     case 'char'
      value_val = value;
     case 'double'
      value_val = num2str( value );
    end
    temp = value_val;
    value_val = deblank(temp(1,:));
    for i = 2:value_info.size(1)
      value_val = [value_val ', ' deblank(temp(i,:))];
    end;
    vals = strvcat( vals, [' : ' value_val] );
  end
  ret = [fields vals];

function s = structcombine( s1, s2, varargin )
%STRUCTCOMBINE Combines two structures together.
%
%   S = STRUCTCOMBINE( S1, S2 ) combines the structures S1 and S2.  The
%   structure S will have all the fields from both S1 and S2.  If a field is
%   present in both S1 and S2, the value from S2 is preserved.
%
%   S = STRUCTCOMBINE( S1, S2, FLAG ) copies only the fields from S2 that
%   are also present in S1 when FLAG=1.  S2 fields that are not present in
%   S1 are discarded.  If a field is present in both S1 and S2, the value
%   from S2 is preserved.

% Bruce Po - igan@mit.edu

  if( nargin>2 )
    flag = varargin{1};
  else
    flag = 0;
  end;

  s.void = '';

  fields = fieldnames( s1 );
  for i = 1:length(fields)
    s = setfield( s, fields{i}, getfield(s1, fields{i}) );
  end;

  fields = fieldnames( s2 );
  for i = 1:length(fields)
    if( ~flag | (flag & isfield(s1,fields{i})) )
      s = setfield( s, fields{i}, getfield(s2, fields{i}) );
    end;
  end;

  s = rmfield( s, 'void' );

function A = unionall( varargin )
%UNIONALL( A1, A2, ... ) Union of two or more sets.
%   The number of sets can be arbitrary.
%
%   See also UNION.

% Bruce Po - igan@mit.edu

  if( nargin==0 )
    A = '';
  elseif( nargin==1 )
    A = varargin{1};
  else
    A = union(varargin{1}, varargin{2});
```

```
end;

n = 3;
while( n<=nargin )
  A = union( A, varargin{n} );
  n = n+1;
end;
```