Graphical Interface for Quantitative Monitoring of 3D MRI Data

by

Meredith L. Gerber

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical [Computer] Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

June 8, 2005
[September 2005]

Author_____
Department of Electrical Engineering and Computer Science
June 8, 2005

Certified by_____
Dr. Deborah Burstein
Thesis Supervisor

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

Graphical Interface for Quantitative Monitoring of 3D MRI Data
by
Meredith L. Gerber

Submitted to the
Department of Electrical Engineering and Computer Science

June 8, 2005

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical [Computer] Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

The recent development of techniques in magnetic resonance imaging allows for the noninvasive monitoring of cartilage for disease progression, effects of lifestyle change, and results of medical interventions. In particular, the dGEMRIC technique has been used. Prior dGEMRIC data have been two-dimensional. Magnetic resonance equipment can currently produce three-dimensional dGEMRIC data, but software that analyzes three-dimensional data sets is lacking in practicality. This research improved existing software to better handle three-dimensional dGEMRIC data sets. Improvements were made to better facilitate (1) image section selection, (2) segmentation, (3) T1 mapping, and (4) statistical data analysis.

Thesis Supervisor: Deborah Burstein
Title: Associate Professor of Radiology, Beth Israel Deaconess Medical Center, Harvard Medical School

Contents

1. Introduction

Osteoarthritis is a disease of the cartilage that affects nearly 21 million people in the United States (Arthritis Foundation 2004). Current imaging techniques in standard use are based on x-ray radiography and provide information only on the bone in the areas near the cartilage. Monitoring cartilage for biochemical abnormalities and changes at the beginning of symptom onset may allow patients and their doctors to enact sufficient preventative measures to slow or prevent the development of osteoarthritis.

1.1 delayed Gadolinium Enhanced MRI of Cartilage (dGEMRIC)

It has been demonstrated that magnetic resonance imaging can be used to monitor biochemical features of cartilage using the delayed Gadolinium Enhanced MRI of Cartilage (dGEMRIC) technique (Bashir, Gray, Burstein. 1996). This technique is designed to image the glycosaminoglycan (GAG) component of cartilage. The GAG component confers much of the tissue's mechanical strength due to the electrostatic repulsion of abundant negatively charged carboxyl and sulfate side-groups on GAG. The negative charge on the GAG molecule is also the basis of the dGEMRIC technique. The dGEMRIC protocol relies on the contrast agent $Gd(DTPA)^{2-}$ (Gadolinium diethylenetriaminepentaacetic acid). In vivo, the charged contrast agent distributes itself throughout the cartilage after intravenous injection. (In vitro, the cartilage sample is bathed in contrast agent.) Lower concentrations of the contrast agent are found in areas of the cartilage that have more GAG, just as there would be lower concentrations of any mobile negative charge carrier in an area where there is abundant fixed negative charge.

The distribution of the contrast agent $Gd(DTPA)^{2-}$ in the cartilage can be visualized using T1 parameters in magnetic resonance imaging. Bashir et al showed that T1 value changes correlated with changes in GAG concentration in young bovine cartilage samples in the presence of gadolinium. There was no significant change in T1 values when gadolinium was not present, even in the case where there was complete GAG loss (Bashir, Gray, Burstein. 1996). Bashir et al reported that in vivo T1 values, and in particular inversion-recovery T1-weighted values differ significantly for regions of high and low GAG concentration in gadolinium-penetrated knee cartilage (Bashir, Gray et al. 1997).

By imaging the GAG concentration of cartilage in vivo, the dGEMRIC technique reflects the biochemical status of the cartilage and can be used to track disease progression, effects of lifestyle change, and results of medical interventions.

1.2 Two-Dimensional and Three-Dimensional dGEMRIC Data

When the dGEMRIC technique was first developed, the protocol dictated acquisition of single slice 2D magnetic resonance data. The need for 3D analysis is highlighted by Williams et al. In analyses on two-dimensional slices, Williams et al demonstrated that low dGEMRIC indices (indicating low GAG concentration) can occur in general, compartmental, or focal regions (Williams, Gillis et al. 2004). When performing analyses on individual two-dimensional slices, it is difficult or impossible to locate all focal dGEMRIC lesions and other features of interest. Each set of two-dimensional dGEMRIC data that is acquired and analyzed can only address a very limited volume of the knee cartilage. It is not clear that the data for this limited volume can be taken to be representative of a larger section of the knee. Further investigation is

necessary to determine when, if ever, it is the case that data gathered from one or a small set of two-dimensional slices accurately reflects the health of the entire knee.

Efficient magnetic resonance pulse sequences have been developed such that time is no longer prohibitive in taking three dimensional dGEMRIC magnetic resonance data. Because it is now feasible to acquire three-dimensional dGEMRIC data, the capability to analyze this data is needed to easily gain an accurate picture of the health of knee cartilage. Two-dimensional dGEMRIC data has been in use longer, and there are established methods of analysis for this kind of data. Because three-dimensional data can be considered as a series of two-dimensional slices, subsets of three-dimensional data can be analyzed as though they are two-dimensional data sets.

Currently, software that visualizes dGEMRIC maps calculated from three-dimensional data sets is lacking in practicality, and as a result, a large percentage of the information is not utilized. Additionally, the potential volume of data that could be acquired exceeds reasonable time constraints for manual analysis. It should be investigated, when it might be the case that smaller subsets of data can be taken as representative of larger volumes.

1.3 dGEMRIC Data Analysis

Analysis of dGEMRIC data requires four primary steps: (1) image section selection, (2) segmentation, (3) T1 mapping, and (4) statistical data analysis. Magnetic Resonance Image data is digital and all four of these steps are carried out with the use of computer software.

All four of these steps are necessary regardless of the kind of study being performed. However, different kinds of studies may require users to spend proportionally

different amounts of times during the particular analysis steps. Two kinds of studies that are mostly representative of the spectrum are the individual case study and the clinical trial.

In a typical case study, one patient would submit to an MRI scan at multiple time points. Researchers may be interested in tracking specific areas of cartilage across the time points. Researchers performing case studies may be willing to invest more time looking at the entire knee cartilage, as opposed to only the areas that they suspect receive the most impact or the areas that suspect will yield the most interesting data. Researchers may locate all suspected lesions throughout the cartilage and track their progression over time.

Because there is a lower volume of data taken as part of a case study (as compared to a clinical trial), more time can be spent analyzing each individual slice involved in the study. Researchers may be more attentive to detail when analyzing case study data. They may choose to segment the data into a greater number of regions of interest and run many kinds of statistical tests on the data that is collected. For a user analyzing case study data, segmentation ease and statistical data analysis flexibility are important.

A clinical trial, in contrast, may involve hundreds or thousands of patients at multiple data acquisition sites. Data for several follow-up time points may be acquired. The large amount of data places unique requirements on the data analysis methods. Time is finite, so efficient analysis is desired. Additionally, in order to be confident in the validity of the data set as a whole, the analysis needs to be consistent from patient to patient and from time point to time point. Typically in a multi-site study, all of the data is sent to one location for analysis. If this is not the case, there is an added complication. In

the case of a clinical trial, the procedures for segmentation and statistical analyses are determined ahead of time and not likely to change much, if at all. For a user analyzing clinical trial data, speed and efficiency are important throughout all steps of the analysis, due to the large volume of data being processed. This includes the first step, image section selection. An analyst for a case study, in contrast, may not mind spending the time poring over each slice. Additionally, this means that the clinical trial analyst values speed and efficiency in statistical data analysis over the flexibility valued by the case study analyst.

Software used to analyze three-dimensional dGEMRIC data sets should strike a compromise in meeting the needs of analysts of both the case study and the clinical trial (and thereby meet the needs of all those analysts for studies in-between these two kinds). The software should have a graphical user interface, so that it is accessible to a wide range of researchers and clinicians who may not have a high level of comfort with the software. The software should facilitate all four steps that compose the process of dGEMRIC image analysis.

1.4 Project Summary

In this thesis project, I have set out to create a piece of software that facilitates the analysis of three-dimensional dGEMRIC images. I have evaluated several software packages currently used for medical image analysis and chosen one to further customize for the specific use of three-dimensional dGEMRIC data analysis.

I have taken existing software that created dGEMRIC maps for two-dimensional data, and I have expanded its capabilities and improved its application to 3D image data sets. I have made improvements to the software pertaining to each of the four primary

steps in addition to making some minor improvements to the general user interface. For the first step, image section selection, interfaces have been created which allow for the user to browse through the entire 3D data set and choose a section of interest for further analysis. For the second step, segmentation, functionality was added to allow the user to view comparison images, which can serve as segmentation guides, in the analysis software, reducing the need to use an additional software program or refer to an alternate media source. For the third step, T1 mapping, the ability to fit data acquired with a recently developed data acquisition/fitting method was added. And for the fourth step, statistical data analysis, functionality was added to facilitate automated export of the data to Excel worksheets (Microsoft, Redmond, WA), avoiding the necessity for the user to hand-enter large data sets into the data processing software. Additionally, functionality was added to facilitate region of interest labeling; these labels are exported to Excel worksheets with the data.

2. Background

Previously, two-dimensional dGEMRIC data were acquired, and they were analyzed using MRIMapper (Po, 2001), software created for magnetic resonance image analysis on MATLab 5.3.

2.1 Current (2D) dGEMRIC Analysis Methods

In the case of two-dimensional dGEMRIC data, the first step of data analysis, image section selection is performed at the time of data acquisition, when the two-dimensional dGEMRIC data are acquired. Typically, one data set is acquired on a knee.

If time is less of a constraint, one data set may be acquired for each of the two condyles in the knee. (More data may be desired, but time restrictions limit the amount of data that can be acquired.) A data set consists of a group of five files all corresponding to the same physical location, a particular area in a particular two-dimensional plane in the knee. Data are commonly acquired in either the sagittal or coronal plane. Each file holds a T1-weighted MR image that was acquired with a different inversion time. The five files are analyzed together on a pixel-by-pixel basis with a 3-parameter T1-fit. This is the T1-mapping step of dGEMRIC data analysis, and the result is a dGEMRIC value for each pixel that has been analyzed. Typically, the dGEMRIC value is mapped to a color scale and overlaid on a grayscale background image.

The second step of dGEMRIC data analysis, segmentation, is performed in order to limit the time the computer takes to fit the data and for the purpose of focusing user attention. Segmentation is the process of defining regions of interest in the image. This can be performed either before or after performing the T1 fit. If segmentation is done after the T1 fit, no processing time is saved, since the software tries to fit all of the pixels in the image. The regions of interest are subgroups of the set of pixels that correspond to cartilage in the patient. Different regions of interest may be chosen, depending on the kind of investigation being performed. If data were taken in the sagittal plane, some of the regions of interest might include: all visible cartilage, all femoral cartilage, all tibial cartilage, central femoral cartilage, anterior femoral cartilage, and posterior femoral cartilage. If data were taken in the coronal plane, some of the regions of interest might be different: medial femoral cartilage, medial tibial cartilage, lateral femoral cartilage, and lateral tibial cartilage. Because one slice of dGEMRIC data is available for either the knee or the condyle, that data is taken as representative of the whole region.

Prior to recording statistical data on the regions of interest that were fitted, the fits are checked for quality. Fit quality may be determined by fit error. In some cases, fit quality is determined to be poor if the resulting dGEMRIC value is not within the expected physiologic range. Pixels with bad fits may be excluded from the subsequent analysis.

The fourth step of dGEMRIC data analysis is statistical data analysis. Data are most typically analyzed in bulk; that is, for the selected region, an average dGEMRIC index is reported. The standard deviation of the region is often noted but not often reported. The area of the region of interest is quantified by the number of pixels that it includes. This value may be used in a quantitative analysis, but it is more frequently demonstrated qualitatively through images or notes on the quality or health of the cartilage.

When data sets are being compared to each other, or treated as a group and compared to other groups, statistical analyses are performed on the quantitative information recorded from each region of interest or from subgroups of regions of interest.

Results are reported either visually or statistically. Results are conveyed visually through images of the color-mapped region of interest. Quantitative results are reported statistically through either tables or plots.

2.2 Current (2D) dGEMRIC Analysis Software: MRIMapper

MRIMapper is software has been developed to perform and visualize quantitative analyses on two-dimensional cartilage MR, including T1-weighted inversion recovery, data (Po 2001). The software runs in MATLab 5.3 (The MathWorks, Natick, MA) via a

graphical user interface (see Figure 1). This MRIMapper software facilitates dGEMRIC image analysis, and will be described in terms of the four steps of dGEMRIC data analysis: (1) image section selection, (2) segmentation, (3) T1 mapping, and (4) statistical data analysis.

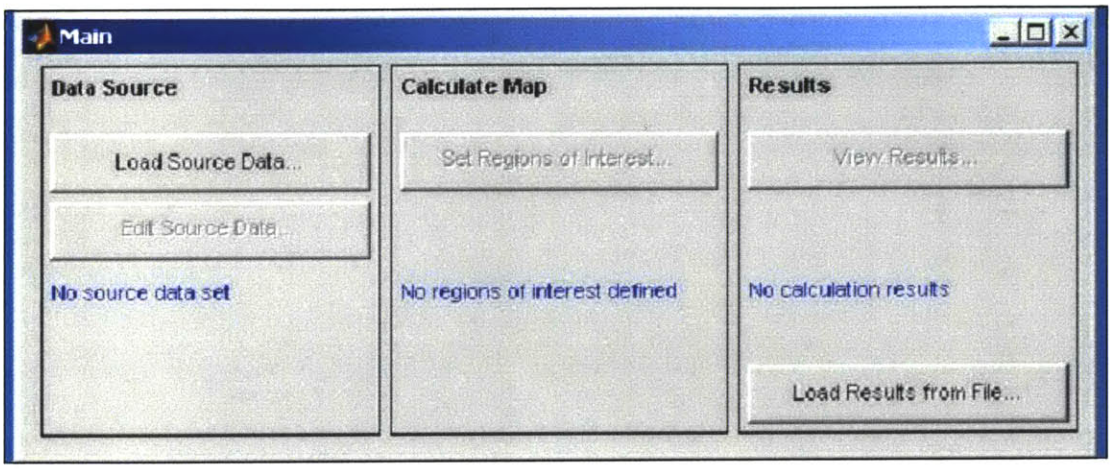

Figure 1: MRIMapper Main interface

2.2.1 Image Section Selection

When the user first opens the MRIMapper software, there are two available functionalities (see Figure 1). Both of the available buttons, 'Load Source Data', and 'Load Results from File…' permit the user to browse the file structure of the computer and load either previously mapped or raw data. Data are selected based on file name only. If the user wishes to select data based on the image itself, the user may open some files in an image-viewing software package or may view previously made printouts. In the case of loading raw data, once the data are selected, the user must refer to documentation from data acquisition and enter the proper image parameters before the user can proceed to segmentation. Using the software to load raw data via the 'Load

Source Data' button is described below. Loading previously mapped data is a similar process, but data acquisition parameters do not need to be entered.

To Load raw data, the user activates the 'Load Source Data button, and via this button, the user is led to the data source interface (see Figure 2). In this interface, the user enters pertinent information about the data that will be analyzed. This includes the computer directory and a list of files (in that directory) to be fit, the manufacturer of the machine that acquired the data, the data spatial resolution, and the repetition time, inversion time, and echo time, where appropriate to the kind of fit.

This information is primarily data acquisition parameters that are important to making a proper fit. The items on this image that are circled in red (source machine manufacturer, data fit type, and repetition and inversion times) are the parameters that are most frequently changed or adjusted. In the case of the pull-down menus, these are adjusted every time this interface is accessed, because their default values are the first values in their respective lists, and it is this default value that is shown whenever the interface is opened. In the case of repetition and inversion times, these values differ from data set to data set, although they do remain the same within most studies. In the case of dGEMRIC data, there is a discrepancy between the way that repetition time data is stored with the MR image and the way that it needs to be input into the software. The difference is along the lines of a linear transformation. In most cases, the user will note the repetition times reported in the header of the MR image and will perform the transformation on paper or in Excel. Barring discrepancies in units, the user can typically type in the inversion time values that are seen in the image header information. The remaining fields in the Data Source window are either typically the same as the default values or input automatically by the MRIMapper software. The parameters that are

determined by the software are no_of_TI, which reports the length of the TI vector, and data_sequence, which generates a vector of the whole numbers from one through no_of_TI.

An additional key piece of information that is entered in the Data Source interface is file location information. By clicking on the 'Select files...' button, the user can browse for the desired files via the graphical interface shown in Figure 3. This is where the first step of dGEMRIC data analysis occurs. In choosing a specific set of files to load, the user has selected a slice from a particular anatomical location for analysis. Prior to slice selection, users may choose to look over different slices in a three-dimensional data set with the use of another software package such as ImageJ (NIH, Bethesda, MD) or eFILM (Merge eFILM, Milwaukee, WI).

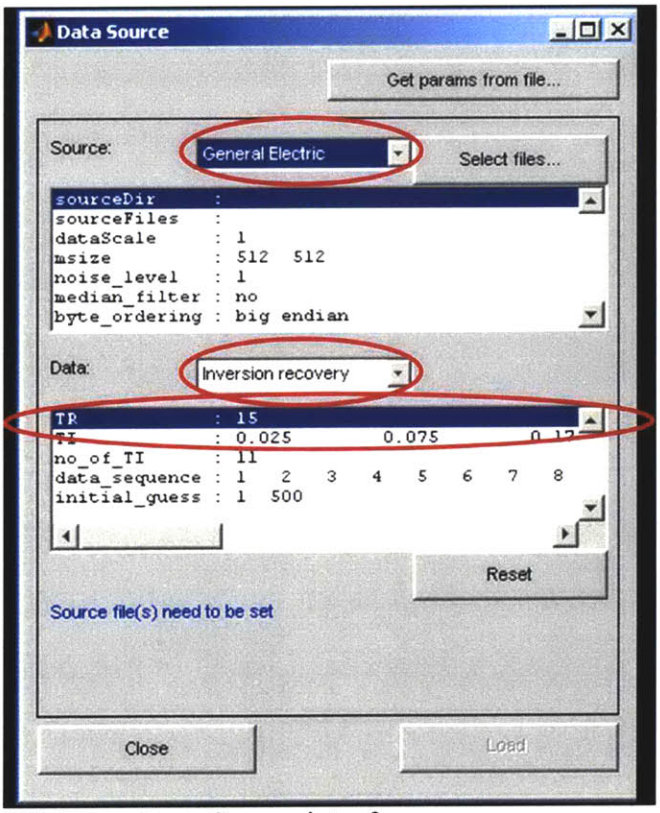

Figure 2: MRIMapper Data Source interface

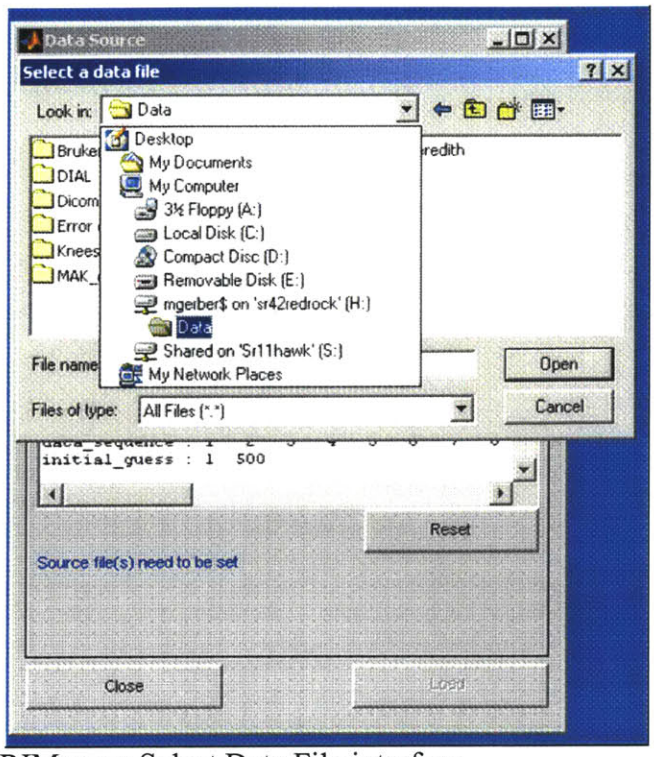

Figure 3: MRIMapper Select Data File interface

After all necessary parameters are specified, the 'Load' button in the Data Source window becomes enabled. When the user presses the 'Load' button, the data files and parameters are loaded into the workspace global variables. At this point, the program brings up an image of the slice that has been loaded. At this point, the user can choose between two main tasks, each accessible by buttons in the Main graphical interface (see Figure 1). Although this may be skipped (usually in the interest of time) the user may choose to perform some data quality control. Clicking on the 'Edit Source Data' button in the main window (see Figure 1) brings up an interface where the user can ask the software to align all or any subset of the group of images that had been loaded. Typically, the image with the longest inversion time is excluded from this alignment because it is so dark that there are few distinct features from which to align. Before deciding whether or not to align, the user can also use this interface to view the five

images subsequently or as a movie. The user may choose to skip this step the first time the data are analyzed and may choose to come back and improve the data quality later if the fits seem to be off or if the data looks interesting and might be shared or published.

2.2.2 Segmentation

At this point, the user may proceed to the second step of dGEMRIC image analysis, segmentation. This is the most effort-intensive step of the process. The contrast between the cartilage and other tissues in the knee is not as good in T1 images as it is in other MR images, so outlining the areas of cartilage is a challenge for the user. The process of segmenting in MRIMapper is described in detail below. Many users choose to use another image-viewing software simultaneously to segmenting in MRIMapper so that an image with better contrast can be viewed to guide segmentation.

To access the interface that facilitates segmentation, the user first presses the 'Set Regions of Interest' button in the Main window (see Figure 1). This brings up the window shown in Figure 4. In the case where no regions of interest have yet been defined, the image will be a black and white, and there will be no ROI listed in the text box in the Calculation Results window. To access the interface that facilitates segmentation, the user now presses the 'Add ROI' button in the Calculation Results window. This brings up the 'Regions of Interest' interface shown in Figure 5.

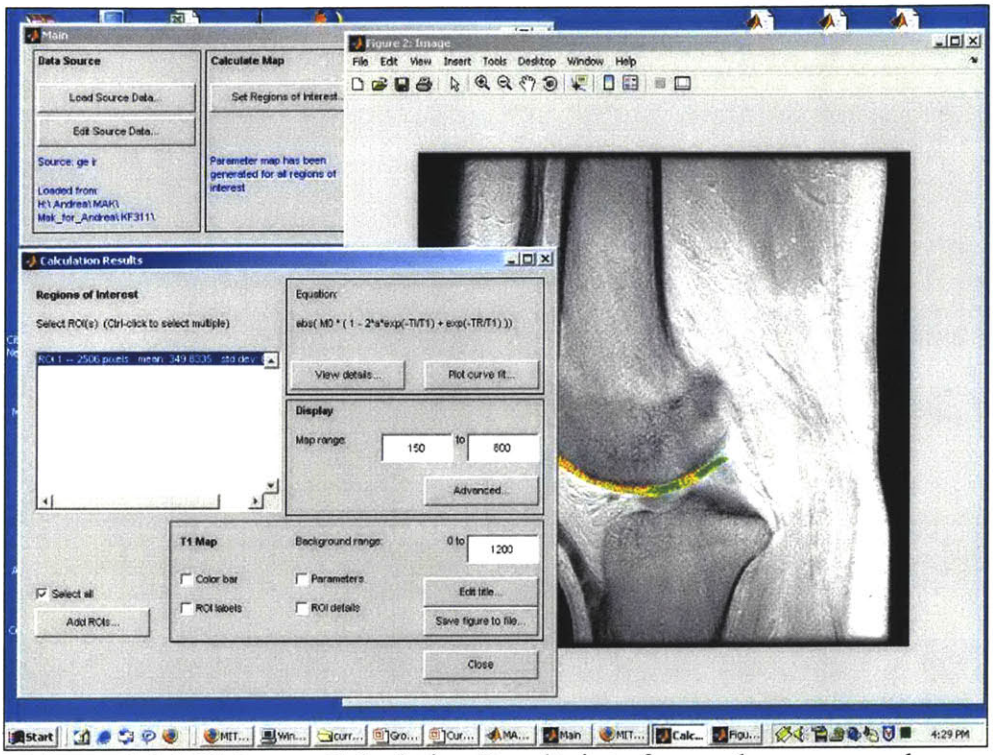

Figure 4: MRIMapper Calculation Results interface and map example.

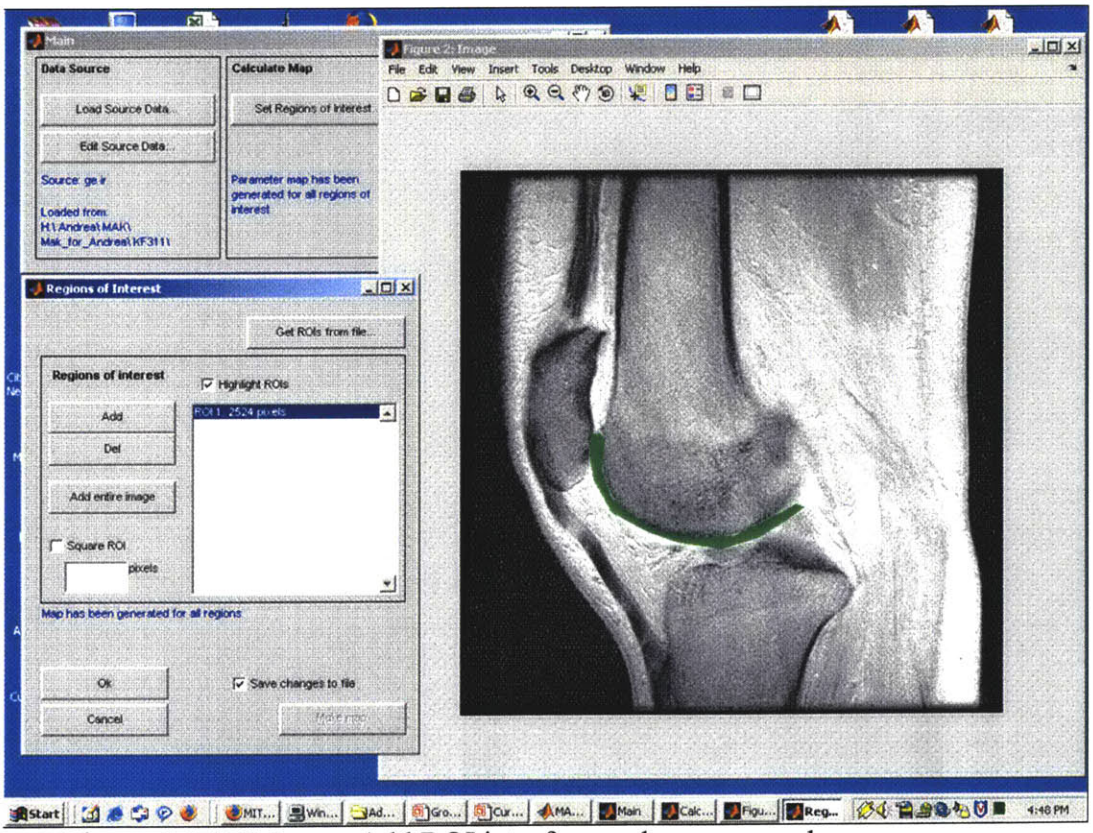

Figure 5: MRIMapper Add ROI interface and map example.

To add an ROI, the user presses the 'Add' button in the 'Regions of Interest' window, and traces the cursor over the desired region of interest, making a single click at each desired vertex. An illustration of this process is shown in Figure 6. Regions of interest that have already been defined are shown in green. The boundaries of the region of interest being currently defined are the black and white dashed lines. There is no limit to the number of ROIs that can be defined here, and users may try several times to trace more accurately the same target area. Highlighting the undesired regions of interest in the text box list in the Regions of Interest window and pressing the 'Del' button beneath the 'Add' button on the left-hand side can discard regions of interest that the user does not want to be mapped at this point. Once all of the desired regions of interest are defined, the user continues to the third step of dGEMRIC image analysis, T1 Mapping, by pressing the 'Make Map' button in the Regions of Interest window.

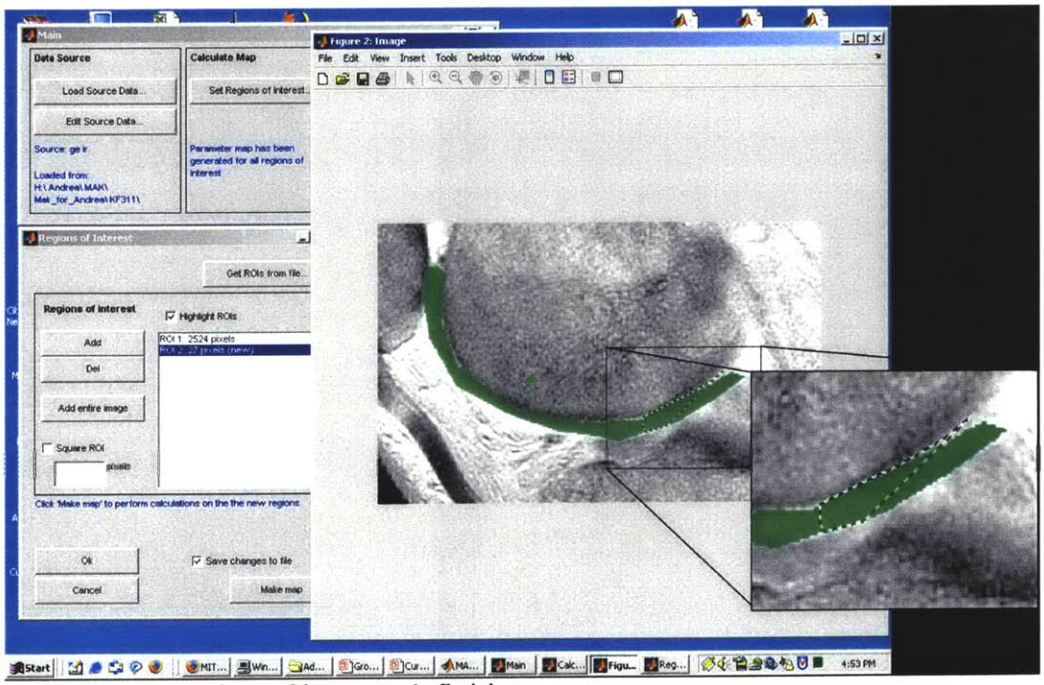

Figure 6: Region of interest definition.

Regions of interest are numbered as they are created and the correspondence between the number label and the anatomical region is clear only when the user can interact with the image via the MRIMapper software, asking it to display specified regions of interest. After regions of interest are mapped (the next step of dGEMRIC image analysis), the data are ready for data analysis (the fourth and final step of dGEMRIC image analysis). At this point, the user carefully notes the anatomical structure that each region of interest corresponds to as the user copies the compiles the basic statistics for each ROI from multiple image sections or subjects.

2.2.3 T1 Mapping

The mapping step of dGEMRIC image analysis is pretty straightforward and some aspects of this task are described below. The type of fit that will be performed on the data is determined when the user selects the kind of data from the Data pull-down menu in the Data Source window that can be seen above in Figure 2.

In the case of dGEMRIC analysis, a T1 fit is performed across the five input images for each of the pixels in any of the regions of interest that are defined. It is possible to define new regions of interest after a mapping operation has already been performed, and if this is the case, the same pixel is not mapped twice. The data from the previous mapping are retained, and only the new pixels are mapped. The T1-fitting step is the singular most computationally intensive step in the dGEMRIC data analysis process.

Once the fit is made pixel-by-pixel, the resulting values are visualized in a map overlay on top of one of the images (map_overlay.jpg). In the map overlay, each pixel in a region of interest is displayed in a color corresponding to the fit value for that pixel.

The background image, previously chosen by the user, is shown in black and white. A color scale bar can be shown to indicate the correspondence between dGEMRIC index values and colors.

As the user evaluates the T1 maps, the user may choose to utilize any of several functionalities. MRIMapper has additional statistical capabilities to give the user information on the pixels that have been chosen for inclusion in the regions of interest. It can exclude or clip pixels with index values that are less than or greater than a user-specified amount. The error for the fit of each pixel is calculated, and this can be displayed as a histogram distribution or as a map overlay on the image. There is also the functionality to exclude from the analysis pixels with an error greater than a user-specified value. The user may use any of these tools to gain insights into the data or to iteratively improve his segment section selection choice, region of interest choice, and segmentation.

2.2.4 Data Analysis

Data analysis is the last step of dGEMRIC image analysis. MRIMapper gives the user a start with data analysis by providing basic statistical information for each region of interest that was mapped. However, as indicated below, it is completely up to the user to determine how to get these basic statistics from each individual slice mapped with MRIMapper to a point where data from all of the slices or subjects can be compiled or compared.

Once the user is satisfied with the quality of the regions of interest and the T1 maps, the user can proceed to the fourth step of dGEMRIC data analysis, statistical data analysis. MRIMapper automatically calculates and displays the mean and standard

deviation values of the dGEMRIC index within each region of interest. This can be seen in the Calculation Results window in Figure 4 above. These are the statistical metrics most frequently used. They are typically reported alongside an image of the slice and map. In the case of a study involving multiple people or time points, the mean and standard deviation of the dGEMRIC index for each of the regions of interest is copied by the user into Excel or some similar software appropriate for statistical data analysis.

3. Methods

3.1 Evaluating Existing Software

There are several different software packages with which one could work in developing an image analysis tool specifically useful for dGEMRIC magnetic resonance images. Those that were considered for use in this project are described below. As the needs for three-dimensional dGEMRIC image analysis were assessed, currently available open source and commercial software packages were assessed for how they could work with MRIMapper to meet those needs.

3.1.1 Image Tool Kit (ITK)

Image Tool Kit is a freely available open source software system designed primarily for medical and physiological applications but can be used in any cases where sophisticated image registration and segmentation schemes are desired (www.itk.org). ITK does not facilitate image visualization, nor do any of its programs provide a graphical user interface for the user. Image Tool Kit program commands can be made from the command prompt, or other open source tool kits can be used to augment ITK

software and provide the user with a graphical user interface or visualization capabilities as desired. ITK software does not have built-in avenues for user-intervention. ITK registration and segmentation schemes run based on parameters provided by the user, but they do not require any input from the user while the images are being analyzed, and they do not take any input from the user that would require the user to analyze the images subjectively. ITK facilitates automatic registration and segmentation. One big advantage to this is that even if these schemes require large amounts of processing time, they require minimal amounts of user time. Another advantage is that the software is open source and one who would be inclined to use it probably has enough experience to modify it or add to it as suits the needs of the project. One main disadvantage is that un-aided segmentation and registration may be unable to achieve reliably accurate results.

3.1.2 Analyze6.0

Analyze6.0 is a commercial software system for biomedical image processing (AnalyzeDirect, Lenexa, KS). Each license is relatively expensive ($4,995 to have the software on one computer at an academic institution). Analyze6.0 has a graphical user interface that would appear familiar to most Windows (or Macintosh) users. It provides a fair amount of functionality that is essentially built-in, including image browsing and segmentation. It allows for segmentation performed by the user, and it also allows for user-aided segmentation. The two kinds of semi-automatic segmentation schemes included in Analyze6.0 are seed-planting and snakes/high gradient method. Analyze6.0 can maintain the association between consecutive slices in the same volume and makes efforts at extrapolating regions of interest from one slice through consecutive slices in a volume. Analyze6.0 also provides minimal capability for the user to perform his own

image processing. This is termed image algebra, and it can involve linear manipulations of multiple images to form a resultant image. (For example, taking the difference of two images.) The main advantages to this software are that it already has the functionality to do a lot of the basic things that you would want image processing software to do and that it is fairly easy for Windows users to learn to use. This software easily (and quickly) loads entire volumes of three-dimensional data, and allows the user to browse through the slices in an intuitive way. Also, this software allows users to easily view multiple images from any source or view simultaneously, which can be helpful in informing the user during segmentation or qualitative image assessment. The main disadvantage to this software is the cost. Additionally, it would not be trivial to create software with complimentary functionality to Analyze6.0. AnalyzeDirect markets a Developer's Add-On product, which qualified users can purchase to develop additional models to Analyze6.0. AnalyzeDirect recommends that users who choose to do this have substantial experience with both the C and Tcl/Tk programming languages. Another option for users who wish to do more than Analyze6.0 can do on its own is to develop additional software in any language that can read the Analyze format images. MATLab is one software that has built-in functionality to read this type of image, so it would be possible for MATLab-based MRIMapper software to read segmentation masks generated using Analyze6.0, but the user would be required to use the two separate software packages independently.

3.1.3 Evaluation of MATLab as software platform for MRIMapper

MATLAB is a commercial software product that has its own high-level technical computing language. MATLab can support graphical user interfaces and is designed to

be an aid in algorithm development, data visualization, data analysis, and numerical computation. MATLab is currently being used for a wide variety of applications in academic, research, and industrial contexts. MATLAB is a popular tool among academic and research institutions. (An individual non-commercial license is $500.) It is not unusual for academic institutions to have large numbers of site licenses for the software, so that students do not need to get their own individual copies. These factors make MATLab an obvious choice for platform in which to develop a freely available research data analysis tool.

The MATLab platform is not perfect, however; there are some disadvantages that were encountered throughout the course of developing this software. The main disadvantage to working on the MATLAB platform is adapting previously written software to run on new releases of MATLAB. The MathWorks is constantly making improvements and updates to their products, including MATLAB. In between major releases the MathWorks issues service packs, which address smaller feature additions or bug fixes. Since the debut of MATLAB 7.0 (Release 14) in June 2004 there have already been two service pack releases (as of March 2005). With the added functionality of each new release comes a new set of bugs, a new set of functions or formats being phased out or replaced, and in some cases, alternate functionality for existing functions. This means that software maintenance for software that runs in MATLAB is non-trivial, if one wishes to continue using the 'old' software with each updated version of MATLAB as it comes out.

3.2 MRIMapper7

There are several different software packages with which one could work in developing an image analysis tool specifically useful for dGEMRIC magnetic resonance images. Those that were considered for use in this project have been described above. Ultimately, MRIMapper was chosen to be used as a base on which to add new features and functionality. The choice was made not to develop functionalities useful for dGEMRIC image analysis in other software to work with the MRIMapper software. This decision was partly motivated by the utility of the MRIMapper software. Although there was plenty of room for improvement, the software was already tailored to handle some of the basic needs of dGEMRIC image analysis. Other software packages were considered in part on the basis of how well they could potentially interface with the MRIMapper software. Other packages promised improved visualization or segmentation, but they would not be able to make a better or faster dGEMRIC fit than MRIMapper already did. Despite some of the attractive features of the other software products, there was not enough reason to motivate re-implementing in a different language those essential and useful features that were already in MRIMapper.

MRIMapper7 was developed using the previous MRIMapper software as a base point. Features were added to aid users in all four steps of dGEMRIC data analysis, to facilitate analysis of three-dimensional data, to map data acquired with a recently developed technique, and to make the software compatible with MATLab 7.0 (R14). The development of the primary features of MRIMapper7 is described below. Some changes are not described, such as those made to ensure MATLab 7.0 compatibility.

3.2.1 Image Browsing

The first task of dGEMRIC analysis is image section selection. If the data are two-dimensional, this task is completed when the data are acquired. At the time of analysis, there is little choice of which images to analyze due to the low volume of data that was acquired. If the data are three-dimensional, the task of image section selection is non-trivial for the analyst. There is a large choice in data, but the time allotted for analysis is usually limited. The analyst employs a method to aid in image section selection, and it would improve the user experience for MRIMapper to facilitate this task rather than to necessitate that the user turn to another software package for this aspect of the analysis.

With the old version of MRIMapper, analysts used other software to aid in image section selection. Some analysts also used the tool designed for aid in image re-alignment to aid them in image section selection. In all cases, analysts first determined that they would analyze a dGEMRIC image and then viewed a selection of imaged sections from one series (corresponding to one of the five delay times) before choosing which section slice to analyze. Analysts preferred to view the slices in anatomical order and chose slices for analysis based on the anatomical features visible in the slice. In some cases where other software was used, it was difficult for the analyst to locate the same image to open in MRIMapper once the image had been chosen using another software. These observations motivated the decision to facilitate image browsing in MRIMapper7 so that the user can easily choose which images are best to analyze.

For the most part, adding this functionality was achieved by building on functions pre-existing in the MRIMapper code. The sub-routine that allows the user to scroll through images of the five different inversion delays being used for analysis was

modified so that it is also used to allow the user to scroll through images of different sections with the same inversion delay. This was done by adding a flag to the code. The flag allows the sub-routine to distinguish between the two different cases and also to take advantage of the functional needs that both cases have in common. In order to properly set the flag, a different entry to the sub-routine was created with the addition of the new functionality.

If the analyst is using MRIMapper7 to browse for an image to be analyzed, MRIMapper7 should also facilitate the loading of that image and the other images that correspond to different inversion delays at the same anatomical location. This was the more challenging aspect of implementing image section selection in MRIMapper7 in the way outlined above. In the previous MRIMapper version, all parameters were entered by the user manually via graphical user interface as shown in Figure 2. Some parameters did not always have to be typed in by the user if they were usually the same as the default values saved in the MRIMapper program. When parameters were entered manually, the graphical user interface guided a series of tasks the user would perform in order to enter the parameter values.

It was possible to incorporate functionality to automatically load parameters for the desired data in the case where the data is in DICOM format. It is now common to use the DICOM image format, which packages into one file both image data and header data. The header data is a structured array containing many fields and values. The particular fields included in the header data depend on the manufacturer of the machine with which the data was acquired. There are some fields that header data from machines of all manufacturers have in common. Two examples are the Width and Height fields that describe the dimension of the accompanying image in terms of pixels. It is these and

other common header fields that contain the parameters that the user would manually enter into the previous MRIMapper. These fields are digitally accessible by loading the header data into a variable and reading the value of the desired field in that variable. In adding the functionality for MRIMapper7 to automatically load parameter values for the chosen image, it was difficult to replicate the previously mentioned order of operations for entering parameters within the program. Ultimately this was achieved by making strategically placed calls to existing sub-routines.

Loading parameters from files that correspond to the same anatomical location as the selected image but different inversion delay times represented another challenge. This required that MRIMapper7 know how to browse the file structure where the data was stored, and locate the appropriate data files. EFILM is software whose use is common for the purpose of retrieving magnetic resonance image data from the scanner and storing it on a local machine. The file structure of the data is determined by the eFILM software and takes the same general organization every time. Each file contains a zero-padded number at the end of the file name and the files for each series are contained in different folders. The dGEMRIC data acquisition protocol dictates that five series of data are collected, and eFILM stores these in five consecutively numbered folders.

One would think that once the address for one image is known (the one selected by the user), it would not be so difficult to find the same image taken with complimentary TI delays by locating the file with the same name, but in a different folder, whose name involves a number with an integer difference less than five. In reality, eFILM does not label the individual files consistently, although it does consistently place all the files for each inversion delay in their own folder. Thus, when MRIMapper7 looks for the files with complimentary inversion delays, it not only checks whether the file exists and

contains inversion recovery data, but it also checks to which slice the file corresponds. When MRIMapper7 is in the proper folder but does not find the proper file on the first try, it loads the header data and reads the appropriate parameters of the files one by one until the proper file is found. The data loading can be time consuming.

In order to cut down on time spent searching for the proper files and to minimize confusion, functionality was built into MRIMapper7 to re-order (via re-naming) the files inside an eFILM series. This requires that the header data from all of the files in the series be read in. This takes a while, but it represents an initial time cost that does not have to be repeated every time data are automatically loaded subsequent to the re-ordering.

3.2.2 Facilitating Comparison Images

Due to the complexity of the human knee, there is a large amount of variability in MRI tissue appearance between people, particularly based on age and joint health. Because of this, automatic segmentation routines have not been developed for knee cartilage in MRI. The most efficient and accurate segmentation scheme with any software is one that is either guided or performed entirely by a human user. The task of segmentation can be involved and time-consuming, and it can become boring if it must be repeated for a series of several or more images. This project has endeavored to develop an aid to segmentation. MRIMapper7 facilitates display of a comparison or reference image to allow the user to have an alternate perspective on the tissue represented in the image being segmented. In the case of segmentation of dGEMRIC data, this could be display of a T2 image, in which the contrast between cartilage tissue and fluid and other tissues is higher than in a T1 image. The user can have more

confidence in the borders that are suspected in the first image by comparing their locations with those of the higher contrast borders in the second image. Alternatively, it could mean the display of a segmentation standard image or an 'ideal knee' image. Viewing these images could give the user an idea of where to find the edges of the cartilage in the image being segmented.

This was achieved primarily by modifying the existent subroutine (run_image.m) that displays images for analysis. Also, an additional button was added in the main window to allow users to access this functionality. Users can activate the button to open a dialog box in which they select the MR image they wish to display. The software checks to see if the image is in a format that is readable, and if so, passes the location information to the image display subroutine which opens the image in a new window. There is no limit on the number of comparison images that can be opened in this manner, even though only one image can be analyzed/segmented at any given time.

Built-in MATLab functions were used to create the user interface where users can browse their files to select the image that they wish to view. Included in the standard MATLab package are commands to open user interfaces to both get and save files. These commands return the file name and the directory location. Also included in the common MATLab platform is a command to complete a file name using the concatenation symbol found in the directory path. In cases, where one knows that all of the directory levels are separated by slashes (/) this is not particularly helpful, but in the case that the user uses MATLab on a different sort of platform, this could be very helpful, and I have chosen to use this command rather than to concatenate my own string using the directory, a dash, and the file name.

The image display subroutine distinguishes between a file being opened for analysis or segmentation and a file being opened simply for comparison purposes by the arguments that are given to the subroutine at the time of the call. In the case of a comparison image, the subroutine is called with a text string argument that would not be there in the case of opening an image for analysis or segmentation. The image display subroutine knows where to expect to find this string, and can thus easily test for its presence in the list of arguments.

This sort of setup where a function can have a variable number of arguments takes advantage of the function argument structure setup in MATLab. In a function definition, if the last argument that a function takes in is called 'varargin', MATLab will place all remaining function arguments in a structure with this name. This variable could have any length, including zero, depending on how the function is called. This feature allows for programming flexibility in cases where it is desired that one function handle more than one specific task.

In the case of the image viewing subroutine, this variable number of input arguments feature is used. If the image is a comparison image, a new figure handle is passed in along with the comparison image to be viewed. If the image is an image to be analyzed, the background image, the map image, and viewing parameters are specified. In this case, the figure handle is not specified, because the image to be analyzed always shows up in the same figure and frame.

3.2.3 Despot Fit

Once the image is segmented, T1 Mapping, the third step of dGEMRIC data analysis, can be performed. As techniques are being developed to acquire three-

dimensional data, techniques are being developed to acquire three-dimensional data more quickly. One of these techniques is the DESPOT technique, which allows collection of data for the whole knee in less than five minutes. This is in comparison to the 3-D SPGR technique, which acquires data for thirty slices, covering the whole knee, in fifteen minutes and the 2-D inversion recovery technique, which acquires data for two slices in fifteen minutes. This technique dictates the specifications in the acquisition of the MRI data, and as a result, it also dictates the appropriate way to fit the data to get the dGEMRIC value as an output. The change in data acquisition specifications dictates a change in data analysis specifications, which needs to be integrated into the data analysis software.

The most important outputs of these analyses are the mapped pixels and the indices that indicate where the mapped pixels belong. A mapped pixel is a pixel for which a fit has been made and a result has been found. The indices that are output are similar to the mask that is input.

To adapt the current software to new data acquisition techniques, the ability to analyze despot data was added. A script to analyze the despot data had already been written by a colleague who had determined the specifications for the data acquisition, Charlie McKenzie. The code that he had written was already in a format that could run on the MATLab platform, and the main challenge was integrating this code into the existing MRIMapper framework, so that this method of analysis could be used as any other method of analysis. Ideally, that would entail creating a new module of software that has the same inputs and outputs as the other analysis modules. In this case, matching up the inputs was not difficult because of the flexible way in which the other analysis

modules were implemented. Many of the necessary parameters are stored in global variables, and are thus accessible from anywhere inside the software.

Matching up the outputs is something that would be ideal, but was not achieved. This was due to the kind of fit that the despot analysis requires. Despot analysis is a new kind of analysis. Within our lab, we experimented with performing the analysis on two, three, or more points. Each point represents a different repetition time. Somewhat surprisingly, the results seemed to be most accurate when only two points were used. So, it seems that in the foreseeable future, the despot analysis will only be used in conjunction with a two-point fit. The other types of analysis that are implemented in MRIMapper are multiple-point fits. In fitting lines to points, the case where there is only two points is significantly different from the case where there are more points. In the two-point case, one typically fits a straight line, but in the other cases there are many choices that are equally obvious. In the case of the fits implemented in MRIMapper, the two-point despot fit uses a least-squares fit scheme, where as the other fits use a minimization scheme on a function of the error of the fit. In trying to minimize the error function, there is some error term, because inevitably the points acquired do not perfectly fit the model. This error is one of the outputs of the analysis module and is used at other points in the software. It can be displayed in different ways so that the user can evaluate the quality of the data. In fitting a straight line to two points, there is no error, so this is not an available output.

3.2.4 Standardization of ROIs and ROI Labeling

Once the data are mapped, data analysis can begin. This is the final step of dGEMRIC data analysis. Both map images and statistical data about regions of interest

are important to dGEMRIC data analysis. In viewing statistical summaries of the regions of interest of one or several images, the user would typically not be viewing the associated dGEMRIC maps simultaneously. It would aid the user to know (without the aid of the image) which statistics correspond to which regions of interest. Without this knowledge proper statistical analyses could not be made. MRIMapper7 allows the user to assign descriptive acronyms to each region of interest that is generated. The list of possible region of interest label assignments may encourage the user to segment each of those regions of interest. For the sake of flexibility, however, the software does not require the user to define all of the regions of interest in the list of labels. The user is prevented from assigning the same label to two different regions of interest in the same image. MRIMapper7 is able to generate this warning by keeping a list of the currently used ROI labels and checking the desired assignment label against the items in this list. In adding this functionality, care was taken with string to character and character to string conversions. Care was also taken with space padding. Padding with spaces is necessary for aesthetic and use reasons, but it can make string comparisons difficult. Human users do not necessarily consider spaces to be as part of strings, but MATLab functions cannot distinguish between a space and any other character, so strings with different numbers of spaces are not considered to be identical.

After segmentation and mapping, the analysis is not complete. For some kinds of studies, such as case studies, images may be the primary way that the data are conveyed. However, in larger studies and clinical trials, statistical analysis is an important part of the methodology.

3.2.5 Saving Data Files for Use in Excel

Once the dGEMRIC values are calculated and maps are produced, the data analysis is not concluded. At this point, the user usually chooses to search for statistical trends in sets of data from multiple knees. This software facilitates this process by providing functionality for users to save a statistical summary of each MR image analyzed (each dGEMRIC map) directly into a format that can be accessed from Excel, a popular spreadsheet software package.

In the previous version of MRIMapper statistical data was displayed for the user in table format as can be seen in the calculation results window in Figure 4. This type of data is necessary for quantitative descriptions in case study situations and for making conclusions from larger studies. MRIMapper could be modified so that statistical analysis could be performed in MRIMapper directly, but there is no need to add so much functionality, storage ability, and organization, when Excel is already a popular and well-functioning tool. In order to make MRIMapper7 interface well with Excel, data from MRIMapper7 needs to be able to be saved in a format that Excel can read. Excel saves files with an .xls extension. These files are in a particular format, but Excel can also read files that are simply stored as tab- or comma-delimited text. Fortunately, in MATLab Release 14 there are xlsread and xlswrite functions built-in that read data from xls files and write data to xls files, respectively. This means that the actual data writing is simplified, since formatting is not as much of a concern. However, it was still an issue of how to prepare the data for use with these functions and also to ensure that data was not being overwritten. In order to feed the data into the xlswrite function it needed to change from being one big text/character string to being stored in a cell array. A text/character string is fine for displaying to the user, but when the software reads it, it is only seen as

being one cell. While one could transfer all of the data into an Excel readable format in this way, once the user had the data open in Excel, it would take a similarly large amount of work to get the data to a format where it can be used from within excel. It is important not only to save the data in an Excel readable format, but also to save the data in a way in which it is easily usable from within Excel.

The basic algorithm is to first check the data that is being written to make sure that it is in a good format both for display when the data is accessed later from within the MRIMapper7 program and for writing into excel. Then, the next step is to gather information from the user as to how/where the data should be written. This information could be standardized, but the program retains the most flexibility if this sort of data organization is left up to the user. In this case, the xlswrite function already has inputs for where the xls file is written to, which sheet the data is written to within the file, and which cell the data starts writing to on the sheet. If the data was originally analyzed with MRIMapper7 and saved into the cell array format, then there is no problem in writing it immediately to the xls format. However, if the data was originally analyzed with an older version of MRIMapper where this statistical information was saved as a character string, then the data needs to undergo a change of formats before it can be effectively written. This kind of juggling posed a challenge. Particularly, it was difficult to find a format that was both readable in Excel and visually pleasant and sustainable within MRIMapper7. In a cell array, the data are stored in different cells, but they are displayed right next to each other, and there are no spaces in between the cell entries. In order to save the data in the cell format, and to display it in a way that makes sense, the data need to be padded with spaces, so that when the values/cell entries are right next to each other, the actual characters are separated by spaces. When Excel reads this type of data in, it

understands that the spaces are superfluous. However, one cannot simply add an extra space or two without already knowing whether there are spaces there. This is because, simply adding extra spaces all the time will cause the cell entries to become larger each time the same data set is re-analyzed or even just re-opened. This is not a problem for Excel inherently, but it makes display within MRIMapper awkward and difficult. To overcome this problem, a system was created whereby the data could be converted between cell array format and matrix format. The matrix format was used for display to the user within MRIMapper7, and the cell array format was used for writing to an xls file. Whenever the data were written to the matrix format, it was ensured that they were padded with a limited number of spaces such that each column had the same width, no greater than ten. This meant that each entry would receive a different number of spaces as padding, but that when the data were converted to cell array format, the column width could be determined by dividing the number of columns in the matrix by four (the number of columns in the cell array).

The other major problem of writing to Excel is that xlswrite has an ability to overwrite data already in the file being saved to, if the name of an existing file is chosen as the file to save to. The user is provided with flexibility in specifying the file name, sheet name, and location within the sheet, but this means that the MRIMapper7 software needs to be vigilant about overwriting data, since this is something that neither the user, nor the xlswrite software will check for. Fortunately, there is xlsread software, which will read in data from a specified location or cell-range in an xls file. Because the MRIMapper7 software knows how large the cell-array is that it would like to write to the xls file, and where in the file it would like to write to, if the xls file already exists, it can check to see if there is anything already written to that specified portion of the file before

writing the new data. In this way, it can be assured that data is not overwritten. The downside is that data will not be written to the location specified by the user. I have chosen to make it such that the data is instead written starting at a cell lower down in the same worksheet. The user is prompted with a warning indicating that the data was saved in a location different from the one specified, and it is indicated to the user where the data was saved.

Another way that accessing the data could have been made easier for the user was if completed maps could automatically be saved into PowerPoint files. Unfortunately, pptwrite and pptread functions (or the like) do not yet exist in MATLab. There are some files floating around the MATLab user community and suggestions as to how to implement this sort of functionality or similar. These approaches were tested, and they were found to be slow, but functional enough to merit inclusion in MRIMapper7. Successfully completing this sort of action is contingent upon timing details regarding the opening of the PowerPoint software. If the correct timing is not achieved, an icon image is saved, but this is not the desired image. In this case, an image is saved to the file specified, and it is difficult to tell that it is not the desired image without actually opening up the PowerPoint file. It was found to be important that the PowerPoint file being saved to be closed at the time of using this functionality. Adhering to this guideline led to consistent results.

4. Results: Using MRIMapper7

MRIMapper7 is software tailored for three-dimensional dGEMRIC image analysis, though it retains all of the functionality of the previous MRIMapper software.

The software resulting from this thesis effort is described in detail here. The software is designed with the hope that anyone who is familiar with magnetic resonance image acquisition and analysis should be able to learn how to use it effectively in fifteen minutes. For those who do not have as much familiarity, a detailed description of how to use the software is provided here. Emphasis will be placed on functionalities that are not in the previous MRIMapper software (Po, 2001).

4.1 Data Preparation and Loading

The first step of dGEMRIC image analysis is image section selection. In the context of MRIMapper7, this task is synonymous with choosing a slice (corresponding to an anatomical plane) to analyze and loading data necessary for analysis of that slice. In MRIMapper7, the user can choose a slice independently of the software, or the user can choose to have the software display a series of slices from which the user will choose a slice for analysis. This functionality requires that the data be organized in a particular file structure. Also, if the data meet the file structure specifications, the user can direct the software to automatically load the four inversion delay files that correspond to the one inversion delay file that the user has selected. File structure requirements for manual loading are much less strict.

4.1.1 Required File Structures

It is necessary to ensure that data is organized in the required file structure before using MRIMapper7. File structure requirements fall into two cases. In the first case, the data are to be manually imported. That is, the five files corresponding to the five delay times will each be selected by the user. In this first case, all of the target data files must

be in the same computer directory, or they must all be in subdirectories that are in the same directory.

In the second case, either the user is using the browse image series functionality, or the data are to be imported semi-automatically. In this case, the data must be in DICOM format and named without an extension. The names of the data files should be the same for all files with the exception of a number at the end of the name. This number could reflect information on the anatomical position of that slice relative to the other slices, but this is not necessary. It is important that the numbers are zero-padded such that the lengths of all of the file names are identical. The files must be organized such that all of the files corresponding to one inversion delay must be in the same folder. Similar to the file names, it is important that the directory names are all the same length and end in a zero-padded number. This number could reflect the series in which the data for that file were acquired, but this is not necessary.

The function Re-Order eFILM/DICOM images also requires this structure. This function may be useful in the case that some of the user's data is processed using eFILM. If the user chooses to acquire his data and move it to his machine for analysis with the help of eFILM, he may notice that the names of the files within the folder for each inversion delay do not correspond perfectly with the location and appearance of the image. Using the function discussed here will allow the user to create a new directory to which MRIMapper7 can copy the files and give them new names corresponding to their slice locations. This task can be performed at any time in the course of using MRIMapper7. To perform this task, the user selects the 'Re-order eFILM/DICOM Files' button in the Data Source box of the Main window (see Figure 7). When this is done, a browsing interface that is standard for the computer platform comes to the front of the

screen. The user is directed by the title of this user interface to 'Pick the DICOM files to be reordered.' The user should navigate the file structure on the computer until all of the files to be reordered can be selected. All of the files which the user desires to re-order in terms of anatomical position must be in the same folder. The user can shift-select or control-select to highlight multiple files. In the case of dGEMRIC data, the user would select a folder corresponding to one of the five inversion delays. The user would open that folder and select all of the files in that folder. After the desired files are selected, the file select user interface closes and another interface comes up directing the user to select a directory. The newly named files will be copied into this directory. It is recommended that the user create a new directory to serve as the target for the copied files. Creating a new directory can be done via this interface. When the user presses 'OK', the directory select user interface closes, and a wait bar appears indicating the progress of the copying and re-naming. The title of the wait bar is 'Please wait...eFILM files being re-ordered'. The user may see some messages echo to the MATLab prompt during this operation. Messages regarding the nature of DICOM header fields are not unusual and are not likely to indicate a problem with the data or with the functionality of MRIMapper7. To re-order all of the data for one three-dimensional dGEMRIC data set, this operation will need to be repeated a total of five times, once for each inversion delay file folder.

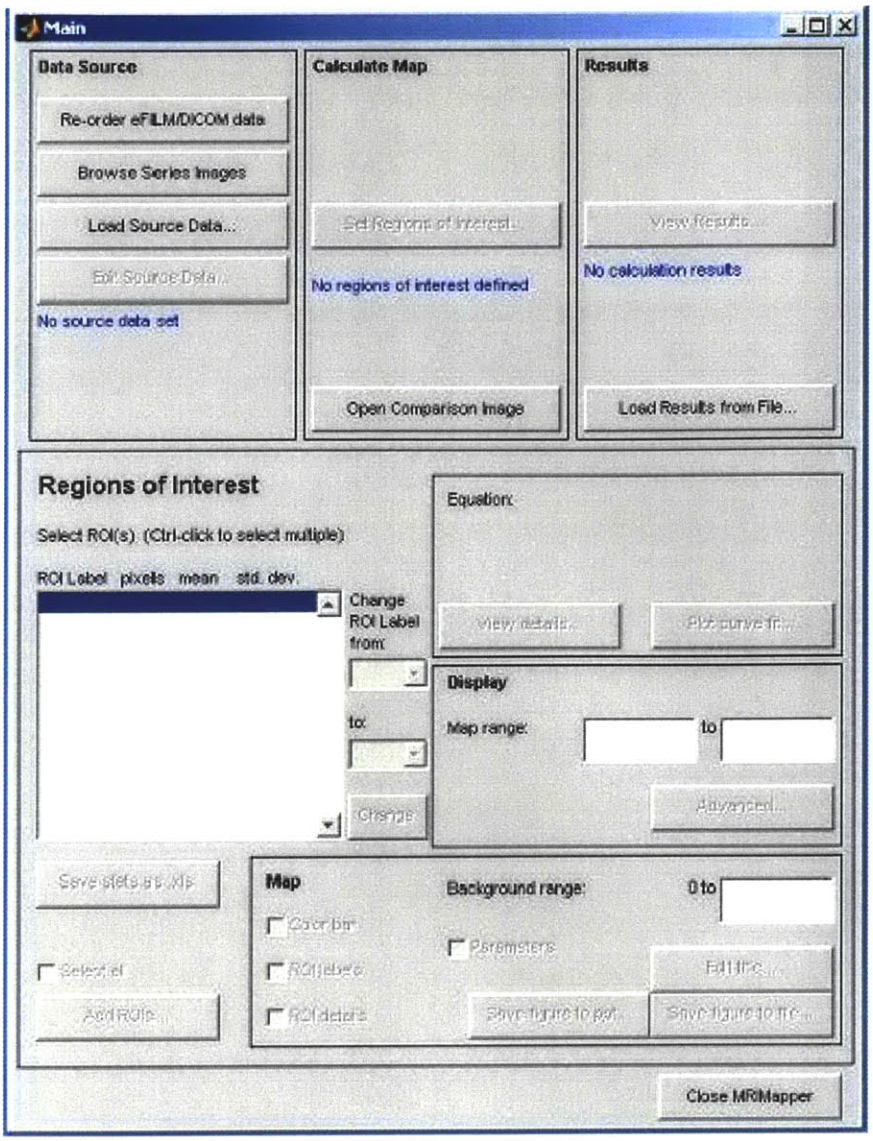

Figure 7: MRIMapper7 Main interface.

4.1.2 Image Browsing

Image browsing is a tool that can be used to aid in image section selection. This functionality can be used in the case that the user has data which is in DICOM format and organized with a separate folder for the data for each inversion delay, as discussed above. All of the images to be browsed must be in the same folder; they should all have the same inversion delay. The user is able to select multiple images through which to page. The

user can compare the images and make an informed choice about which image is best for analysis. When the user chooses an image, MRIMapper7 automatically finds the corresponding files for each of the other four inversion delays and automatically loads the parameters for the set of files.

To browse through multiple slices, the user starts by activating the 'Browse Series Images' button in the Data Source box of the Main MRIMapper7 user interface (see Figure 7). When this is done a standard platform-based file selection interface comes up on the screen. The user navigates the file structure of the computer to locate the images to be browsed. Multiple files are selected using shift-select or control-select. Some users may wish to see all of the slices that were acquired. Other users may have a general idea of which slice they would like to analyze and may choose only four images from which to select one for analysis. When the user presses 'OK' the file-select interface closes and a wait bar comes up as the previously-highlighted files are loaded for viewing. When the wait bar completes and closes all of the files are ready to be viewed and two new windows come up on the screen (see Figure 8). The larger window is a figure window displaying the one of the selected images, and the smaller window contains the image browsing controls. By moving the slide bar in the smaller window, the user can determine which image is shown in the larger window. As the user scrolls through the different files, the title of the image window changes to reflect the section number of the image currently in view.

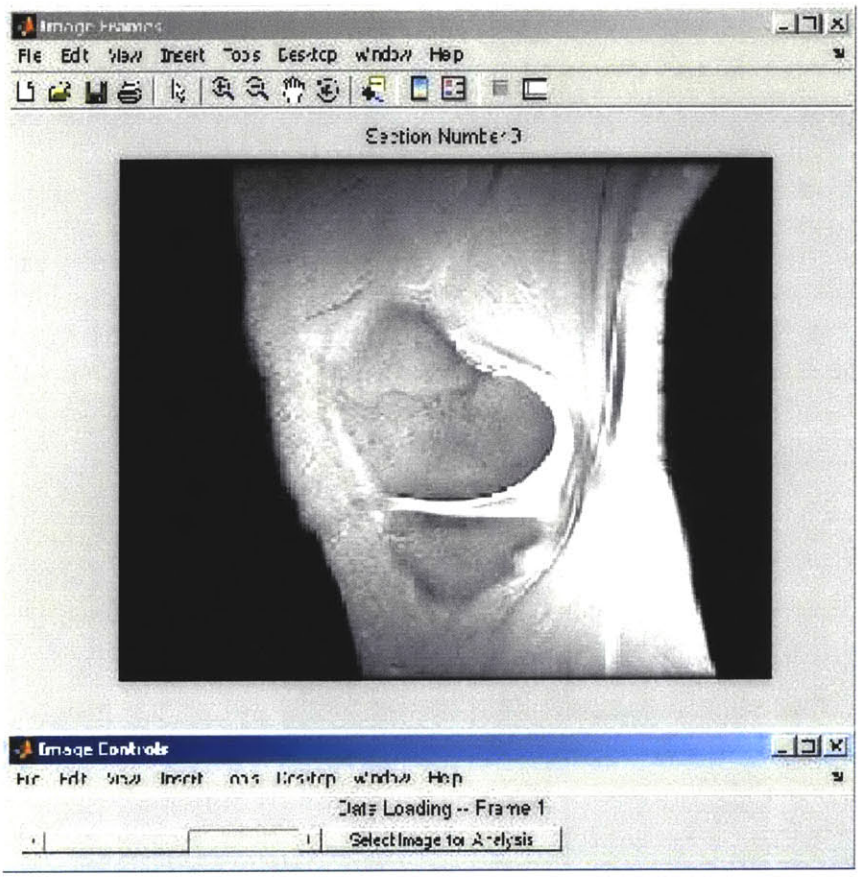

Figure 8: MRIMapper7 Browse Image and Control interfaces.

When the user is finished browsing the files, the user can choose to load the data for one of the files. This is done by pressing the 'Load File' button next to the slide bar when the desired file is in view in the larger window. When this is done, the image window and controls window remain on the screen. A wait bar comes up onto the screen indicating the progress made in finding the four files of the same location with complimentary inversion delays. The image window and controls window may be manually closed by the user. Similar to what was noted above, some warnings regarding the header fields of the DICOM files may echo to the MATLab prompt. This should not be surprising and is likely not a problem. When MRIMapper7 has located all of the proper files, it automatically loads the data for these files. When this is done, the Load

Source Data window will open on the screen. When this happens, a small window will come up on the screen in front of the larger Edit Source Data window. This window contains the times (in seconds) of the inversion times for the loaded files. The user must press 'OK'. When the user presses 'OK', the small window disappears and the TI and number_of_TI fields are updated in the Load Source Data window (similar to Figure 2). At this point, all of the necessary parameters should be visible to the user in the Load Source Data interface. If the user presses 'Load Data', the data will start loading.

4.1.3 Automatic File Loading

If the data and files structure meet the previously-specified requirements, files can also be loaded automatically without first browsing through them. To do this, the user starts at the 'Main' interface and clicks on 'Load Source Data'. When the user interface comes to the screen, the user should start by selecting the proper Source. In the case of dGEMRIC data, this will be either 'Siemens' or 'General Electric'. Next, the user should press the 'Select files...' button. At this point, a question box appears (see Figure 9). The purpose of the question box is to guide the user towards automatic files selection or manual file selection as appropriate and as desired by the user. The dialog box specifies to the data format and structure requirements described above as they pertain to automatic loading. To proceed with automatic loading, the user selects the 'No' button. ('No' is the response to the question indicated in the title of this question box: Will you select all files manually?)

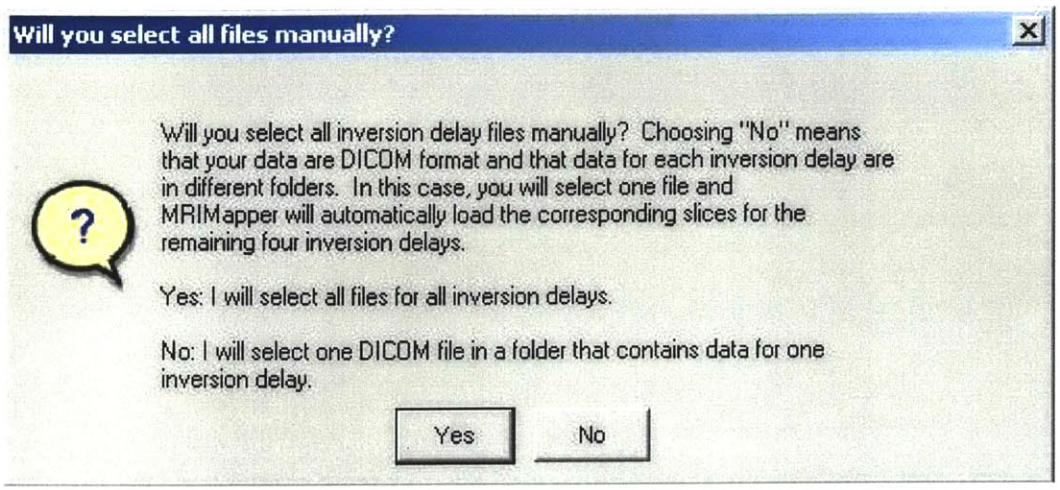

Figure 9: MRIMapper7 Manual Data Selection Dialog.

When the 'No' button is pressed, the question dialog box closes and a platform-based interface for file selection appears. It is titled 'Select a data file'. The user uses this interface to browse the computer file structure to select one file. When the user confirms that single selection, MRIMapper7 closes the file select window and begins searching for the other files that correspond to the same location but have different inversion delays. A wait bar comes to the front of the screen to indicate the progress made. As mentioned previously, echoes to the MATLab prompt regarding header field data of DICOM files are not cause for concern. When all of the appropriate files have been found, MRIMapper7 automatically loads the necessary parameters into the Data Source interface. Again, as mentioned previously, when the Data Source window comes up, there will be a small TI confirmation window. Simply, press 'OK' to confirm the values that MRIMapper7 found. After this is done, the small window will go away, and the fields 'TI' and 'no_of_TI' will be updated to appropriately reflect the data. From here, the data can be loaded by pressing the 'Load' button.

4.1.4 Manual File Loading

Manual File loading begins in a manner similar to automatic file loading. From the 'Data Source' window, the user selects the source (either Siemens or General Electric) and then presses the 'Select files...' button. When the question dialog comes up (see Figure 9), the user chooses 'Yes' to indicate that all files will be selected manually. At this point, the question dialog is covered by directory selection interface. The question dialog may not completely disappear from the screen right away, and this should not hinder the user.

In the directory selection interface, the user should navigate the computer file system to locate the file where all of the data of interest is stored. Data files for the five different inversion delays may all be in the same folder. Alternatively, they may be in different folders, one for each inversion delay. In the former case, the folder that contains all of the data is selected. In the latter case, the folder that contains the folders for each inversion delay is selected. In both cases, the user selects the directory corresponding to the most specific common root of all of the target data files.

Once the directory is selected, MRIMapper7 takes a moment to determine information about the directory contents prior to displaying the next window to the user. The next window is a MATLab-generated window that prompts the user to select the target data files. In the case that the folder selected did not contain any subfolders, the data file selection window contains a list of file names in the directory that the user can scroll through. Control-click and shift-click can be used for multiple file selection. In the case that the folder selected contained subfolders, file names listed in the window will appear as: subfolderName/filename (see Figure 10). These files can be selected in the same way.

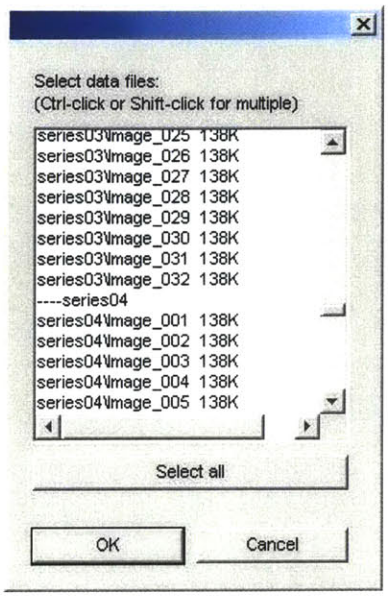

Figure 10: MRIMapper Manual File Selection interface.

Once the target files are highlighted, the user loads their location information into the Data Source window by pressing 'OK'. At this point, the sourceDir and sourceFiles fields in the Data Source window are appropriately filled out. The remaining fields need to be updated by the user manually. Some fields update automatically, and some are almost always the same as their default value. In the case of analyzing a dGEMRIC data set, the user must set the source pull-down menu to 'Inversion Recovery'. The other fields that the user is most likely to need to update are msize, TR, and TI. The msize field contains a two-entry vector indicating the height and width of the image matrix. Typical values are [256 256] and [512 512]. To enter these values, the user clicks on the msize line and a small interface comes to the front of the screen where the user can enter to the two appropriate values (brackets not necessary) and load that parameter value by pressing 'OK'. TR and TI are entered in similar manners. These are vectors indicating the recovery times and inversion times, respectively. It is important that the values in

these fields are entered in the order corresponding to the order in which the files are listed in the sourceFiles field higher up in the Data Source location. It is also important that the recovery and inversion times are entered in the proper units. In the case of an Inversion Recovery analysis, this is seconds. When all of the parameters are properly entered, the data can be loaded by pressing the 'Load' button at the bottom of the Data Source interface.

4.2 Segmentation: Creating Regions of Interest

The second step of dGEMRIC image analysis is segmentation. This functionality is accessed via either the 'Set Regions of Interest...' button at the top of the main interface or the 'Add ROIs...' button towards the bottom of the main interface (see Figure 7). The former is used in cases where no map has yet been calculated for the data being analyzed. The latter is used when at least one map has already been calculated for a previously defined region of interest. In either of these cases, the procedure for defining the region of interest is the same. Additionally, it is the same in MRIMapper7 as in MRIMapper. For details, please see the description given in the MRIMapper section 2.2 above. In MRIMapper7, there are two new functionalities to aid the user in the segmentation task. They are access to comparison images during the segmentation process and the ability to label regions of interest. The use of these functionalities is described here.

4.2.1 Using Comparison Images

At any point in using MRIMapper7, the user may elect to view a comparison image. This functionality is accessed via the 'View Comparison Image' button in the

Calculate Map block at the top of the Main MRIMapper7 window (see Figure 7). This opens a platform-based dialog box that directs the user to choose a comparison image file. The user can browse the computer file structure and choose any image file that MRIMapper7 can open. These image files do not need to be in DICOM format. In the case of dGEMRIC image analysis, the user might find a T2 comparison image to be most helpful. The contrast between cartilage, bone, and fluid in a T2 image is higher than in typical dGEMRIC images, and this contrast may aid the user in segmenting the cartilage. The user may open as many comparison images at a time as are desired. Each comparison image will open in a new image window. The comparison image window will open in the same location (on top of) the main image window. From there, the user may drag it to any convenient location on the desktop. As with all images in MRIMapper7, the user can also control the zoom factor of this window.

4.2.2 Labeling Regions of Interest

In MRIMapper7, the user's experience with segmentation is also aided by the option to be able to label regions of interest. Users can label regions of interest with any of eight pre-determined standard labels (see Figure 11). These labels are acronyms/abbreviations that have been adopted by the cartilage research community. They correspond to anatomical locations. For example, pFC indicates the posterior femoral condyle. The first label on the list is blank. This is to encourage users to activate the list by using the pull down menu before pressing 'Change' (obscured in Figure 11) to assign an ROI the label 'FC'. The last label in the list is a pound sign. Users can assign this label to make the region of interest label an ordinal value corresponding to its

position on the region of interest list. This generally corresponds to the order in which the regions of interest were defined.

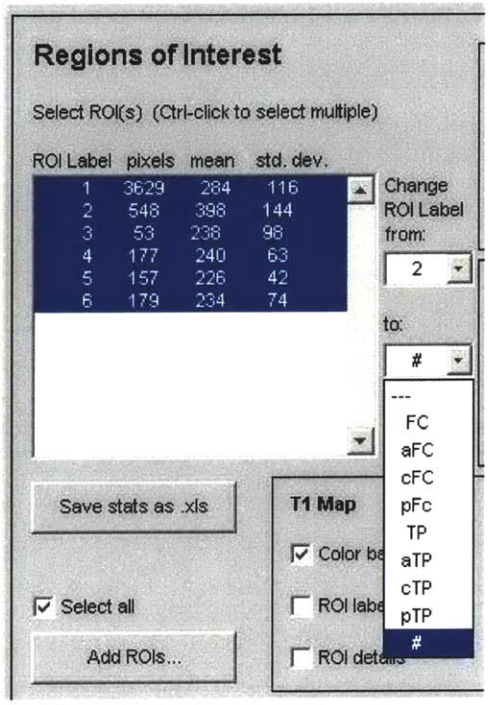

Figure 11: MRIMapper7 Region of Interest Labeling interface.

The region of interest labeling tools are located in the Regions of Interest box in the Main MRIMapper7 window, to the right of the region of interest text box (see Figures 7 and 11). There are two pull down menus indicating the current and possible region of interest labels. There is also a button marked 'Change'.

To re-label a region of interest, the user starts by using the top pull-down menu, above which is stated, 'Change ROI Label from:' (see Figure 11). This menu contains a list of all of the current ROI labels. The user selects the label to be changed. Then the user selects the desired ROI label from the lower pull down menu. If the user sets the lower pull-down menu to an ROI text label that has already been assigned, an error will

come up on the screen directing the user to choose another label. Only one ROI may have any label at any given time. There is no limit to the number of regions of interest that can be labeled with ordinal numbers. Once the user is satisfied with the values shown, the 'Change' button should be pressed to affect the change. Once this happens the list box contents will be updated.

4.3 T1 Mapping

In dGEMRIC image analysis using MRIMapper software, the T1 mapping process is not very involved for the user, as described above. The process is very similar in MRIMapper7. In the new software, there is an additional curve fit option. A new method of T1 image acquisition has been developed. This magnetic resonance imaging scheme is called DESPOT, and MRIMapper7 can perform the T1 fit necessary to analyze data acquired in this way. This choice is made towards the beginning of the analysis session when the user loads the data. The choice is made using the Data pull down menu in the Data Source window (see Figure 2).

4.4 Data Analysis

The fourth step in dGEMRIC image analysis is data analysis. After the dGEMRIC index map is made, the data are ready to be further analyzed. As described above, users typically use both the map as a visual tool and statistics of the regions of interest that were generated throughout data analysis and presentation. In MRIMapper7, both of these tendencies were addressed. In order to aid users in their handling of the relevant statistical data, a direct link was made between MRIMapper7 and Excel. Complimentarily, a direct link was made between MRIMapper7 and PowerPoint

(Microsoft, Redmond, WA), in order to aid users in archiving the maps that they generate for use in future data analysis and presentation.

4.4.1 Saving Data Files for Use in Excel

When the user generates a map for a region of interest, statistical metrics about that region of interest are automatically generated and displayed in the text box of the main window (see Figures 7 and 11). Once the user has generated a map for any number of regions of interest, the user has the option to save these statistics in an .xls file that can be read by Excel. In order to use this functionality, the 'Save stats as .xls' button beneath the text box is pressed (see Figure 11). Statistics for the regions of interest in the text box that are highlighted will be saved in an .xls file specified by the user. It is important to note that the data will not be written to the .xls file if that file is currently open on the desktop. In order to ensure that the data are properly written, it is advisable for the user to close all .xls files, though the Excel program may be left running.

When the button is pressed, the user is first prompted to indicate the file to which the data are to be saved. This is done via a platform-standard dialog box. The user can navigate the file structure on the computer and either specify the name of a new file or choose an existing .xls file. If an existing .xls file is chosen, a warning dialog will appear asking the user if the existing file should be replaced. The user does not need to worry about this sort of warning, because the function in MRIMapper7 write additional data to the .xls file and does not replace the file entirely. Additionally, before writing any data to an existing file, MRIMapper7 checks to ensure that it will not overwrite previously written data.

After the user indicates the .xls file where the data should be written, the user is prompted to enter a worksheet name and a cell location. The worksheet name may be a new worksheet, or a worksheet that the user knows is already in existence in the specified file. The cell specified by the user is the upper-left-hand cell where the data will be written and it should be addressed as A1, where A indicates the first column and 1 indicates the first row. The data for all of the regions of interest will be written across two rows, and the number of columns used will depend on the number of regions of interest that are being written (see Figure 12). For each save operation, the top row is written to label the data, and the bottom row contains the statistics. There are also additional labels on the left- and right-hand sides of the data indicating the organization of the Excel chart and the origins of the data files used to generate the regions of interest. Section Number indicates which number slice was analyzed if the data were in DICOM format. The 'ROIs in:' file address indicates where the .mat file is saved that contains the regions of interest. Data from multiple files can be saved to one worksheet. Even though there will be a row of labels in between each row of data, the user can direct Excel to perform further statistical analyses on columns of data, given that appropriately corresponding regions of interest are lined up to be in the same columns.

	A	B	C	D	E	F	G	H	I	J
1	ROI Label--metric	1 -- pixels	1 -- mean	1 -- std. dev.	2 -- pixels	2 -- mean	2 -- std. dev.		Section Number: 12	
2	value	63	366	63	95	388	69	ROIs in:	H:\mrimapper09.07.03\tests\edit_sourc	

Figure 12: MRIMapper7 Data Exported to Excel.

4.4.2 Saving Map in PowerPoint File

In order to expedite future data analysis sessions or to lend visual aids to presentation, the user may wish to save an image of the map that has been generated (along with the background image). This can be done in MRIMapper7 by saving the image to a ppt file readable by the popular PowerPoint software. To use this functionality, the user should first check to make sure that all PowerPoint files are closed. The PowerPoint program itself can remain open. Once all PowerPoint files are closed the user can save the currently mapped image to a .ppt file by pressing the 'Save figure to ppt' button in the T1 Map box of the Main window of MRIMapper7. Generally, this function is slow, so the user should have some patience throughout the process. First, the user will be prompted to give a title for the slide where the image will be saved. This slide title does not need to be unique. Next, the user will be prompted to indicate the PowerPoint file the slide is to be saved into. The user can browse the computer file structure and choose an existing PowerPoint file, or the user may name a new file. After the user confirms the name of the file, the image will be saved. If an existing file is chosen, the user may be asked whether or not the file should be replaced. No data that is already in the file is overwritten. Each time this functionality is used, a new slide is written to the file. If the PowerPoint software is not open when this happens, MRIMapper7 will open the software in order to save the image. The user may or may not see the image briefly on the screen as MRIMapper7 saves it. This should not be taken as a sign of error. When MRIMapper7 has completed saving the image, it will display a message to the MATLab prompt indicating the location of the saved file. Once this has been printed, the user may open PowerPoint and view the file that had been

specified to confirm that the save was performed properly. If the user elects this option, the user should remember to close the PowerPoint file before saving another image.

5. Discussion: Comparison

While the overall look and feel of the new MRIMapper7 is similar to that of the older MRIMapper, the new software has added functionality which improves the convenience of three-dimensional dGEMRIC image analysis. All four stages of the process were addressed, and the user may notice the most difference at the image section selection and data analysis stages.

5.1 Image Section Selection

Many magnetic resonance scanners now save data in DICOM format. This format includes in the data file meta information about the data. MRIMapper7 takes advantage of this frequently available information to facilitate image section selection. When using the previous MRIMapper software, the user must know which image section to analyze before the user begins to use the program. This was reasonable when the data available for analysis were one or two slices. In the case where a three-dimensional data set was to be analyzed, other software had to be used to perform image section selection prior to loading the data into MRIMapper. With MRIMapper7, the user has several options that speed up image section selection and loading DICOM data. The user can use MRIMapper7 to page through multiple slices—anywhere from two to all of the slices for one knee. When the slice to be analyzed is identified, the user need only press a button to have all of the parameters for that slice and its corresponding other inversion delays to be

loaded into the MRIMapper7 software. Then the user clicks once to confirm the parameters, and once again to load the actual images. Then the user can proceed immediately to segmentation, if desired. If the user already knows which slice is to be analyzed, the user can browse the computer file system for that slices data file and can direct MRIMapper7 to load the parameters and corresponding inversion delay data files automatically. If the data that the user is analyzing are in DICOM format, the user does not have to manually enter any parameters. Whereas the previous MRIMapper did not aid the user in data management, using MRIMapper7, the user can copy an entire directory of DICOM images, renaming them so that their filenames are numbered in anatomical order. This feature is complimentary to the image browsing function. With the previous MRIMapper, it was necessary to use another software to perform image section selection when analyzing three-dimensional dGEMRIC data. MRIMapper7 fills user needs with regards to image section selection saving time and hassle.

5.2 Segmentation

The previous MRIMapper facilitated manual segmentation. The user could define any number of regions of interest by dragging the cursor across the screen. In MRIMapper7, segmentation is not automated, but the user has more tools to aid in this task. In order to be confident in the segmentation with the previous MRIMapper, the user might have opened a book of standard magnetic resonance images or printed out a T2 image of the same slice. In MRIMapper7, the user can open up any number of comparison images via the software itself. The user may elect to open a T2 image or may choose to view neighboring slices to aid in segmentation. Additionally, in MRIMapper7, there is a static list of eight region of interest labels to help guide the user as to what type

of segmentation might be appropriate. The user can then use these labels to highlight which regions of interest are the 'final' versions. This could be helpful if the user makes iterative attempts at defining, for instance, the central tibial plateau, region of interest. The regions of interest are given ordinal labels as default (as in the previous MRIMapper), but in MRIMapper7, the user can label the most satisfactory region of interest with the cTP label. Segmentation is not by any means a trivial task in MRIMapper7, but with the aid of comparison images and region of interest labeling, it is an easier chore.

5.3 T1 Mapping

Since the development of the previous MRIMapper, new T1 data acquisition schemes have been developed. MRIMapper7 incorporates into the software the ability to map data acquired with the DESPOT technique, one of the most promising of these new schemes. MRIMapper7 retains all of the mapping ability of the previous MRIMapper and adds DESPOT fitting to its list of tools.

5.4 Data Analysis

The previous version of MRIMapper provided little in terms of data analysis tools beyond filtering based on pixel intensity and display of basic region of interest statistics (size, mean, and standard deviation) to an MRIMapper window. Users read these values off of the screen and typed them manually into an Excel worksheet. When using the previous MRIMapper users made PowerPoint slides of different slices that were analyzed in one subject's knee, copying each image and color bar manually. MRIMapper7 provides additional tools to help the user bridge the gap between maps in MRIMapper7,

data analysis in Excel, and data presentation in PowerPoint. In MRIMapper7, the user can export the basic region of interest statistics to an Excel worksheet by pressing a button. This reduces both the time cost for the task and the risk of human error in transcribing the values. Statistics are written to the Excel sheet in such a way as to minimize the time necessary to re-organize them for analysis with statistics from related regions of interest. Additionally, in MRIMapper7, users can press a button to have the mapped image saved in a PowerPoint slide. In MRIMapper7 this function is slow, but it may be easier for some users than copying and pasting image, color bar, and statistics. MRIMapper7 helps users to start in other software data analysis that cannot be done within MRIMapper7.

6.Conclusion: Future Work

MRIMapper7 is a tool that will be freely available to many researchers as dGEMRIC data analysis becomes accepted and established, and as three-dimensional data acquisition becomes the norm. However, as the magnetic resonance imaging fields advance, new needs will arise. And, as dGEMRIC image analysis continues to gain popularity, new needs will become evident as the use of MRIMapper7 becomes more widespread. These last sections highlight some of the anticipated future directions of development for MRIMapper7.

6.1 Robustness

As the number of people who use MRIMapper increases, it becomes more and more important to make the software extremely robust. It is difficult to predict how

exactly a user will try to use a piece of software, and it is important that the software can withstand a reasonable degree of misuse. Analysis software like MRIMapper should not necessitate that the user re-start MATLab every time the user does something that the software had not predicted.

The current functionality in MRIMapper7 that permits it to automatically read parameter data from DICOM headers has proved to be extremely helpful to data analysts throughout testing. As the robustness of the software is improved, it would be good to also improve the intelligence of the software. Whenever possible, the software should fill in parameters whose values the software can access via header data.

6.2 Flexibility

As the field of magnetic resonance imaging grows, newer, better, faster, data acquisition techniques will be developed. There will be demand to analyze data acquired with these techniques for quite a while, and it would be ideal if MRIMapper could be made to include enough flexibility such that new mapping techniques can be integrated into the software easily. So, in addition to needing some greater robustness, MRIMapper would also benefit from a greater sense of flexibility.

6.3 Statistical Metrics

Lastly, investigation should continue into how MRIMapper could better serve analysts for three-dimensional data in particular. Statistical analyses that take into account data from neighboring slices may prove helpful, and new statistical analysis methods should continue to be investigated so that they can be integrated into the software as soon as possible.

Works Cited

Arthritis-Foundation (2004). Osteoarthritis (OA). Disease Center, Arthritis Foundation. 2004.

Bashir, A., M. L. Gray, et al. (1997). "Glycosaminoglycan in articular cartilage: in vivo assessment with delayed Gd(DTPA)(2-)-enhanced MR imaging." Radiology 205(2): 551-8.

Bashir, A., M. L. Gray, D. Burstein. (1996). "Gd-DTPA^{2-} as a Measure of Cartilage Degradation." Magnetic Resonance in Medicine (36) 665-673.

"The Insight Segmentation and Registration Toolkit". www.itk.org

Po, B. (2001). Graphical Interface for Quantitative T1 and T2 Mapping of MRI Data. Electrical Engineering and Computer Science. Cambridge, Massachusetts Institute of Technology: 119.

Rasband, W.S., ImageJ, U. S. National Institutes of Health, Bethesda, Maryland, USA, http://rsb.info.nih.gov/ij/, 1997-2005.

Williams, A., A. Gillis, et al. (2004). "Glycosaminoglycan Distribution in Cartilage as Determined by Delayed Gadolinium-Enhanced MRI of Cartilage (dGEMRIC): Potential Clinical Applications." AJR (American Roentgen Ray Society)(182): 167-172.

Appendices: MRIMapper7 code

A.1 User Interface Control Files

```
function ret = run_analyzeCurve( action, varargin )
%RUN_ANALYZECURVE  User interface method file


  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' );
    push_getVal = findobj( allchild(fh), 'Tag', 'push_getVal' );
    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' );

    set( push_getVal, 'String', ['Get ' parameterType ' value'] );

    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 = sourceParams.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;
delete( h );  %changed from close to delete 31.12.2004
       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);
    val = theMapSave(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 = sourceParams.msize;
    %set( push_curves, 'Enable', 'off' );

    % load source data (if needed)
    if( ~get(rad_ROI,'Value') | get(chk_showRange, 'Value') )
      if( ~figsValid )
    a = loadSourceData( fh );
    if( a==0 ) 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;
delete( h );  %changed from close to delete 31.12.2004
      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);
      %class_pfittedSaveia=class(pfittedSave)
      pfitted = double(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' ))
      infoStr = ['(' num2str(x) ',' num2str(y) ')'];
```

```
      title( ['Data for ' infoStr] );
      i = getIndex( x, y );  % this just converts x,y to linear index
      ii = find(theMapi==i);
      if( isempty(ii) )
    val = '';
    p = '';
    p=double(p);
      else
    val = theMapSave(y,x);  % the T1 or T2 value for that pixel
    p = double(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( [parameterType ':' num2str(val) '  '], ...
               ['pfitted (estimated parameters): [' num2str(p) ']  ']
);
    %class_p1=class(p)
    %p1=p
      end; %if
      figure( new_h );
      axis off;
      text( 0, 0, infoStr );
      axes( 'position', [.1, .25, .8, .65]);
      %class_p2=class(p)
      %p2=p
      %class_dataPoints=class(dataPoints)
      feval( ['plot_' dataParams.type], 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( 'Data for region of interest' );
      axis off;
      text( 0, 0, infoStr );
      axes( 'position', [.1, .25, .8, .65]);
      feval( ['plot_' dataParams.type], '', pMean, ...
         get(chk_logScale,'Value'));
    end;


    if( get(chk_showRange, 'Value') &
~strcmpi(dataParams.type,'perfusion'))
      hold on;
      t = feval(['def_' dataParams.type], 'get_timeaxis');
      %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 sourceParams;
  msize = sourceParams.msize;
  i = (x-1)*msize(1)+(y-1)+1;


function ret = run_analyzeParams( action, varargin )
%RUN_ANALYZEPARAMS  User interface method file


  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', 'right' );            %%%%%%% changed 10.05.2005
    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 = strvcat( [parameterType ' map'], ...
            feval(['def_' dataParams.type], 'get_paramList' ));
    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 )
      dataUnion = tempMap(tempMapi);
    else
      [c ia ib] = intersect(theMapi, tempMapi);
      dataUnion = pfittedSave(ia,whichField-1);
    end;

    if( isempty(dataUnion) )
      set( edit_fieldInfo, 'String', 'no. pixels: 0' );
      return;
    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) | isempty(dataUnion)) return; end;
    % dataUnion should already be update
    figure;
    plot( double(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'
     if( isempty(dataUnion)) return; end;
     figure;
     hist( double(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(sourceParams.msize);
       displayMap(c) = pfittedSave(ia,whichField-1);
     end

     new_h = run_image('init','another');
     placeFig( new_h, 'bottom', 'right' );
     disp('have run_image with more than standard number of arguments');
     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 = sourceParams.msize;

     h = dlgInfo( 'Select an image and draw a line', '', [200 40 10] );
     waitfor( 0, 'CurrentFigure' );
     figHandle = gcf;
delete( h );  %changed from close to delete 31.12.2004
     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, varargin )
%RUN_DATASOURCE  User interface method file


  include_globals;
  persistent fh;
  persistent pop_source pop_data list_dataParams list_sourceParams ...
      push_loadParam push_sourceDir txt_message ...
      push_load push_close;
  global source_list sourceLabel_list data_list dataLabel_list;

ret = '';

if isfield(sourceParams, 'type')           %11.04.2005 seems like this
could be commented out
if strcmp('linux', sourceParams.type)
  sourceParams = struct( ...
    'sourceDir', sourceParams.sourceDir, ...
    'sourceFile', sourceParams.sourceFile, ...
    'dataScale', sourceParams.dataScale, ...
    'msize', sourceParams.msize, ...
    'no_of_slices', sourceParams.no_of_slices, ...
    'slice', sourceParams.slice, ...
    'offset', sourceParams.offset, ...
    'noise_level', sourceParams.noise_level, ...
    'median_filter', sourceParams.median_filter, ...
    'byte_ordering', 'little endian', ...
    'type', 'bruker' );
end
end

  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_source = findobj( allchild(fh), 'Tag', 'pop_source' );
    pop_data = findobj( allchild(fh), 'Tag', 'pop_data' );
    list_dataParams = findobj( allchild(fh), 'Tag', 'list_dataParams'
);
    list_sourceParams = findobj( allchild(fh), 'Tag',
'list_sourceParams' );
    txt_message = findobj( allchild(fh), 'Tag', 'txt_message' );
    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_source, 'String', strvcat(sourceLabel_list) ); %
definitions.m
    set( pop_data, 'String', strvcat(dataLabel_list) ); % 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( sourceValid )
      run_dataSource( 'update' );
    else
      set( list_dataParams, 'Value', 1 );
      set( list_sourceParams, 'Value', 1 );
      feval( ['def_' source_list{1}], 'default' );  % reset
sourceParams
      feval( ['def_' data_list{1}], 'default' );    % reset dataParams
      viewParams = def_view( 'new' );       % reset viewParams
%07.02.2005 changed from 'default' to 'new
      run_dataSource( 'update' );
    end;
    ret = fh;

      case 'init_from_browse'
          run_dataSource('init');
          %disp('in run_dataSource, case init_from_browse');
          %allow functionality to time out
          t1=clock;
          t2=clock;
          fileNameLong=char(varargin{1});
          [firstFilePath, fileNameShort, ext]=fileparts(fileNameLong);
          [fileRoot, firstFileSeries]=fileparts(firstFilePath);
          lengthFileName=length(firstFileSeries);
          firstFileNumber=firstFileSeries(lengthFileName-
1:lengthFileName);
          fileInfo=dicominfo(fileNameLong);  %this is the dicominfo on
the original file
          thisImage=fileInfo.InstanceNumber;
          allFileInfo(1)=fileInfo;

          lengthFileNameShort=length(fileNameShort);
          firstFileNameShortNumber=fileNameShort(lengthFileNameShort-
1:lengthFileNameShort);
          imSearchRadius=0;
          imSearchDir=1;
          %to locate other inversion delays, check that the field
          %ProtocolName contains dGEMRIC? <--human entered, prone to
error,
          %unnecessary?  is it the same automatically for all of the
          %series?  probably.
          %check that the inversion/repetition fields are what would be
          %expected?
          %repitition time should be same or nearly same
```

```
            %inversion time should be in order, anywhere from 100-4000ms
            %inversion time should not be zero for dGEMRIC.
            %SeriesDescription field should make sense <--this is
            %user-entered.

            %there should be a total of five different inversion delays
for
            %dGEMRIC analysis, so let's make a vector that counts what we
            %have so far.
            imagesLocated=1;
            fileNames=fullfile(firstFileSeries,fileNameShort);
            fileNamesArr{1}=fullfile(firstFileSeries,fileNameShort);
            dirSearchRadius=1;
            dirSearchDir=1;
            maxFileNumberFlag=0;
            maxFileNumber=str2num(firstFileNameShortNumber);
            h = waitbar(0,'Finding Files...');
            while length(imagesLocated)<5 & etime(t2,t1)<60*3
                %will keep on looking until it finds five files
                %or it times out
                %imagesLocated  %for debug 18.05.2005

nextFileNumber=str2num(firstFileNumber)+dirSearchDir*dirSearchRadius;

                nextFileNumber=num2str(nextFileNumber);

nextFileNameShortNumber=str2num(firstFileNameShortNumber)+imSearchDir*i
mSearchRadius;
                nextFileNameShortNumber=num2str(nextFileNameShortNumber);
                if length(nextFileNumber)==1
                    nextFileNumber=strcat('0',nextFileNumber);
                end  %if length(nextFileNumber)
                if length(nextFileNameShortNumber)==1

nextFileNameShortNumber=strcat('0',nextFileNameShortNumber);
                end%if length(nextFileNameShortNumber)
                nextFileSeries=strcat(firstFileSeries(1:lengthFileName-
2),nextFileNumber);
                nextFileDir=fullfile(fileRoot,nextFileSeries);
                if exist(nextFileDir,'dir')
                    nextFileNumber=str2num(nextFileNumber);
                    if nextFileNumber>maxFileNumber
                        maxFileNumber=nextFileNumber;
                    end %if nextFileNumber>maxFileNumber
                else
                    %disp('this directory does not exist')
                    dirSearchRadius=1;
                    dirSearchDir=dirSearchDir*-1;
                    continue %to next step in while loop
                end %if exist(nextFileDir)

nextFileNameShort=strcat(fileNameShort(1:lengthFileNameShort-
2),nextFileNameShortNumber);

nextFileNameLong=fullfile(fileRoot,nextFileSeries,[nextFileNameShort
ext]);
```

```
            if ~exist(nextFileNameLong)  %if file does not exist,
image number is too large or small
                %disp(['This file does not exist, I will look
elsewhere ' nextFileNameLong]);
                if nextFileNumber==maxFileNumber  %has the
maxFileNumber been reached?
                    maxFileNumberFlag=1;
                    dirSearchRadius=1;
                    dirSearchDir=dirSearchDir*-1;
                else   %if nextFileNumber== (if you change dir, check
same file)
                imSearchRadius=1;
                imSearchDir=imSearchDir*-1;
                end %if nextFileNumber==
                t2=clock;
                continue  %continues to next step of while loop
            end %if ~exist(nextFileNameLong)
            fileinfoNext=dicominfo(nextFileNameLong);
            nextImage=fileinfoNext.InstanceNumber;
            if fileinfoNext.InversionTime==0
                %disp(['file ' nextFileNameLong ' does not appear to
be in a dGEMRIC series.']);
                if maxFileNumberFlag==1  %if you've reached the last
dir, look at previous ones
                    dirSearchRadius=1;
                    dirSearchDir=dirSearchDir*-1;
                end %if maxFileNumberFlag
                dirSearchRadius=dirSearchRadius+1;  %if you haven't
reached the last, look at the next
            elseif ~(nextImage==thisImage)
                %image does not correspond to proper slice.  stay in
same
                %directory (same search radius) but change
fileNameShort
                %disp(['file ' nextFileNameLong ' has image number '
num2str(nextImage) ' which is different from the desired image '
num2str(thisImage)]);
                imSearchRadius=imSearchRadius+1;
            else
                imagesLocated=[imagesLocated max(imagesLocated)+1];
                waitbar(length(imagesLocated)/5,h);
                allFileInfo(length(imagesLocated))=fileinfoNext;

fileNames=strvcat(fileNames,fullfile(nextFileSeries,nextFileNameShort))
;

fileNamesArr{length(imagesLocated)}=fullfile(nextFileSeries,[nextFileNa
meShort ext]);
                dirSearchRadius=dirSearchRadius+1;
                imSearchRadius=0;
                if dirSearchRadius>4
                    dirSearchRadius=1;
                    dirSearchDir=dirSearchDir*-1;
                end %if dirSearchRadius
            end %if fileinfoNext.InversionTime
        t2=clock;
        end  %while length(imagesLocated)
```

```
            %disp('all images located')  %debug 18.05.2005
            close(h)
            t2=clock;
            if etime(t2,t1)>60*2.5  %operation timed out
                disp('*');
                disp('MRIMapper could not find all coresponding inversion
delays.');
                disp('Please select files manually.');
                return;
            end %if etime
            %fileNames=sort(fileNames)  %make sure names are ordered
alphanumerically
            %above is wrong because it sorts by column
            [fileNamesArr sortIndex]=sort(fileNamesArr);
            fileNames=char(fileNamesArr(1));
            for i=2:5
                fileNames=strvcat(fileNames,char(fileNamesArr(i)));
            end

            sourceParams.type='ge';
            feval(['def_' sourceParams.type], 'default');
            dataParams.type='ir';
            feval(['def_' dataParams.type], 'default');
            run_dataSource('update');
            sourceParams.sourceDir=fileRoot;
            sourceParams.sourceFiles=fileNames;   %%does this need to be
comma seperated, or how is it?
            w=fileInfo.Width;
            h=fileInfo.Height;
            sourceParams.msize=[w h];
            dataParams.TR=0;
            dataParams.TI=0;
            dataParams.section_number=thisImage;
            if maxFileNumberFlag
                numImages=maxFileNumber+4;
            else numImages=36;
            end
            for i=1:5
                sI=sortIndex(i);
                %one of these needs to be changed according to number of
slices, etc
                %assumes that there are 32 slices taken
                dataParams.TI(i)=allFileInfo(sI).InversionTime/1000;
%time is entered in seconds

dataParams.TR(i)=((allFileInfo(sI).RepetitionTime*numImages)/1000)+data
Params.TI(i);  %time is entered in seconds
            end  %for i
            %TR=TI+tr*numSlices(before slices dropped)  %how to find
number
            %of slices? go through files in one directory and find max
            %instanceNumber

            if mean(dataParams.TI)<0.01
                dataParams.TI=dataParams.TI*1000;
                dataParams.TR=dataParams.TR*1000;
            end  %in case it was already in milliseconds
```

```
          %SeriesNumber %the number of this series (if you start at
one,
          %but the directory label starts at zero)
          %InstanceNumber  %the number of this slice (if you start at
one,
          %but the directory label starts at zero)
          run_dataSource('update');
          def_ir('input','TI');
          def_ir('TI');
          run_dataSource('update');


   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'))
        disp('run_dataSource is not open')   %debug 12.05.2005
        return;
    end;

    %set( pop_source, 'Value', find(strcmp(sourceParams.type,
source_list)));
    %set( pop_data,'Value', find(strcmp(dataParams.type, data_list)));
    sourceParamsType=sourceParams.type;
    dataParamsType=dataParams.type;
    set( pop_source, 'Value', 3);  %corresponds to source_list
    set( pop_data,'Value', 3);   %corresponds to data_list
    tempStruct = rmfield( sourceParams, {'type'} );  %actually removes
the field
    %and returns the resulting structure (with the remaining fields)
    %set( list_sourceParams, 'String', struct2string( tempStruct ))
    set( list_sourceParams, 'String', struct2string( sourceParams ));
%changed 12.05.2005
    %get(list_sourceParams, 'String')
    %get(list_sourceParams, 'Value')    %debug 12.05.2005
```

```
    tempStruct = feval(['def_' dataParams.type], 'display_struct');
%removes the fields type and timevar
    %ren_struct   %ren_struct is the rename structure; see
struct2string.m
    %12.05.2005
    %set( list_dataParams, 'String', struct2string( tempStruct , ...
          %    ren_struct ))
    set( list_dataParams, 'String', struct2string(dataParams));
%changed 12.05.2005
    %get(list_dataParams, 'String')
    %get(list_dataParams, 'Value')    %debug 12.05.2005

    % update the message box
    if( isSourceEmpty( sourceParams ))
      set( txt_message, 'String', 'Source file(s) need to be set' );
      set( push_load, 'Enable', 'Off' );
      sourceValid = 0;
    elseif( isempty( dir(sourceParams.sourceDir)))
      set( txt_message, 'String', ['Data source directory is invalid']
);
      set( push_load, 'Enable', 'Off' );
      %sourceValid = 0;    ** Do not change valid status **
    elseif( ~sourceValid )
      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_source'
            conCh=confirmChange   %%added/changed 23.05.2005
            if ~(conCh), return;, end;
    a = get( pop_source, 'Value' );
    newType = source_list{a};
    if( ~strcmp(sourceParams.type, newType) )   % user changed
something?
      feval( ['def_' newType], 'default' ); % reset sourceParams
      viewParams = def_view( 'new' );   % reset viewParams %07.02.2005
changed from 'default' to 'new'
      sourceValid = 0;
      set( list_sourceParams, 'Value', 1 );
      feval(mfilename,'update');
    end
    run_main( 'update' );

   case 'pop_data'
            conCh=confirmChange   %%added/changed 23.05.2005
            if ~(conCh), return;, end;
    a = get( pop_data, 'Value' );
    newType = data_list{a};
    if( ~strcmp(dataParams.type, newType) ) % user changed something?
```

```
      feval( ['def_' newType], 'default' ); % reset dataParams
      viewParams = def_view( 'new' );   % reset viewParams  %07.02.2005
changed from 'default' to 'new'
      sourceValid = 0;
      set( list_dataParams, 'Value', 1 );
      feval(mfilename,'update');
    end
    run_main( 'update' );

  case 'push_sourceDir'
            conCh=confirmChange   %%added/changed 23.05.2005
          if ~(conCh), return;, end;
    temp = feval( ['def_' sourceParams.type], 'input', 'uigetfile' );
    if( temp==1 )           % user changed something?
      sourceValid = 0;
      run_dataSource( 'update' );
    end;

  case 'list_sourceParams'
    a = get( list_sourceParams, 'Value' );  % which field selected?

    fields = fieldnames(sourceParams);
    fieldname = fields{a};

    % These won't work if fieldnames were renames
    %strlist = get(list_sourceParams, 'String');
    %str = strlist(a,:);
    %i = findstr(str,' ');
    %fieldname = str(1:i-1);

    % Some parameters are okay to edit without affecting already
    % calculated data.
    if( strcmp(fieldname,'noise_level'))
      critical_field = 0;
    else
      critical_field = 1;
    end;

    if( critical_field )
              conCh=confirmChange   %%added/changed 23.05.2005
          if ~(conCh), return;, end;  % check calculated data
    end;
    temp = feval( ['def_' sourceParams.type], 'input', fieldname );

    if( temp==1 & critical_field )      % user changed something?
      sourceValid = 0;
      run_dataSource( 'update' );
    end;

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

    % These won't work if we rename fields
    %strlist = get(list_dataParams, 'String');
    %str = strlist(a,:);
```

```
    %i = findstr(str,' ');
    %fieldname = str(1:i-1);

    % Some parameters are okay to edit without affecting already
    % calculated data.
    if( strcmp(fieldname,'initial_guess') )
      critical_field = 0;
    else
      critical_field = 1;
    end;

    if( critical_field )
                conCh=confirmChange   %%added/changed 23.05.2005
            if ~(conCh), return;, end;
    end;

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

      sourceValid = 0;
      run_dataSource( 'update' );
    end;

   case 'push_loadParam'
              conCh=confirmChange   %%added/changed 23.05.2005
            if ~(conCh), return;, end;
    [filen, pathn] = uigetfile('*.mat', ['Select file to load' ...
            ' parameters from'] );
    if( filen~=0 )          % not a cancel?
      sourceValid = 0;
      s = load( [pathn filen], 'sourceParams', 'dataParams' );
      if( ~isfield(s,'sourceParams') | ~isfield(s,'dataParams' ))
    errordlg( [filen ' does not contain the required data'], ...
          'Error', 'modal' );
    return;
      end;
      feval( ['def_' s.sourceParams.type], 'default' ); % reset
sourceParams
      feval( ['def_' s.dataParams.type], 'default' );   % reset
sourceParams
      viewParams = def_view( 'new' );       % reset viewParams
%07.02.2005 changed from 'default' to 'new'
      sourceParams = structcombine( sourceParams, s.sourceParams, 1 );
      dataParams = structcombine( dataParams, s.dataParams, 1 );
      run_dataSource( 'update' );
    end;

   case {'push_resetSourceParams','push_resetDataParams'}
              conCh=confirmChange   %%added/changed 23.05.2005
            if ~(conCh), return;, end;
    if( strcmp(action,'push_resetSourceParams') )
      a = get( pop_source, 'Value' );
      feval( ['def_' source_list{a}], 'default' );  % reset
sourceParams
    else
      a = get( pop_data, 'Value' );
```

```
      feval( ['def_' data_list{a}], 'default' );    % reset dataParams
    end;
    viewParams = def_view( 'new' );         % reset viewParams
%07.02.2005 changed from 'default' to 'new'
    sourceValid = 0;
    set( list_dataParams, 'Value', 1 );
    set( list_sourceParams, 'Value', 1 );
    run_dataSource( 'update' );

   case 'push_load'
    if( strcmpi(get(push_load, 'Enable'), 'On') )
      a = loadSourceData( fh, ['Error loading data.  Make sure all' ...
            ' parameters are correct.'] );
      if( a==0 ) return; end;
      feval( ['def_' dataParams.type],'set_backgroundImg' );
      sourceValid = 1;
      figsValid = 1;
      doSaveSrcLocal = 0;
      run_dataSource( 'close' );
      run_ROIlist( 'close' );
      run_editSource( 'close' );
      %run_viewResults( 'close' );  %no need because it is attached to
main
  %13.01.2005
    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 = isSourceEmpty( sourceParams )
  global dataParams;

  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;

  ok = feval( ['def_' dataParams.type],'ok_to_load' );
  if( ~isempty(ok) )
    if( ~ok )
```

```
      ret = 1;
    end;
  end;



function ret = run_editSource( action, varargin )
%RUN_EDITSOURCE  User interface method file


  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_editSource('isopen') )
      ret = run_editSource('raise');
      return;
    end;

    fh = ui_editSource;
    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_editSource( 'update' );

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

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


   case 'update'
    if( ~run_editSource('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_editSource('isopen') )
      run_editSource( 'push_cancel' );
    end;

   case 'edit_dataSequence' % this affects all the calculated results
    a = str2num(get( edit_dataSequence, 'String' ));
    if( ~sum(a<=0) & ~sum(a>feval(['def_' dataParams.type], ...
                  'get_no_of_timeVar')))
      newDataParams.data_sequence = round(a);
      dataParamsModified = 1;
    end;

    run_editSource( 'update' );

   case 'push_setBackground'
```

```
    imageList = 1:feval(['def_' dataParams.type],'get_no_of_timeVar');
    %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 = loadSourceData( fh );
    if( a==0 ) return; end;
    figsValid = 1;
      end;
      if( ndims(figs)==3 )  % now set the new background
    img_background = figs(:,:,imgNum);
      else
    img_background = reshape(figs(:,imgNum),sourceParams.msize);
      end;

      run_editSource( 'push_autoBrightness' );
    end;

   case 'push_viewPoints'
    if( ~sourceValid ) return; end;
    no_images = feval(['def_' dataParams.type],'get_no_of_timeVar');
    %no_images = newDataParams.no_of_timeVar;
    msize = sourceParams.msize;
    run_image('display', struct('imgmax',newViewParams.maxImgVal), ...
          img_background );


    h = dlgInfo( 'Select a pixel from the image', '', [200 40 10] );
    run_image('raise');
    %figure( fh_image );
    [x,y] = ginput(1);  % user selects pixel from image now
delete( h );  %changed from close to delete 31.12.2004
    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 = loadSourceData( fh );
      if( a==0 ) 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 = feval(['def_' dataParams.type], 'get_timeaxis');
    %t = gettimeaxis;
    plot( t(newDataParams.data_sequence), dataPoints, '.' );
    xlabel( ['Coordinates: (' num2str(x) ',' num2str(y) ')'] );

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

    % if the source images are not loaded already, then load them now
    if( ~figsValid )
      try
    a = feval( ['load_' sourceParams.type] );
      catch
    a = 0;
      end;
      if( a==0 )
    closeWaitBars;
    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(['Alignment should be done only for images that' ...
         ' appear crooked.'], ...
        'Specify which 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 = loadSourceData( fh );
      if( a==0 ) return; end;
      figsValid = 1;
    end;
    newFigs = figs;

    % align using background image as base image
    h = waitbar( 0, 'Aligning images...' );
```

```
    for i=1:length(todo)
      if( i==1 )
    newFigs(:,:,todo(i)) = alignData( img_background, ...
                        figs(:,:,todo(i)));
      else
    newFigs(:,:,todo(i)) = alignData( [], ...
                        figs(:,:,todo(i)), 'reuse');
      end;
      waitbar((i-1)/(length(todo)-1));
    end;
    clear alignData;
delete( h );  %changed from close to delete 31.12.2004

    figsModified = 1;

   case 'sld_brightness'
    if( ~sourceValid ), 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( ~sourceValid ), 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_editSource('isopen') )

      if( (dataParamsModified | figsModified) & theMapSaveValid )
    a = questdlg( ['Existing data will be nullified!  Do you wish to'
...
                ' continue?'], 'Warning', 'No' );
    if( ~strcmp(a,'Yes') )
      return;
    end;
      end;

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

```
      if( viewParamsModified )
    viewParams = newViewParams;
    run_viewResults( 'update' );  %goes here when change background
image, but this causes OKerror 19.05.2005
      end;

      if( figsModified )
    figs = newFigs;
    theMapSaveValid = 0;
    doSaveSrcLocal = 1;
      end;

      % If dataParams or figs were modified, then they should NOT be
      % saved to the results file since the T1/T2 is void.  Things will
      % get saved once the T1/T2 map is calculated again.
      if( viewParamsModified & ~dataParamsModified & ~figsModified &
...
      ~isempty(saveFile) )
    disp( [ 'Saving to ' saveFile] );
    save( saveFile, 'viewParams', 'img_background', '-append' );
      end;
      run_editSource( 'close' );
      run_main( 'update' );
    end;

   case 'push_cancel'
    run_editSource( 'close' );

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

      delete( fh );
      disp( 'run_editSource 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_filterPixels( action, varargin )
%RUN_FILTERPIXELS  User interface method file


  include_globals;
  persistent fh mapParam errorParam TminCount TmaxCount PminCount;

  persistent txt_equation push_close list_tMin list_tMax ...
      list_errorMax edit_tMin edit_tMax edit_errorMax txt_t1range ...
      txt_errorRange txt_mapParam1 txt_mapParam2 edit_interRadius ...
      txt_TminAffected txt_TmaxAffected txt_PminAffected

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

  definitions;

  switch( action )

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

    fh = ui_filterPixels;
    setupUI( fh, mfilename );
    placeFig( fh, 'top', 'right' );   %%%%%%%%%changed 10.05.2005

    mapParam = varargin{1}; % e.g. 'T1', 'T2'

    obj_list = allchild(fh);

    % set up UI controls
    txt_equation = findobj( obj_list, 'Tag', 'txt_equation' );
    push_cancel = findobj( obj_list, 'Tag', 'push_cancel' );
    push_close = findobj( obj_list, 'Tag', 'push_close' );
    list_tMin = findobj( obj_list, 'Tag', 'list_tMin' );
    list_tMax = findobj( obj_list, 'Tag', 'list_tMax' );
```

```
    list_errorMax = findobj( obj_list, 'Tag', 'list_errorMax' );
    edit_tMin = findobj( obj_list, 'Tag', 'edit_tMin' );
    edit_tMax = findobj( obj_list, 'Tag', 'edit_tMax' );
    edit_errorMax = findobj( obj_list, 'Tag', 'edit_errorMax' );
    txt_t1range = findobj( obj_list, 'Tag', 'txt_t1range' );
    txt_errorRange = findobj( obj_list, 'Tag', 'txt_errorRange' );
    txt_mapParam1 = findobj( obj_list, 'Tag', 'txt_mapParam1' );
    txt_mapParam2 = findobj( obj_list, 'Tag', 'txt_mapParam2' );
    edit_interRadius = findobj( obj_list, 'Tag', 'edit_interRadius' );
    txt_TminAffected = findobj( obj_list, 'Tag', 'txt_TminAffected' );
    txt_TmaxAffected = findobj( obj_list, 'Tag', 'txt_TmaxAffected' );
    txt_PminAffected = findobj( obj_list, 'Tag', 'txt_PminAffected' );

    run_filterPixels( 'update' );

    ret = fh;

   case 'update'
    if( ~run_filterPixels('isopen') ) return; end;

    eqn = feval( ['def_', dataParams.type], 'showEqn');
    set( txt_equation, 'String', eqn );

    set( edit_tMin, 'String', num2str(viewParams.minTval) );
    set( edit_tMax, 'String', num2str(viewParams.maxTval) );
    set( edit_errorMax, 'String', num2str(viewParams.maxErrorVal) );

    set( txt_mapParam1, 'String', mapParam );
    set( txt_mapParam2, 'String', mapParam );

    set( txt_t1range, 'String', [mapParam ' range:' ...
           num2str(min(theMapSave(:))) ' to ' ...
           num2str(max(theMapSave(:)))] );
    errorParam = feval( ['def_', dataParams.type], 'get_errorParam');
    errorMap = pfittedSave(:,errorParam);
    set( txt_errorRange, 'String', ['Error range: ' ...
           num2str(min(errorMap)) ' to ' ...
           num2str(max(errorMap))] );

    set( list_tMin, 'String', strvcat(outRange_list{:}) );
    set( list_tMax, 'String', strvcat(outRange_list2{:}) );
    set( list_errorMax, 'String', strvcat(outRange_list{:}) );
    a = strmatch( viewParams.minTvalAction, outRange_list );
    set( list_tMin, 'Value', a );
    a = strmatch( viewParams.maxTvalAction, outRange_list2 );
    set( list_tMax, 'Value', a );
    a = strmatch( viewParams.maxErrorValAction, outRange_list );
    set( list_errorMax, 'Value', a );
    set( edit_interRadius, 'String', ...
            num2str(viewParams.interpolationRadius) );

    set( txt_TminAffected, 'String', [num2str(TminCount) ' pixels
affected']);
    set( txt_TmaxAffected, 'String', [num2str(TmaxCount) ' pixels
affected']);
    set( txt_PminAffected, 'String', [num2str(PminCount) ' pixels
affected']);
```

```
    case 'pixelCount'
     switch( varargin{1} )
      case 'Tmin'
       TminCount = varargin{2};
      case 'Tmax'
       TmaxCount = varargin{2};
      case 'Pmin'
       PminCount = varargin{2};
     end;
     run_filterPixels( 'update' );

    case 'edit_tMax'
     a = get( edit_tMax, 'String' );
     run_viewResults( 'edit_tMax', a );

    case 'edit_tMin'
     b = get( edit_tMin, 'String' );
     run_viewResults( 'edit_tMin', b );

    case 'edit_errorMax'
     c = str2num(get( edit_errorMax, 'String' ));
     if( ~isempty(c) )
       viewParams.maxErrorVal = c;
     end;
     set( edit_errorMax, 'String', num2str(viewParams.maxErrorVal) );
     run_viewResults( 'update' );
     run_viewResults( 'updateFig' );

    case {'list_tMin', 'list_tMax', 'list_errorMax'}
     viewParams.maxTvalAction = outRange_list2(get(list_tMax,'Value'));
     viewParams.minTvalAction = outRange_list(get(list_tMin,'Value'));
     viewParams.maxErrorValAction =
outRange_list(get(list_errorMax,'Value'));

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

    case 'edit_interRadius'
     c = str2num(get( edit_interRadius, 'String' ));
     if( ~isempty(c) )
       if( c<=.5 )
     viewParams.interpolationRadius = .5;
       else
     viewParams.interpolationRadius = c;
       end;
     end;
     set( edit_interRadius, 'String', ...
               num2str(viewParams.interpolationRadius));
     if( strcmpi(viewParams.maxTvalAction, 'interpolate') | ...
     strcmpi(viewParams.minTvalAction, 'interpolate') | ...
         strcmpi(viewParams.maxErrorValAction, 'interpolate' ))
       run_viewResults( 'update' );
       run_viewResults( 'updateFig' );
     end;

    case 'push_close'
```

```
    run_filterPixels( 'close' );

   case 'close'
    if( run_filterPixels( 'isopen' ));
      delete( fh );
    end;

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

```
function ret = run_image( action, varargin )
% RUN_IMAGE   Image display methods
%
%   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 structure, [] to use defaults.  To display only
%   BACKGROUND_IMG, omit MAP_IMG.
%
%   The fields in OPTIONS are:
%   'imgmax' --   input image max intensity
%   'mapmax' --   input map maximum intensity
%   'mapmin' --   input map minimum 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)
                        %%is it ok for totalColors to be 512 if matrix
is
                        %%256x256????  18.05.2005

  args = varargin;
  if( nargin>1 )
      %varargin  %display varargin for debug 10.01.2005
```

```
      %--maybe handle is not in varargin{1}, but it could be in a
different location/address
  lengthVarargin=length(varargin);
  for i=[1:lengthVarargin]
  if( ischar(varargin{i}))
      if( strcmp('handle', varargin{i}))
          %disp(['I recognize that a handle alternate to the default
should be used for display: ' varargin{i+1}]);
    h = varargin{i+1};
    args = {varargin{1:i-1} varargin{i+2:lengthVarargin}};  %not sure
about this definition
    isMainFigure = 0;
      end; %if
    end;  %if
  end; %for
     end; %if

  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', 'manual' );
    set( h, 'Name', 'Image' );
    set( h, 'DeleteFcn', ['delete(' num2str(h) ');axis off;'] );
    %changed from close to delete 06.01.2005
    placeFig( h, 'center', 'right' );
    ret = h;

   case 'isopen'
       %disp('in run_image, case isopen')
    if( isempty(h) ) ret = 0;       % if h = ''
        %disp('my so-called handle was empty')
    elseif( ~ishandle(h) ) ret = 0; % if h is invalid handle
        %disp('my so-called handle was not a handle')
    else
        %here, run_image only allows figures with the name image
        %to have things displayed in them.  why then, does it bother
        %having the option of a handle input???
        %let's see if it still functions if I take this out.
11.01.2005
      %if(strcmp( get(h, 'Name'), 'Image' ))
    ret = h;
    %disp('i am passing on the same handle that i was given')
      %else
```

```
    %ret = 0;
      %end;
    end;

   case 'raise'
       %disp('in run_image, case raise')
    if( run_image( 'isopen', 'handle', h ))
      figure(h);
      ret = h;
    else
      ret = 0;
    end;

   case 'close'
    if( run_image( 'isopen', 'handle', h) )
      delete(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.  Note: use with axis handle hasn't been tested yet.

    createNewFig = 0;
    saveZoom = 0;

    set(0,'units','pixels');           %screen specifications
pix_ss=get(0,'screensize');
swidth=pix_ss(3);
sheight=pix_ss(4);
%disp('screensize determined'); %for debug purposes

%something is wrong in here, because it's not using the specified
%handle to display the figure.  (that's called from run_main, line 307
    if( isvalidfigure(h) )
      if( run_image( 'raise', 'handle', h ))
    if(~isempty(get(h,'CurrentAxes')))  saveZoom = 1; end;
      else
    createNewFig = 1;   % if raise failed, then need new figure
    %disp('raise failed so I need to create a new figure')
      end;
    else
      if( isaxeshandle(h) ) % if the handle looks like it's for an axis
          %disp('ok, handle I have is for an axis')
    if( isvalidaxes(h) )
      axes(h)       % ok, so make it current
    else
        %disp('hmmm...axis does not exist')
      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;
    if( saveZoom )
      prevXlim = xlim;
      prevYlim = ylim;
    end;

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

    % parse arguments ..........................................
    OPTIONS = args{1};   %this is not quite right. because args was
defined to be varargin{3:the end}
    bgImg = args{2};
    %disp('bgImg has the following dimensions')  %debug 14.02.2005
      %size(bgImg)%debug 14.02.2005
    if(length(args)>2)
      mapImg = args{3};
     % disp('mapImg, args3, has the following dimensions')
      %size(mapImg)  %debug 14.02.2005
    else
      mapImg = '';
      %disp('mapImg, from blank')   %%22.05.2005
    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
      if( isfield(OPTIONS,'mapmin') )
    mapMinVal = OPTIONS.mapmin;
      else
    mapMinVal = 0;
      end;
      mapscale = totalColors / (mapMaxVal-mapMinVal);
    end;

    if( isfield(OPTIONS,'colormap') )       % set colormap
      colormapType = OPTIONS.colormap;
```

```
      switch( colormapType )
       case 'green'
    cm = greenColormap(totalColors);
       case 'hsv'
    cm = hsvColormap(totalColors);
       case {'default', ''}
    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);
    %disp('tempImg1 has the following dimensions')  %debug 14.02.2005
    %size(tempImg)%debug 14.02.2005
    clear bgImg;

    if( isempty(mapImg))
        %disp('mapImg is empty');
      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
        %disp('mapImg is not empty');
      % overlay with map ........................................
      mapImg = round( (mapImg-mapMinVal)*mapscale );    % rescale
      mapImg = max( mapImg, 0 );        % clip dark
      mapImg = min( mapImg, size(cm,1) );   % clip light
      size_mapImg=size(mapImg);   %%debug 22.05.2005
%       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
      %disp('mapImg has the following dimensions before columnization')
      %size(mapImg)  %debug 14.02.2005
      mapImg = mapImg(:);
      %disp('mapImg has the following dimensions before find')
      %size(mapImg)  %this is 256*256*4 in the case of despot phantom
data with two images 14.02.2005
      %actually it seems to be 512 by 512, which is wierd because I
thought
      %that the data were 256 by 256
      [mapi j vals] = find(mapImg);  %mapImg is 256x256 during ROI
definition
      %but then is 512x512 when i press make Map.  22.05.2005
      %disp('mapi has the following dimensions')  %debug 14.02.2005
```

```
      %size(mapi)
      %disp('mapi has the following max value')  %debug 14.02.2005
      %max(mapi)
      %disp('j has the following dimensions')
      %size(j)
      %disp('j has the following max value')
      %max(j)  %14.02.2005 both j and mapi should be indices.  mapi
should not be greater than dim, but j could account for trailing
dimensions.
      %disp('vals has the following dimensions')  %debug 14.02.2005
      %size(vals)
      clear mapImg j;
      %disp('tempImg2 has the following dimensions')  %debug 14.02.2005
      %size(tempImg)%debug 14.02.2005
      tempImg = tempImg(:);
      display_r = tempImg;
      %disp('display_r1 has the following dimensions')  %debug
14.02.2005
      %size(display_r)%debug 14.02.2005
      size_CM=size(cm);
      size_mapi=size(mapi);   %mapi becomes too large and max value is
too large 22.05.2005
      max_mapi=max(mapi);
      size_vals=size(vals);
      display_r(mapi) = cm(vals,1);  %this line is changing dimensions
in a wrong way.  14.02.2005
      %14.02.2005  mapi and vals both come from line 247
      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 );
      %disp('display_r2 has the following dimensions')  %debug
14.02.2005
      %size(display_r)%debug 14.02.2005
      %disp('dims are the following:')%debug 14.02.2005
      %dims%debug 14.02.2005
      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;

      % save color scale
      % This is used by colorbar, which is limited to 256 colors.
      figSettings = struct( ...
      'colorbarScale',    256/(mapMaxVal-mapMinVal), ...
      'colorbarStartVal', mapMinVal );
      set( gcf, 'Userdata', figSettings );
    end

    % make figure/axis current
    if( isvalidfigure(h) )
      figure(h);
      % colormap command requires color map to be 256 in size or less
      if max(size(cm))>256   %if statement added 10.01.2005
      %10.01.2005 here, colormap has only two dimensions (is that
right?)
```

```
      colormap( cm(1:totalColors/256:totalColors,:));
      else
          colormap(cm);
      end
      %control size of figure array: [left, bottom, width, height]
      set(h,'Position',[swidth/2, sheight/8, swidth/2, sheight/2]);
      %sizeoffigurecontrolled=1   %debug puposes
    elseif( isvalidaxes(h) )
      axes(h);
      %controls size of axis (0,0) is lower left of figure. (1,1) is
upper
      %right
      set(h,'position',[.125,.125,.75,.75]);
      %sizeofaxiscontrolled=1   %debug purposes
    end;

    % restore zoom
    imshow( display_img );
    if( saveZoom )
    zoom(1);
    xlim( prevXlim );
    ylim( prevYlim );
    end;
    ret = 1;
    %hpos=get(h,'position')  %debug purposes

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

   case 'colorbar'
    overlaycolorbar( h );
    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  User interface method file


  include_globals;
  persistent push_dataSource push_editSource txt_sourceDir txt_magnet
...
      push_ROIlist push_loadResults txt_ROImsg push_viewResults ...
      txt_resultsSaved push_close push_openImage push_browseImages
push_orderImages
  global fhm childs;  %13.01.2005 made childs global
```

```
  ret = '';

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

    % set up a whole slew of variables
    sourceValid = 0;
    figsValid = 0;
    ROIlistValid = 0;
    theMapSaveValid = 0;
    doSaveSrcLocal = 0;

    fhm = ui_main;
    axis off;  %don't want axis in main window  added 20.05.2005
    placeFig( fhm, 'top', 'left' );
    setupUI( fhm, mfilename );
    run_main_setup_closereq=1;
    ret = fhm;

    % Set up event callbacks.

    % set up UI controls
    childs = allchild(fhm);
    push_dataSource = findobj( childs, 'Tag', 'push_dataSource' );
    push_editSource = findobj( childs, 'Tag', 'push_editSource' );
    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' );
    push_close = findobj( childs, 'Tag', 'push_close');
    push_openImage = findobj( childs, 'Tag', 'push_openImage' );
    push_browseImages = findobj( childs, 'Tag', 'push_browseImages' );
    push_orderImages = findobj( childs, 'Tag', 'push_orderImages');

    run_main( 'update' );

   case 'push_close'
       %disp('push_close depressed') %for debug 06.010.05
       push_close = findobj( childs, 'Tag', 'push_close');
    run_main('close');

   case 'close'
       disp( ['In run_main, case close, Saving to ' saveFile] );
%26.01.05 saving added
      save( saveFile, 'sourceParams', 'dataParams', 'viewParams', ...
        'ROIlist', 'theMapSave', 'theMapi', 'pfittedSave', ...
        'img_background' );
    run_dataSource( 'close' );
    run_ROIlist( 'close' );
    run_editSource( 'close' );
    %viewResults should not need to be closed since it is not a
seperate
    %window
```

```
    %run_viewResults( 'close' );
    run_image('close'); %close primary figure added 13.01.2005
    if( run_main( 'isopen' ));
        %disp('main is open and is about to be closed');  %for debug
06.010.05
      delete( fhm );
      disp( 'run_main closed' );    % for debugging
      close all;  %closes any comparison images/other windows added
13.01.2005
    end;
    %disp('all variables being cleared')
    %clear all;   %this is just annoying because I can't work with
    %variables after I close it.

   case 'isopen'
    if( isempty(fhm) ) ret = 0;     % fhm = ''
    elseif( ~ishandle(fhm) ) ret = 0;   % fhm = invalid handle
    else
        %disp('about to compare name of figure with "Main"'); %for
debug 06.010.05
      ret = (strcmp( get(fhm, 'Name'), 'Main' ));
    end;

   case 'raise'
    if( run_main('isopen') )
      figure(fhm);
      axis off;  %added 20.05.2005  don't want axis in background of
figure
      ret = 1;
    else
      ret = 0;
    end;

   case 'update'

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

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

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

```
    % Region of Interest panel ..............................
    if( sourceValid )
      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' )
      set( push_openImage, 'Enable', 'On' )   %11.01.2005
    else
      set( push_viewResults, 'Enable', 'Off' )
      %set( push_openImage, 'Enable', 'Off' )   %11.01.2005
      %keep push_openImage open all the time
    end;
    if( run_viewResults( 'isenabled' ))
      if( theMapSaveValid==0 )
    %run_viewResults( 'close' );  %no need because it is attached to
main
  %13.01.2005
      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) | ~theMapSaveValid )
      set( txt_resultsSaved, 'String', 'No calculation results' );
    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_editSource'
     run_editSource( 'init' );

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

    case 'push_loadResults'
     %if( ~isempty(figs))
     if( sourceValid )
       temp = questdlg( ['This action will replace the data already
loaded!' ...
            '  Do you wish to continue?'], ...
               'Warning', 'Yes', 'Cancel', 'Cancel' );
       if( ~strcmp(temp,'Yes'))
           return;
       else  %if it is 'yes' make sure to save first  26.01.05
           disp( [' In run_main, case close, Saving to ' saveFile] );
%26.01.05 saving added
       save( saveFile, 'sourceParams', 'dataParams', 'viewParams', ...
         'ROIlist', 'theMapSave', 'theMapi', 'pfittedSave', ...
         'img_background' );
       end; %if
     end; %if
     [filen dirn] = uigetfile( '*.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 ' is not a valid results file'], ...
           'Error', 'modal' );
     return;
       end;

       run_dataSource( 'close' );
       run_editSource( 'close' );
       run_ROIlist( 'close' );
       %run_viewResults( 'close' );  %no point in closing
run_viewResults
       %when it's right there attached to main

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

       disp( ['  Loading for analysis: ' saveFile] );
```

```
      load( saveFile, 'sourceParams', 'dataParams', 'viewParams', ...
        'ROIlist', 'theMapSave', 'theMapi', 'pfittedSave', ...
        'img_background' );
      % we do this to maintain backwards compatibility with older
      % saved files that do not contain newer fields
      if isfield(viewParams,'ROILabels')  %26.01.05 formed into if
statement
          %disp('i think that ROILabels is a field in viewParams')
%was
          %for debug
          default_viewParams = def_view( 'new' );
      else
         %disp('i think that ROILabels is not a field in viewParams')
%was
         %for debug
          default_viewParams = def_view( 'default' );
      end
      %viewParams
      %default_viewParams
      viewParams = structcombine( default_viewParams, viewParams );

      sourceValid = 1;
      ROIlistValid = 2;
      theMapSaveValid = 2;

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

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

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

    end;

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

case 'push_openImage'  %added 10.01.2005
    %to open background-type image (not fitted map!!) to help
segmentation
```

```
    %by providing a comparison image.  for example, in doing dGEMRIC
    %fitting, a T2 image would be helpful, since cartilage contrast is
    %better in T2 images.
  %disp('push_openImage activated');
  %use run_image with option to display background image only
  %use a new handle to open this auxiliary image
  haux=figure('Name','Comparison Image');
  %want colormap to be gray (a grayscale colormap)
  OPTIONS=struct('colormap', 'gray'); %options is a structure array
  %to find comparison image, use uigetfile
  [fileName pathName] = uigetfile( {'*.*';'*.MR'}, 'Open Comparison
Image' );
  comparison_image_name=fullfile(pathName,fileName);  %changed from
strcat to fullfile 06.02.2005
  disp( ['  Loading as comparison image: ' comparison_image_name] );
  %comparison_image=dicomread(comparison_image_name);  %this would read
only dicom files
  %to read non-dicom files
  %use the code from load_siemens added 10.01.2005

  %use sourceParams.byte_ordering assuming the byte ordering of
comparison
  %image is same as loaded image and that loaded image is already
loaded
  endian = sourceParams.byte_ordering(1);

  [fid, msg] = fopen( comparison_image_name, 'r', endian);
    if( fid==-1 )
      lasterr( ['Could not open ' comparison_image_name '.  ' msg] );
      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');
    %disp(['found a string: ' str])  %10.01.2005
    if( strcmp(str, 'DICM'))
        %disp('string indicates file is dicom') %10.01.2005
      fclose( fid );
      %fid=fopen(filen,'r');
      comparison_image = dicomread(comparison_image_name);
    else
    %disp('string indicates file is not dicom')  %10.01.2005
    % here's a more robust way to find the offset ....................
    finfo = dir( comparison_image_name );
    msize = sourceParams.msize;  %putting this here assumes several
things:
    %it assumes:
    %1. that the image to be analyzed is loaded first
    %2. that the image to be analyzed has the same matrix size as the
    %comparison image.
    offset = finfo.bytes - (msize(1)*msize(2)*2);
    fseek(fid,offset,-1);

    %figs(:,:,n)=fread(fid,[msize(1),msize(2)],'short');
```

```
    comparison_image=fread(fid,[msize(2),msize(1)],'uint16');
    fclose(fid);

    comparison_image = comparison_image';
    end  %end if
  comparison_image=double(comparison_image);  %class(comparison_image)
  %must be double in order to go through run_image ok.
  %for now, only works with dicom images
  %image must be loaded into
  %the workspace, otherwise, just dealing with the string that is the
  %image's address.
  sizeComparison_image=size(comparison_image);  %try to get a sense for
  %dimensions of background images, because run_image might be trying
to
  %see the three dimensions of a full-color image
  %disp('about to display auxiliary image which has been selected');
  ret=run_image('display',OPTIONS,comparison_image,'handle',haux);
  %comparison image needs to be after options, and handle needs to be
last
  %right now, loading results after opening comparison image deletes
the
  %comparison image.  10.01.2005
  %also, loading comparison after loading results, displays the
comparison
  %over the results.  (although it does open a new figure and call it
  %comparison image; it just doesn't put the comparison image there.)

      case 'push_browseImages'  %case added 07.03.2005
          %disp('push_browseImages activated')
          disp('Please select all images to be browsed')
          [fileName pathName] = uigetfile( {'*.*';'*.MR'}, 'Select
Images in Series to be Browsed', 'MultiSelect', 'on' );
          %multiselect "on" returns one fileName per column in one long
          %row.
          %cannot use fullfile if multiselect is on.
          %browse_image_1_name=fullfile(pathName,fileName)
          %generates an error if user presses cancel inside the ui
          n_images=length(fileName);
          for i=1:n_images
              browse_image_name{i}=fullfile(pathName,fileName{i});
%fileName is a cell array
          end
          %see run_viewFigs for how to make the files into something
that
          %you can play through.
          %make a call to run_viewFigs?
          %would have to tell it to take in different data than it's
used
          %to.  It might be good, though, since it would be nice to
play
          %the movie differently (maybe not play it at all?) and I
would
          %only have to do that once, if I make a call to run_viewFigs
          %here.
          %call as run_viewFigs(action,varargin) where varargin is a
          %structure with a data_sequence field
          browseDataParams=dataParams;
```

```
          browseDataParams.data_sequence=1:n_images;  %this should be
enough to make browseDataParams a structure
          eBDP=isempty(browseDataParams);
          ebin=isempty(browse_image_name);

run_viewFigs('init_browse',browseDataParams,browse_image_name);

      case 'push_orderImages'
          Rename_DICOM;

      otherwise
       disp([action ' unrecognized by run_main being ignored']);
%07.01.2005
       %run_viewResults(action);   %%% 22.05.2005
    % ignore unrecognized actions

  end; %switch



function ret = run_ROIlist( action, varargin )
%RUN_ROILIST  User interface method file


  include_globals;
  persistent fh itemCount newROIlist;
  persistent push_addROI push_delROI push_addEntire list_ROIlist ...
      push_loadFile push_exe push_close txt_message chk_square
edit_square ...
      chk_saveChanges chk_highlight;
  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;
    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'
);
    chk_highlight = findobj( allchild(fh), 'Tag', 'chk_highlight' );

    baseMaxImgVal = viewParams.maxImgVal;

    if( ROIlistValid & ~isempty(ROIlist) & ...
    theMapSaveValid & ~isempty(theMapSave) )
      % allow display of ROIs using map values rather than green
highlight
      set( chk_highlight, 'Value', 1 )
      set( chk_highlight, 'Enable', 'On' )
    else
      set( chk_highlight, 'Value', 1 )
      set( chk_highlight, 'Enable', 'Off' )
    end;

    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( ~sourceValid )
    %  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'   %change list_ROIlist, 'string',
          %to be an array rather than a string 05.01.2005
          set(fh,'Pointer','watch');
          if( isempty(newROIlist) )
              set( list_ROIlist, 'String', {}  );
          else
              tempStr={};
              %tempStr = {'ROI label' 'pixels' 'mean' 'std. dev.'};
              for ROILabel = 1:length(newROIlist)
                  pixelCount = length(newROIlist(ROILabel).maski);
                  if( newROIlist(ROILabel).modified ), stat = '
(new)';
                  else, stat = '';
                  end; %if
                  if length(viewParams.ROILabels) < ROILabel
                      tempStr = vertcat(tempStr, ...
                          {ROILabel '    ' pixelCount stat});
```

```
                    viewParams.ROILabels{ROILabel}=ROILabel;
%26.01.05 extend Labels list as ROIs added
                else
                    tempStr = vertcat( tempStr, ...
                        {viewParams.ROILabels{ROILabel} '    '
pixelCount stat});  %26.01.05 references global
                end; %if
            end; %for

            %added 06.01.05 to make sure that array displays properly
in listbox
            % Ensure that all elements of cell array are strings for
display
            for i=1:numel(tempStr) % Goes through all elements
                if ~iscellstr(tempStr(i))
                    tempStr{i}=num2str(tempStr{i});
                end  %if
                charLength(i)=length(char(tempStr{i}));
            end  %for
            % Insert spaces to separate the strings for display
purposes
            maxCharLength=max(charLength);
            if maxCharLength>8   %26.01.05 labels were getting to be
30 long
                maxCharLength=8;
            end
            maxCharLength_withBuffer=maxCharLength+2;
%%%%%%%%%added 05.05.2005
            for i=1:numel(tempStr)
                if length(char(tempStr(i))) >
maxCharLength_withBuffer   %added to deal with labels to long 27.01.05
                    myEntry=char(tempStr(i));
                    len=length(myEntry);
                    myEntry=myEntry(len-
maxCharLength_withBuffer+1:len);  %use last N characters, because space
padding is at the beginning.
                    tempStr{i}=myEntry;
                end %if
                while length(char(tempStr(i))) <
maxCharLength_withBuffer
                    tempStr(i)=strcat({' '},tempStr(i));
                end  %while
            end   %for
            tempStr=strcat(tempStr,{'  '});   %why is this done?
27.01.05
            % Concatenate the cell array of strings
            tempStr=cell2mat(tempStr);

            set( list_ROIlist, 'String', tempStr );
        end;
        showROIImage(newROIlist, list_ROIlist, chk_highlight );
        set(fh,'Pointer','arrow');

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

```
   case 'warning' % This means ROIlist is potentially dirty due to
                  % modification elsewhere.
    if( run_ROIlist('isopen') )
      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 'chk_highlight'
    run_ROIlist( 'update' );

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

    if( ~run_image('isopen') )
      showROIImage(newROIlist, list_ROIlist, chk_highlight);
    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 = sourceParams.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) )
      if( isempty(newROIlist) )
    newROIlist =  struct( 'maski', maski, 'modified', 1);
      else
    newROIlist = [newROIlist ; struct( 'maski', maski, 'modified', 1)];
      end;
      run_ROIlist( 'update' );
```

```
    end;
    figure( fh );
    set(fh,'Pointer','arrow');

   case 'push_addEntire'
    set(fh,'Pointer','watch');
    mask = ones( sourceParams.msize );
    maski = find(mask(:));  % indices of non-zero's in mask
    if( isempty(newROIlist) )
      newROIlist =  struct( 'maski', maski, 'modified', 1);
    else
      newROIlist = [newROIlist ; struct( 'maski', maski, 'modified',
1)];
    end;
    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( '*.mat', ['Select file to load ROI' ...
            ' set from' ] );
    if( filen==0 ) return; end; % cancelled
    s = load( [dirn filen], 'ROIlist', 'sourceParams', 'dataParams' );
    % check if ROIlist exists in the file
    if( ~isfield(s, 'ROIlist') )
      % load didn't work
      errordlg( [filen ' does not contain a ROI set'], 'Error', 'modal'
);
      return;
    end;

    % check if the msize in the file is the right size
    if( isfield(s.sourceParams, 'msize' ))
      filemsize = s.sourceParams.msize;
    elseif( isfield(s.dataParams, 'msize' ))
      filemsize = s.dataParams.msize;
    else
      filemsize = [0 0];
    end;
    if( ~prod(double(filemsize==sourceParams.msize )))
      errordlg( ['Matrix size in ' filen ' is incompatible with' ...
         ' current data'], 'Error' );
      return;
    end;

    % loaded okay
```

```
    % 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
    if( isempty(newROIlist) )
      newROIlist = s.ROIlist;
    else
      newROIlist = [newROIlist; s.ROIlist];
    end;
      end;
    end;

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

   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)
      viewParams = def_view( 'new' );       %reset viewParams
%07.02.2005 changed from 'default' to 'new
      theMapSave = zeros(sourceParams.msize);
      theMapi = '';
      theMapi = double(theMapi);
      %how to initialize an array in a non-character manner?
      pfittedSave = '';
      pfittedSave = double(pfittedSave);
    end;

    if( isempty(newROIlist)), return;, end;

    disp( 'execute now' );

    if( isempty(saveFile) )     % saveFile is a global variable
      saveFile = getSaveFile;
      state_saveChanges = 1;
    end;

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

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

    removei = intersect(maski,theMapi);     % Don't recalculate pixels
    if( ~isempty(removei) )          %  that have already been
      mask(removei) = 0;             %  calculated
    end;

    if( sort(ismember(maski,theMapi))==1 )  % New ROI falls entirely
                                            %  within existing
ROIs?
      ROIlist = newROIlist;     % commit newROIlist to actual ROIlist
                  % Don't need to do anything else

    else                % New ROI has some uncalculated pixels
      if( ~figsValid )
    a = loadSourceData( fh );
    if( a==0 ) return; end;
    figsValid = 1;
      end;

      ROIlist = newROIlist;     % commit newROIlist to actual ROIlist


      % run calculations now ------------------------------------------
---
      %if( isempty(which('fminsearch')) )
      %  [newMap, newMapi, newPfitted] = feval( ['exe_old_'
dataParams.type], ...
      %                  mask );
      %else
```

```
      [newMap, newMapi, newPfitted] = feval( ['exe_' dataParams.type],
mask );
      %end;
     %newPfitted100=newPfitted(100)
      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
    if( isempty(theMapi) )
      isNewMap = 1;
    else
      isNewMap = 0;
    end;
    theMapi = [theMapi ; newMapi];      % new theMapi -- indices
    %class PFittedSave here is char
    %classPFittedSave=class(pfittedSave)
    %%numel(PFittedSave100)=0
    %PFittedSave100=pfittedSave(100)
    %classNewPFitted=class(newPfitted)
    pfittedSave = double([pfittedSave ; newPfitted]);
    theMapSave(newMapi) = newMap(newMapi);  % 64x64 (e.g.) grid

    % do some masking of pixels (a la filterPixels.m)

    if( isNewMap )      % completely new T1 calculation?
      viewParams.maxTval = getMapMax(theMapSave);   % max T1 displayed
      viewParams.maxErrorVal = Inf;         % max error allowed
    else
      if( strcmp(viewParams.maxTval, 'auto') )
        viewParams.maxTval = getMapMax(theMapSave);
      end;
    end;

      end; %if ~isempty(newMapi)

    end; % if( isempty( mask )), else

    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 a local copy of the source data, unless the file already
    % exists. (doSaveSrcLocal flag forces save, e.g. after aligning
    % images.)
    [pathn,name,ext] = fileparts(saveFile);
```

```
    figsFile = [pathn '/' name '-src.mat'];
    if( doSaveSrcLocal | isempty(dir(figsFile)) )
      disp( [' Saving figs to ' figsFile] );
      save( figsFile, 'figs' );
      doSaveSrcLocal = 0;
    end;

    ROIlistValid = 2;       % valid & modified flag
    theMapSaveValid = 2;        % valid & modified flag
    run_ROIlist( 'close' );
    %enable_viewResults(fhm);     %added 22.05.2005  %prevents full
    %run_viewResults initiation.???????????????
    run_viewResults( 'init' );
    run_main( 'update' );

   case 'push_close'
    % Pushing the close button updates things.
    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( '*.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, chk_highlight)
  include_globals;
  useHighlight = get( chk_highlight, 'Value' );

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

    % highlight the selected ROI
    selectedROI = get( list_ROIlist, 'Value' );
    if( ~isempty(selectedROI) )
      unionMaski = unionall( newROIlist(selectedROI).maski, [] );
      if( useHighlight )
    unionMask(unionMaski) = img_background(unionMaski)*3;
      else
    unionMask(unionMaski) = theMapSave(unionMaski);
      end;
    end;
  else
    unionMask = '';
  end;
  if( run_image('raise') )  % preserve figure window position
    savePosition = get( gcf, 'Position' );
  else
    savePosition = '';
  end;
  if( useHighlight )
    colorMapType = 'green';
  else
    colorMapType = 'default';
  end;
  disp('putting in unionMask instead of MapImg')
  unionMask_size=size(unionMask)
  run_image( 'display', ...
         struct( 'imgmax', viewParams.maxImgVal, ...
             'colormap', colorMapType ), ...
         img_background, unionMask );
  if( ~isempty(savePosition) )
    set(gcf,'Position',savePosition);
  end;



function ret = run_viewFigs( action, varargin )
%RUN_VIEWFIGS  Method file for viewing 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 dataParams
```

```
  persistent fh ah_mov mov sld_mov push_play browseDdataParams rad_old
...
      rad_new push_undo f frame_axis fileNames browse_flag num_frames
browseFigs ...
      figInstanceNumber
  %make browseFigs global?
  ret = '';
  if ~exist('browse_flag') %added 23.03.2005
  browse_flag=0;  %added 23.03.2005
  end
  switch( action )
      case 'init_browse'  %added 23.03.2005
          browse_flag=1;
          %varargin  %debug 23.03.2005
          dataParams=varargin{1};  %added 23.03.2005   %dataParams
should be global
          %disp('dataParams come via a call to init_browse in
run_viewFigs');
          %changed to browseDataParams 16.05.2005 %%unchanged
          fileNames=varargin{2};  %added 23.03.2005
          fileNames=sort(fileNames);  %make sure names are ordered
alphanumerically
          %actually want to sort fileNames based on InstanceNumber
          %dataParams = varargin{1} %debug 23.03.2005
          %if you are browsing, need a different make_movie case
          %also, need to send in the addresses of the images to browse.
          %that can be in varargin{2}
          %disp('I am in run_viewFigs, init_browse')
          run_viewFigs('init');

   case 'init'
    %disp('in run_viewFigs, init')  %23.03.2005 debug
    if( feval(mfilename,'isopen') ) return; end;  %23.03.2005 what is
mfilename?
    %varargin %debug 23.03.2005
    %varargin{1} %debug 23.03.2005
    %varargin{2} %debug 23.03.2005
    %browse_flag
    if ~browse_flag  %added 23.03.2005
        %disp('dataParams come via a call to init in run_viewFigs');
     dataParams = varargin{1};
    end
    f=figure( 'Name', 'Image Frames',...
        'NumberTitle', 'off');
    set( f, 'DeleteFcn', 'run_viewFigs(''close'');' );
    frame_axis=axes('Parent',f);   %frame_axis axis in which individual
frames are seen
        %,'Visible', 'off');
    %sld_test = uicontrol( 'Parent', f, ...
             %'Position',[20 20 100 15], ...
             %'Style','slider', ...
             %'Tag','sld_mov1');
    %plot(1:1:10,'Parent',f);
     %disp('in run_viewFigs, init, line 49')  %23.03.2005 debug
    fh = figure( 'Name', 'Image Controls', ...   %name changed from
Image Movie 11.04.2005
             'NumberTitle', 'off' );
```

```
        axis off;  %added 20.05.2005 don't want axis in image controls
background
    set( fh, 'DeleteFcn', 'run_viewFigs(''close'');' );
    ret = fh;
    %disp('in run_viewFigs, init, line 54')  %23.03.2005 debug

    num_frames = length(dataParams.data_sequence);
    if( isempty(mov) )
        is_empty_mov=1;
        if browse_flag
            run_viewFigs('makeMovieBrowse',varargin);  %added
23.03.2005
        else
        run_viewFigs( 'makeMovie' );
        end
    else is_empty_mov=0;
    end;

    % set up the rest of the movie window

    %disp('displayed num_frames in init')  %debug 23.03.2005
    %title( ['Data ' num2str(dataParams.data_sequence(1))] );
    %ah_mov = axes('Parent',fh);   %ah_mov axis in which movie plays
%commented out 11.04.2005
                  %, 'Visible', 'off');

    %mapos=get(ah_mov,'Position');  %position and size of movie axis
    fapos=get(frame_axis,'Position');  %position and size of frame axis
    cfpos=get(fh,'Position');        %position and size of movie figure
window %controls figure window 11.04.2005
    ffpos=get(f,'Position');      %position and size of frame figure
window
    %sets frame axis to be position of frame axis and size of movie
axis
    %set(frame_axis, 'Position', [fapos(1) fapos(2) mapos(3)
mapos(4)]);
    %%commented out 11.04.2005
    %sets movie figure window to be next to frame figure window and
330x314
    %does this need to be wider and taller if images are 512x512? (and
not
    %256x256)
    fh_h=30;
    fh_w=ffpos(3);
    set(fh,'Position',[ffpos(1) ffpos(2)-fh_h-30 fh_w fh_h]);
%commented out
    %11.04.2005
    %sets frame figure window to be same location and 330x314
    %set(f, 'Position',[ffpos(1) ffpos(2) 330 314]);  %commented out
    %11.04.2005

    sld_mov = uicontrol( 'Parent', fh, ...
              'Position',[10 10 200 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 browse_flag  %if you are not browsing, you don't need this button
    push_select = uicontrol( 'Parent', fh, ...  %added to select image
to analyze  11.04.2005
               'Position', [210 10 160 15], ...
               'String', 'Select Image for Analysis', ...
               'Callback', 'run_viewFigs(''select'');' );
end %if browse_flag

    if( ~isempty(newFigs) )
      push_undo = uicontrol( 'Parent', fh, ...
                 'Position', [ 210 10 100 15], ...
                 'String', 'Undo align', ...
                 'Callback', 'run_viewFigs( ''undoAlign'');');
    end;

    %run_viewFigs('play');  %commented out 11.04.2005
    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;
    fhtemp=figure;

    %%downsample movie to fit completely within frame if larger than
256x256
    %[height width blah depth]=size(X);
%     for i=[1:depth]
%     Y=X(:,width*(i-1)+[1:width]);
%     if (height>256)
%         Y=downsample(Y,height/256);
%     end
```

```
%     if (width>256)
%         Z=downsample(Y',width/256);
%         Y=Z';
%     end
%     X2([1:256], [1:256], blah, i)=Y;
%     end

    mov = immovie(X,gray(256));
    delete(fhtemp);
    %clear X2;
    ret = 1;

      case 'makeMovieBrowse' %added 23.03.2005
          %disp('in run_viewFigs, makeMovieBrowse');
          %num_frames
          %dataParams.data_sequence
          h_wait = waitbar(0,'Loading files...');
          for n=1:num_frames
              fileName=char(fileNames(n));
              %class(fileName)
          browseFigs(:,:,n)=dicomread(fileName);
          info_temp=dicominfo(fileName);
          figInstanceNumber(n)=info_temp.InstanceNumber;
          waitbar(n/num_frames,h_wait)
          end
          close(h_wait)
    imgMax = getImgMax(browseFigs(:,:,floor(num_frames/2)));
    %best constrast for slices in middle of knee (as opposed to first
or last slice)
    X(:,:,1,:) = uint8(browseFigs(:,:,dataParams.data_sequence) / ...
              imgMax*256);
    fhtemp=figure;
    mov = immovie(X,gray(256));
    delete(fhtemp);
    ret = 1;

      case 'select'  %added 11.04.2005
          %if confirmChange  %%added 23.05.2005 to check to make sure
user wants to clear existing data
            conCh=confirmChange   %%added/changed 23.05.2005
            if conCh
          %disp('in run_viewFigs, select');
          frame_no = round(get(sld_mov,'Value'));
          disp('you have selected:')
          fileName_selected=fileNames(frame_no)   %try using function
cellstr
          %open run_dataSource with information from selected file so
that
          %user can load similar files for analysis
          %classFileName_selected=class(fileName_selected);
          run_dataSource('init_from_browse',fileName_selected);

          end %if confirmChange

%    case 'play'  %commented out 11.04.2005
%     set( sld_mov, 'Enable', 'off' );
%     set( push_play, 'Enable', 'off' );
```

```
%     title( 'playing movie' );
%     movie(ah_mov,mov,1,4);
%     %fh_position=get(fh, 'position');  %for debug. commented out
31.12.2004
%     title( 'done playing movie');
%     set( sld_mov, 'Enable', 'on' );
%     set( push_play, 'Enable', 'on' );
%     run_viewFigs( 'frame' );
%     %ret = 1;

   case 'frame'
    %set( sld_mov, 'Enable', 'on' );
    %set( push_play, 'Enable', 'on' );
    frame_no = round(get(sld_mov,'Value'));
    %frame_no_before = frame_no
    set(sld_mov,'Value',frame_no);
    %dataParams.data_sequence(frame_no)
    title( ['Data Loading -- Frame '
num2str(dataParams.data_sequence(frame_no))] );
    %movie(ah_mov,mov,[20 frame_no]);
    %as recommended by mathworks consultant: (fh-figure handle; ah_mov-
axis
    %handle)
    %figure(fh);
    set(fh, 'colormap',mov(frame_no).colormap);
    set(f, 'colormap',mov(frame_no).colormap);
    %fh_set=frame_no

    %parent_handle=get(get_children_1,'Parent')
    %parent_handle_type=get(parent_handle, 'Type')
    %get_ah_mov_1=get(ah_mov)
    %frame_no=3
    %get_children_1=findall(fh,'Tag','sld_mov')
    %image(mov(frame_no).cdata, 'Parent', ah_mov);
    %%m is for movie, and f is for frame.
    %%want to set the frame axis so that it is the size of the movie
    %%and it doesn't appear so distorted
    %%330x314 good figure size?
    %%what about size of axis?
    image(mov(frame_no).cdata, 'Parent', frame_axis);
    %f_position=get(f, 'position')

    axis([frame_axis ah_mov], 'off');
    %get_children_2=findall(fh,'Tag','sld_mov')
    %save savefile mov
    %image(rand(13),'Parent',frame_axis);
    %image_set=frame_no
    %end recommendation
    %get_ah_mov_2=get(ah_mov)

    if browse_flag  %only have this info if we are browsing
       % disp('I am printing the instance number in the figure window')
    title(frame_axis, ['Section Number '
num2str(figInstanceNumber(frame_no))]);
    else
        title(frame_axis, ['Data Frame '
num2str(dataParams.data_sequence(frame_no))] );
```

```
    end   %if browse_flag %added 16.05.2005
    %frame_no_after = round(get(sld_mov,'Value'))
    %set( sld_mov, 'Enable', 'on' );
    %set( push_play, 'Enable', 'on' );
    %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  User interface method file

% This function maintains a temporary array, tempMap, separate from the
% results array theMap.  tempMap is a masked version of theMap.  theMap
% is never modified here.

  include_globals;
  persistent list_ROIlist chk_selectAll push_ROIs push_saveImg
push_prompt ...
      edit_bgMax edit_tMax edit_tMin push_figTitle txt_eqn chk_colorbar
...
      chk_labels chk_ROIdetails chk_paramsTitle push_analyzeParams ...
      push_analyzeCurve push_filterPixels push_saveStats push_ROILabel
...
      pop_ROInew pop_ROIold ROIold ROInew push_saveImgPpt;
  persistent txt_figROI title_ah obj_list bad_ROInew_flag ;
  global selectedMapi tempMapi tempMap thisROIlist textLabels ...
      parameterType fhm ROILabel_list;


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

  ret = '';
```

```
  arg1 = '';
  if( nargin>1 )
    arg1 = varargin{1};
  end;

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

  definitions;

  switch( action )

   case 'init'
       sizeTempMap_t2=size(tempMap);   %tempMap is wrong dimensions
here 23.05.2005
       %disp('in run_viewResults, case "init"');  %for debug 15.12.2004
    run_viewResults('UIControls');  %added 12.01.2005
       if run_viewResults('isenabled')
        inside_run_viewResults_init=1;
      figure( fhm );  %should bring fhm to the front.  does _not_
create a new version of fhm.
      axis off;  %added 20.05.2005  don't want axis in background of
figure
      ret = fhm;
      sizeTempMap_e=size(tempMap)
      run_viewResults( 'updateFig' );
      return;
    end;
    %fhm=ui_main;  %creates another copy of ui_main which is unneeded
    viewResultsEnabled=enable_viewResults(fhm);
    %if ~enable_viewResults;    % start the UI
    setupUI( fhm, mfilename );  %this is also why the buttons whose
callbacks are
    %defined in run_main weren't working; because, their callbacks are
    %re-defined, but nonexistent, here.
    % this is why the buttons weren't working; the callbacks weren't
set up
    %from the run_main function
    %set( fhm, 'Name', 'Calculation Results' );
    %placeFig( fhm, 'center', 'left' );
    run_image( 'close' );
    ret = fhm;


    % only ROIs with maps generated already matter to this function
    thisROIlist = ROIlist(find([ROIlist.modified]==0));
    %typeFHM=get(fhm,'type')
    %typeParentFHM=get(get(fhm,'parent'),'type')
    %disp('in run_viewResults, line 58')  %for debug 30.12.2004
    %why was allchild called on invalid handle when ?(new ROI was being
    %mapped)?
    %the fhm causing this error does not have a name.
    %fhm should be reserved for the Main Function Handle


    % set up UI controls
    %24.01.05 so that run_viewResults knows all of these variables
    run_viewResults('UIControls');
```

```
    %obj_list = allchild(fhm);
%     % remove semicolons to see that all handles/objects
%     list_ROIlist = findobj( obj_list, 'Tag', 'list_ROIlist' );
%     chk_selectAll = findobj( obj_list, 'Tag', 'chk_selectAll' );
%     edit_bgMax = findobj( obj_list, 'Tag', 'edit_bgMax' );
%     edit_tMax = findobj( obj_list, 'Tag', 'edit_tMax' );
%     edit_tMin = findobj( obj_list, 'Tag', 'edit_tMin' );
%     chk_colorbar = findobj( obj_list, 'Tag', 'chk_colorbar' );
%     chk_labels = findobj( obj_list, 'Tag', 'chk_labels' );
%     chk_ROIdetails = findobj( obj_list, 'Tag', 'chk_ROIdetails' );
%     chk_paramsTitle = findobj( obj_list, 'Tag', 'chk_paramsTitle' );
%     push_saveImg = findobj( obj_list, 'Tag', 'push_saveImg' );
%     push_ROIs = findobj( obj_list, 'Tag', 'push_ROIs' );
%     push_analyzeCurve = findobj( obj_list, 'Tag', 'push_analyzeCurve'
);
%     push_analyzeParams = findobj( obj_list, 'Tag',
'push_analyzeParams' );
%     push_figTitle = findobj( obj_list, 'Tag', 'push_figTitle' );
%     push_filterPixels = findobj( obj_list, 'Tag', 'push_filterPixels'
);
%     % not actually in the UI 15.12.2004
%     %push_prompt = findobj( obj_list, 'Tag', 'push_prompt' )
%     txt_eqn = findobj( obj_list, 'Tag', 'txt_eqn' );



    % figure out T1 or T2
.....................................
    parameterType = feval(['def_' dataParams.type], 'get_commonLabel'
);
    set(findobj(obj_list, 'Tag', 'txt_mapSettingsBox'), 'String', ...
              [parameterType ' Map'] );

   set( pop_ROInew, 'String', strvcat(ROILabel_list) ); % definitions.m
   %added 24.01.05

    run_viewResults( 'setup_list_ROIlist' );

     run_viewResults('update_pop_ROIold');  %25.01.05 added

     sizeTempMap_t=size(tempMap);  %tempMap is 512x512 here  23.05.2005
    run_viewResults( 'update' );
    run_viewResults( 'updateFig' );

   case 'UIControls'  %added 12.01.2005
    obj_list = allchild(fhm);

    % set up UI controls
    % remove semicolons to see that all handles/objects
    list_ROIlist = findobj( obj_list, 'Tag', 'list_ROIlist' );
    chk_selectAll = findobj( obj_list, 'Tag', 'chk_selectAll' );
    edit_bgMax = findobj( obj_list, 'Tag', 'edit_bgMax' );
    edit_tMax = findobj( obj_list, 'Tag', 'edit_tMax' );
    edit_tMin = findobj( obj_list, 'Tag', 'edit_tMin' );
    chk_colorbar = findobj( obj_list, 'Tag', 'chk_colorbar' );
    chk_labels = findobj( obj_list, 'Tag', 'chk_labels' );
```

```
    chk_ROIdetails = findobj( obj_list, 'Tag', 'chk_ROIdetails' );
    chk_paramsTitle = findobj( obj_list, 'Tag', 'chk_paramsTitle' );
    push_saveImg = findobj( obj_list, 'Tag', 'push_saveImg' );
    push_ROIs = findobj( obj_list, 'Tag', 'push_ROIs' );
    push_analyzeCurve = findobj( obj_list, 'Tag', 'push_analyzeCurve'
);
    push_analyzeParams = findobj( obj_list, 'Tag', 'push_analyzeParams'
);
    push_figTitle = findobj( obj_list, 'Tag', 'push_figTitle' );
    push_filterPixels = findobj( obj_list, 'Tag', 'push_filterPixels'
);
    % not actually in the UI 15.12.2004
    %push_prompt = findobj( obj_list, 'Tag', 'push_prompt' )
    txt_eqn = findobj( obj_list, 'Tag', 'txt_eqn' );
    push_ROILabel = findobj( obj_list, 'Tag', 'push_ROILabel' ); %added
24.01.05
    pop_ROInew = findobj( obj_list, 'Tag', 'pop_ROInew' ); %added
24.01.05
    pop_ROIold = findobj( obj_list, 'Tag', 'pop_ROIold' ); %added
24.01.05
    push_saveImgPpt = findobj( obj_list, 'Tag', 'push_saveImgPpt' );
%added 02.02.2005

   case 'isenabled'
       %disp('in run_viewResults, case "isenabled"');   %for debug
15.12.2004
    if( isempty(fhm) )
        %disp('fhm figure handle is empty');
        ret = 0;        % fhm = ''
    elseif( ~ishandle(fhm) )
        %disp('fhm is not a handle');
        ret = 0;    % fhm = invalid handle
    else
        run_viewResults('UIControls');  %added 13.01.2005 due to an
error starting program.
        %disp('fhm is good; chk_selectAll is as follows:')  %22.05.2005
      ret = (strcmpi( get(chk_selectAll, 'Enable'), 'On' ));
%22.05.2005  ignore case
    end;

   case 'raise'
    if( run_viewResults('isenabled') )
      figure(fhm);
      axis off;  %added 20.05.2005  don't want axis in background of
figure
      ret = 1;
    else
      ret = 0;
    end;

   case 'update'        % does not re-display the figure (??30.12.2004-
-why not re-display the figure?-->because updateFIg does that)
      %disp('in run_viewResults, case "update"');  %for debug
15.12.2004
    if( run_viewResults('isenabled')) %removed ~ 22.05.2005
        %disp('run_viewResults good, updateFig')  %22.05.2005
        NotRun_viewResults=1;
```

```
        run_viewResults('updateFig');   %30.12.2004
        %return;                            %commented out 30.12.2004
    else
        NotRun_viewResults=0;
        disp('run_viewResults not on, no fig update')  %22.05.2005
        return;
    end;

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

    %remove semicolons to see if values are being updated  15.12.2004
    set( chk_colorbar, 'Value', viewParams.figColorbar );
    set( chk_labels, 'Value', viewParams.figROIlabels );
    set( chk_ROIdetails, 'Value', viewParams.ROIdetails );
    set( chk_paramsTitle, 'Value', viewParams.paramsTitle );

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

    % update the tempMap
    %disp('about to command to go to update_displayMap(fhm)')
    %if thisROIlist.modified  %if thisROIlist is empty, don't do it
added 19.05.2005
    %above line illicits an error
    update_displayMap(fhm);  %handle is set to fhm inside
update_displayMap anyways--see that code.
    %end %if this ROIlist
    %30.12.2004 is the wrong handle being passed into
update_displayMap???
    %30.12.2004 update_displayMap says that it takes in the main figure
    %handle...But it never prints out line 118 in update_displayMap...

    run_analyzeParams( 'update' );
    run_filterPixels( 'update' );

      case 'update_pop_ROIold'   %added as case 25.01.05
          %disp('in run_viewResults, case "update_pop_ROIold"')  %for
debug 04.02.2005
          %use code from push_saveStats.
   %24.01.05
   array=viewParams.ROILabels;  %26.01.05  saving ROI Labels as a
global
   %this is already a cell array
   %array=get(list_ROIlist, 'string'); %these might be generated by
counting
   %so they may not save name information from session to session
   %commented out 26.01.05
   [h w]=size(array);  %h=# ROIs, w=1
```

```
   %array=mat2cell(array,ones(h,1),[w/4 w/4 w/4 w/4]);  %26.01.05
commented
   %out
   ROILabels_current=array;  %these have extra spaces appended on the
   %left hand side of each item, so they might be too wide for our pull
   %down (also two spaces to the right of each item)
   %LengthROILabelsCurrent=length(ROILabels_current)
   for i=1:length(ROILabels_current)   %to limit lables to length 5
24.01.05
       myLabel=ROILabels_current{i};
       le=length(myLabel);
       if le>5
       myLabel=myLabel(le-4:le);
       %disp('Label truncated to 5');  %was for debug
       ROILabels_current{i}=myLabel;
       end %if
   end %for
   set(pop_ROIold, 'String', ROILabels_current);


   case 'warning'   % another process updated ROIlist while this window
                        %  is still open
     %disp('in run_viewResults, case "warning"');  %for debug
15.12.2004
    if( ~run_viewResults('isenabled')) 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' );
    sizeTempMap_p=size(tempMap);
    run_viewResults( 'update' );
    run_viewResults( 'updateFig' );

    %view results is not an empty window, so it should not need to be
    %closed  (but no harm in closing popups)

%     %case 'close' should only be relevant in run_main?  15.12.2004
%       case 'close'
%     run_analyzeCurve( 'close' );
%     run_analyzeParams( 'close' );
%     run_filterPixels( 'close' );
%     %if( run_viewResults('isenabled'))
%     %  delete( fhm );
%      % clear( mfilename );
%      % disp( 'run_viewResults closed' );  % for debugging
%     %end;

   case 'updateFig'
       %disp('in run_viewResults, case "updateFig"');  %for debug
15.12.2004
    isNewFig = ~run_image('raise');
```

```
    if( ~isNewFig )
      savePosition = get( gcf, 'Position' );
    end;

    %is tempMap empty??? 30.12.2004
    sizeTempMap_b=size(tempMap);  %%%%this is causing the problem.
    %tempMap is 512by 512 but the MapImag should be 256x256
22.05.2005
    maxTempMap=max(max(tempMap));
    %h=sizeTempMap(1);
    %w=sizeTempMap(2);
    %tempMap(h/2:h/2+10,w/2:w/2+10)   %for debug purposes
        %disp('in run_viewResults')

    %disp('tempMap(before run_image) has the following dimensions')

    % display the image ..........................................
    run_image('display', struct('imgmax',viewParams.maxImgVal, ...
                'mapmax',viewParams.maxTval, ...
                'mapmin',viewParams.minTval), ...
          img_background, tempMap );
    if( ~isNewFig ) set(gcf,'Position',savePosition); end;

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

    % ROI labels on the figure ...................................
    selectedROIs = get( list_ROIlist, 'Value' );
    textLabels = zeros(length(thisROIlist),1);
    %ROIlist
    if ROIlist(min(selectedROIs)).maski(1)  %if ROIlist is empty, don't
do it.  19.05.2005 this was 1 after loading new data
        %ROIlist is a structure, so you can't do this to it.
    for n = selectedROIs
        %ROIlistN=ROIlist(n);  %for debug   19.05.2005
        %maskiONE=maski(1);   %for debug   19.05.2005
      i = ROIlist(n).maski(1);  %a source of error. Improper index
matrix reference. 19.05.2005
      [y,x] = getyx(i);
      if iscellstr(viewParams.ROILabels)
          thisLabel=char(viewParams.ROILabels(n));
          lengthThisLabel=length(thisLabel);
          label=thisLabel(lengthThisLabel-2:lengthThisLabel);
      else  %cell contains integers
          label=cell2mat(viewParams.ROILabels(n));
      end
      textLabels(n) = text('String', ['ROI ' label],'Position', ...
%changed from num2str(n) 25.05.2005
              [x,y], 'Color', 'white', 'FontWeight', 'bold');
    end;  %for n
    end; %if ROIlist
    if( get(chk_labels, 'Value')==1 )
    %disp('updating ROI Labels')
        set( textLabels,'Visible','on' );
    else
      set( textLabels,'Visible','off' );
    end;
```

```
    % figure titles ..............................................
    title_ah = updateTitles( title_ah );    % sets up the handles
    run_viewResults( 'chk_paramsTitle', 'noclobber' );  % updates the
text
    run_viewResults('chk_ROIdetails', 'noclobber' );    % updates the
text

   case 'setup_list_ROIlist'        % method .....................
     % Set up the listbox displaying the ROIs.  The information in
     % this list will be replaced and updated as pixels are masked
     % out and the ROI is selected.
     %disp('in run_viewResults, case "setup_list_ROIlist"');  %for
debug 15.12.2004
     %change list_ROIlist, 'string',
     %to be an array rather than a string 05.01.2005
     %tempStr = {'ROI label' 'pixels' 'mean' 'std. dev.'};
     if ~(length(thisROIlist)==0)
     tempStr = {};
     theMapLin = theMapSave(:);
     %lengthThisROIlist=length(thisROIlist) %debug 26.01.05
     %disp('thisROIlist is ');  %debug 26.01.05
     %thisROIlist
     for i = 1:length(thisROIlist)
       %b = length(thisROIlist(i).maski);
       map = theMapLin(thisROIlist(i).maski);
       [a b map] = find(map);   % non-zeros
       %i is ROILabel, length(map) is number of pixels
       if (~isfield(viewParams,'ROILabels')) |
(length(viewParams.ROILabels)<i) %26.01.05  (pipe is logical OR)
       tempStr = vertcat(tempStr, {i length(map) int32(mean(map))
int32(std(map))});
       %%%17.05.2005 cast to int32 to suppress display of decimal
places
       viewParams.ROILabels{i}=i;   %26.01.05  also initiate global
variable value
       %this may add the global variable to the structure.
       else
           tempStr = vertcat(tempStr, {viewParams.ROILabels{i}
length(map) int32(mean(map)) int32(std(map))});
       end %if
     end%for

     %%%%%%how is this different from what happens in update display
map???
     %%%%%% 03.05.2005
     %added 06.01.05 to make sure that array displays properly in
listbox
% Ensure that all elements of cell array are strings for display
for i=1:numel(tempStr) % Goes through all elements
if ~iscellstr(tempStr(i))
tempStr{i}=num2str(tempStr{i});
end  %if
charLength(i)=length(char(tempStr{i}));
end  %for
% Insert spaces to separate the strings for display purposes
maxCharLength=max(charLength);
```

```
maxCharLengthMax=10;   %04.02.2005 trying out different maxCharLength
values
if maxCharLength>maxCharLengthMax   %26.01.05 labels were getting to be
30 long
    maxCharLength=maxCharLengthMax;
end
%disp('tempStr before length operation: ')
%tempStr
maxCharLength_withBuffer=maxCharLength+2;   %%%%%%%%%added 05.05.2005
for i=1:numel(tempStr)
    if length(char(tempStr(i))) > maxCharLength_withBuffer   %added to
deal with labels to long 27.01.05
        myEntry=char(tempStr(i));
        len=length(myEntry);
        myEntry=myEntry(len-maxCharLength_withBuffer+1:len);  %use last
N characters, because space padding is at the beginning.
        tempStr{i}=myEntry;
    end %if
    while length(char(tempStr(i))) < maxCharLength_withBuffer  %trying
to add buffer in between numbers
tempStr(i)=strcat({' '},tempStr(i));
    end  %while
end   %for
%disp('tempStr after length operation: ')
%tempStr
%tempStr=strcat(tempStr,{'  '})  %27.01.05 line causing trouble, so I
commented it out
% Concatenate the cell array of strings
tempStr=cell2mat(tempStr);

     else %if ~(length(thisROIlist)==0)
         tempStr='-';
     end %if ~(length(thisROIlist)==0)

     set( list_ROIlist, 'String', tempStr );
     %disp('tempStr written to list_ROIlist, "String"');
     %run_viewResults('getROIString');  %for debug (will need to be
commented out)  04.02.2005
     %above commented out 17.05.2005
     set( list_ROIlist, 'Value', 1:length(thisROIlist) ); % select all
ROIs
     set( chk_selectAll, 'Value', 1 );
%disp('exiting run_viewResults, case "setup_list_ROIlist"');  %for
debug 04.02.2005
   case 'getSelectedROIs'       % method .....................
      %disp('in run_viewResults, case "getSelectedROIs"');  %for debug
15.12.2004
    ret = get( list_ROIlist, 'Value' );

   case 'getROIString'          % method .....................
      %disp('in run_viewResults, case "getROIString"');  %for debug
15.12.2004
    temp= get( list_ROIlist, 'String' );
    ret = temp(get(list_ROIlist, 'Value'),:) ;  %returns one row of the
list
```

```
   case 'list_ROIlist'   %13.01.2005, this could be modified to similar
to case
       %list_dataParams in run_dataSource, so that user can click on
the
       %ROI to change its Label.  Suggested labels could be given in
the
       %static text.  This method could minimize the size of additional
       %popup windows.  the code that displays the window in def_ir?
       %disp('in run_viewResults, case "list_ROIlist"');  %for debug
       %15.12.2004
    selected = get(list_ROIlist,'Value');  %show how selected values
are output  13.01.2005
    if( length(selected)==length(ROIlist) )  %26.01.05 ROIlist is
global variable where actual data are kept?
      set( chk_selectAll, 'Value', 1 );
    else
      set( chk_selectAll, 'Value', 0 );
    end;
    sizeTempMap_l=size(tempMap);
    run_viewResults( 'update' );
    run_viewResults( 'updateFig' );

   case 'chk_selectAll'
       disp('in run_viewResults, case "chk_selectAll"');  %for debug
15.12.2004
       get(chk_selectAll,'Enable')  %%22.05.2005
    if( get(chk_selectAll,'Value')==1 )
      numROIs = length(thisROIlist);
      set( list_ROIlist, 'Value', 1:numROIs );  % select all ROIs
      sizeTempMap_w=size(tempMap)
      run_viewResults( 'update' );
      sizeTempMap_q=size(tempMap)
      run_viewResults( 'updateFig' );
    end;

   case 'push_ROIs'     % call ROIlist dialog box
       %disp('in run_viewResults, case "push_ROIs"');  %for debug
15.12.2004
    h = run_image('raise');
    if( h ) savePosition = get( h, 'Position' ); end;
    run_ROIlist( 'init' );
    if( h ) set( h, 'Position', savePosition ); end;

   case 'edit_bgMax'
       %disp('in run_viewResults, case "edit_bgMax"');  %for debug
15.12.2004
    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) );
    sizeTempMap_r=size(tempMap)
    run_viewResults( 'updateFig' );

   case {'edit_tMax', 'edit_tMin'}
```

```
      %disp('in run_viewResults, case "edit_tMax/edit_tMin"');  %for
debug 15.12.2004
    if( nargin>1 )
      a = varargin{1};
      b = varargin{1};
    else
      a = get( edit_tMax, 'String' );
      b = get( edit_tMin, 'String' );
    end;

    if( strcmp(action, 'edit_tMax') )
      % update map maximum value
      if( strcmp(a,'auto') )
    map = theMapSave(:);
    viewParams.maxTval = getMapMax( map(selectedMapi) );
      elseif( str2num(a)>0 )
    viewParams.maxTval = str2num(a);
      end;
      set( edit_tMax, 'String', num2str(viewParams.maxTval) );
    else
      % update map minimum value
      b = str2num(b);
      if( ~isempty(b) )
    if( b>=0 & b <=viewParams.maxTval )
      viewParams.minTval = b;
    end;
      end;
      set( edit_tMin, 'String', num2str(viewParams.minTval) );
    end;

    run_viewResults( 'update' );
    run_viewResults( 'updateFig', 'theMap' );
    run_analyzeParams( 'update' );

   case 'push_figTitle'
       %disp('in run_viewResults, case "push_figTitle"');  %for debug
15.12.2004
    a = inputdlg( {'Figure title:', 'x label:', 'y label:'}, 'Input',
...
          [2;2;1], {viewParams.figTitle, viewParams.figxlabel, ...
            viewParams.figylabel} );
    if( ~isempty(a) )
      if( ~strcmp(viewParams.figTitle,a{1}) )
    viewParams.paramsTitle = 0;
    set( chk_paramsTitle, 'Value', 0 );
      end;
      if( ~strcmp(viewParams.figxlabel,a{2}) )
    viewParams.ROIdetails = 0;
    set( chk_ROIdetails, 'Value', 0 );
      end;
      viewParams.figTitle = a{1};
      viewParams.figxlabel = a{2};
      viewParams.figylabel = a{3};
      if( run_image( 'isopen' ))
    updateTitles( title_ah );
      end;
    end;
```

```
   case 'chk_colorbar'
       %disp('in run_viewResults, case "chk_colorbar"');  %for debug
15.12.2004
    viewParams.figColorbar = get( chk_colorbar, 'Value' );
    sizeTempMap_u=size(tempMap)
    run_viewResults( 'updateFig' );

   case 'chk_ROIdetails'
       %disp('in run_viewResults, case "chk_ROIdetails"');  %for debug
15.12.2004
    viewParams.ROIdetails = get( chk_ROIdetails, 'Value' );
    if( ~strcmp(arg1,'noclobber') )
      viewParams.figxlabel = '';
    end;
    if( viewParams.ROIdetails )
        %disp('about to enter another if loop..is this why stuff prints
out?');  %debug 04.02.2005
      if( ~isempty(get(list_ROIlist,'Value')) )
    str = get(list_ROIlist,'String');
    viewParams.figxlabel = strvcat('ROI label  pixels  mean  std.
dev.',str(get(list_ROIlist,'Value'),:));

      end;
    end;
    if( run_image( 'isopen' ))
      updateTitles( title_ah );
    end;
    %disp('in run_viewResults, at end of case "chk_ROIdetails"');  %for
debug 15.12.2004
   case 'chk_paramsTitle'
       %disp('in run_viewResults, case "chk_paramsTitle"');  %for debug
15.12.2004
    viewParams.paramsTitle = get( chk_paramsTitle, 'Value' );
    if( ~strcmp(arg1,'noclobber') )
      viewParams.figTitle = '';
    end;
    if( viewParams.paramsTitle )
      viewParams.figTitle = struct2string(dataParams, 'no_');  %this is
where the actual params are stored 22.02.2005
    end;
    if( run_image( 'isopen' ))
      updateTitles( title_ah );
    end;

   case 'chk_labels'
       %disp('in run_viewResults, case "chk_labels"');  %for debug
15.12.2004
    viewParams.figROIlabels = get( chk_labels, 'Value' );
    try
      if( get(chk_labels, 'Value') )
    set( textLabels,'Visible','on' );
      else
    set( textLabels,'Visible','off' );
      end;
    catch
        sizeTempMap_k=size(tempMap);
```

```
      run_viewResults( 'updateFig' );
    end;

   case 'pop_outRange'
       %disp('in run_viewResults, case "pop_outRange"');  %for debug
15.12.2004
    write_outRange( pop_outRange );

   case 'push_saveImg'
       %disp('in run_viewResults, case "push_saveImg"');  %for debug
15.12.2004
    h = run_image( 'raise' );
    exportFig( h, theMapSave );

   case 'push_analyzeParams'
       %disp('in run_viewResults, case "push_analyzeParams"');  %for
debug 15.12.2004
    run_analyzeParams( 'init' );

   case 'push_analyzeCurve'
       %disp('in run_viewResults, case "push_analyzeCurve"');  %for
debug 15.12.2004
    run_analyzeCurve( 'init' );

   case 'push_filterPixels'
       %disp('in run_viewResults, case "push_filterPixels"');  %for
debug 15.12.2004
    run_filterPixels( 'init', parameterType );

   case 'push_saveStats'   %added 04.01.2005
       display('WARNING: If desired destination file is open, it needs
to be closed to ensure data is saved properly');
     warning('If desired destination file is open, it needs to be
closed to ensure data is saved properly');
     %errordlg('If desired destination file is open, it needs to be
closed to ensure data is saved properly', 'Please close Excel file');
     %disp('push_saveStats activated');
     rangeStart='A2';   %rangeStart is Excel address of the upper left
hand corner
     %where the array will be put.  letters represent columns, and
numbers
     %indicate rows.  05.01.2005
     %get(list_ROIlist, 'value')  %value seems to be an array of
numbers corresponding
     %to the ROI numbers.  ie, 4 ROIs ==> [1 2 3 4]
     %want array to be the basic stats on the ROIs 05.01.2005
     array=get(list_ROIlist, 'string');  %string is a string 05.01.2005
     %array needs to be a matrix where some entries are numbers and
some
     %entries are strings
     %strings saved into .xls files are entered one character per cell
     [h w]=size(array);
     array=mat2cell(array,ones(h,1),[w/4 w/4 w/4 w/4]);

     new_arr=cell(2,h*3);   %%%aded 04.05.2005

     for i=1:h   %to limit lables to length 5  24.01.05
```

```
       myLabel=array{i,1};
       le=length(myLabel);
       if le>7
       myLabel=myLabel(le-4:le);   %%%%%%%%needs to be last five so
excel can find the labels  03.05.2005
       %disp('Label truncated to 7');  %was for debug
       array{i,1}=myLabel;
       end %if

 if ~isempty(str2num(cell2mat(array(i,1))))
    temp_array{i,1}=strcat('x',num2str(str2num(cell2mat(array(i,1)))));
    else temp_array(i,1)=(array(i,1));
 end %if
     end %for i
       [sort_vec,I_sort]=sort(temp_array);

       for i=1:h
    ji=I_sort(i);
    sort_array(i,:)=array(ji,:);
        for j=1:3  %columns    %for loop added 04.05.2005
    col=(i-1)*3+j;
     if j==1
    new_arr{1,col}=strcat(num2str(sort_array{i,1}),' -- pixels');
    elseif j==2
        new_arr{1,col}=strcat(num2str(sort_array{i,1}),' -- mean');
    else
        new_arr{1,col}=strcat(num2str(sort_array{i,1}),' -- std.
dev.');
    end  %if
    new_arr{2,col}=sort_array{i,j+1};
        end  %for j

   end %for i



   labelCellArray={'ROI Label--metric'; 'value'};
     %%%%%%%%%%commented out above and below lines 04.05.2005
     new_arr=cat(2,labelCellArray,new_arr); %put the labels on top of
the values 06.01.2005
     if isfield(dataParams,'section_number')   %added 16.05.2005
     saveFileIndicator={'' ['Section Number: '
num2str(dataParams.section_number)];'ROIs in:' saveFile};
     else
         saveFileIndicator={'' '';'ROIs in:' saveFile};
     end %if isfield
     new_arr=cat(2,new_arr,saveFileIndicator);          %last time
new_arr is modified prior to saving
     visualBuffer={'' '' '' ''};              %%%%%%%how to make visual
buffer of proper width????
     array=cat(1,visualBuffer,array);
     [arrayHeight arrayWidth]=size(new_arr);
     %the .mat extension of the saveFile string needs to be replaced
with
     %.xls
     %can use xlsfinfo (or xlsread) to read information on an xls file
that has already
```

```
     %been created.  (such as what sheets already exist)

     %27.01.05 use uiputfile to have user specify name of file to save
to
     [fileName, pathName, affirmed] = uiputfile('*.xls', 'MRI Mapper
data .xls file name');
     fileName=fullfile(pathName,fileName);

     %changed from strcat to fullfile  06.02.2005
     %does a slash need to be added?  NO 27.01.05
%      %fileName=saveFile;
%      fileName=char(saveFile);  %tried saveFile without quotes and it
saved to that .mat file, overwriting the other data.
%      %saveFile is a global variable, which holds the full
%      %path name (including extension) of the .mat file... 05.01.2005
%      %ideally this has the same name as the .mat file that holds the
ROIs
%      [h w]=size(fileName);
%      fileName=fileName(h,1:w-3);        %remove .mat extension
06.01.2005
%      fileName=strcat(fileName,'xls');     %and add .xls extension
06.01.2005
    dlgName='MRI Mapper data worksheet name and initial cell';
    prompt={'Worksheet name:','Upper-Lefthand cell:'};
    sheetAndCell=inputdlg(prompt,dlgName);   %added 27.01.05
    sheetName=sheetAndCell{1};
    %if ~ischar(sheetName)  %not necessary because inputdlg makes the
outputs strings
     %   sheetName=char(sheetName);
    %end
    rangeStart0=sheetAndCell{2};    %don't let user enter range?  too
much risk for overwrite?
    rangeStart=rangeStart0;
    len=length(rangeStart0);
    numBeg=str2num(rangeStart0(2:len));
    numEnd=numBeg+arrayHeight;
    numEnd=num2str(numEnd);
    rangeEnd=strcat('A',numEnd);
    wholeRange=strcat(rangeStart0,':',rangeEnd);
    %if ~ischar(rangeStart)   %not necessary because inputdlg makes the
outputs strings
     %   rangeStart=char(rangeStart);
   %end

   if exist(fileName)   %if the file already exists, then check to make
sure data is not overwritten
       %sheetName is user-entered.  Also check that the sheet exists
within
       %the file     03.05.2005    %%%use xlsfinfo
       [type, sheets] = xlsfinfo(fileName);
       if ismember(sheetName,sheets)   %if it's a new sheet, overwrite
isn't a problem  03.05.2005
   [n t r]=xlsread(fileName,sheetName,wholeRange);
   while (~isempty(n))|(~isempty(t))  %checks if there is already data
in cell: rangeStart
        rS=str2mat(rangeStart);  %this will be a letter followed by a
number
```

```
        le=length(rS);
        rSnum=str2num(rS(2:le))+1;
        rSnum=num2str(rSnum);
        l=length(rSnum);
        rS(2:1+l)=rSnum;
        rangeStart=char(rS);   %this last line is redundant
        len=length(rangeStart);
    numBeg=str2num(rangeStart(2:len));
    numEnd=numBeg+arrayHeight;
    numEnd=num2str(numEnd);
    rangeEnd=strcat('A',numEnd);
    wholeRange=strcat(rangeStart,':',rangeEnd);
    [n t r]=xlsread(fileName,sheetName,wholeRange);
   end   %while
   if ~(strcmp(rangeStart,rangeStart0))  %28.01.05  display a warning
message that data was saved at a different location
       disp(['Warning: cell ' rangeStart0 ' already contained data.
Data was saved instead at cell ' rangeStart '.']);
   end %if
   end %if
   end %if

   %sheetName='MRIMapper data'; %this is an option that we could use to
label different sheets within the excel file.
     warning off MATLAB:xlswrite:AddSheet  %to supress the addsheet
warning message 06.01.2005
     [success,message]=xlswrite(fileName,new_arr,sheetName,rangeStart);
     if success==1
         disp(['ROI statistics successfully saved to: ' fileName]);
     end
     %success  %can print out success and message for
debug/notification
     %message;

  case 'pop_ROIold'   %added 24.01.05
      %disp('pop_ROIold activated');
      %disp('pop_ROIold value: ');
      ROIold = get(pop_ROIold, 'Value');
      %disp('ROIold from case pop_ROIold')
      %ROIold

 case 'pop_ROInew'   %added 24.01.05
      %disp('pop_ROInew activated');
      %disp('pop_ROInew value: ');
      ROInew = get(pop_ROInew, 'Value');
      LastLabel=length(ROILabel_list);
      if ~(ROInew==LastLabel)  %if ROInew is the last label (#), this
is not necessary
      ListROIOld=get(pop_ROIold, 'String'); %this is a cell 16.02.2005
      ROILabelNew=get(pop_ROInew, 'String');  %whole string of new ROI
Labels
      [hiN wiN]=size(ROILabelNew);  %how wide is this matrix?
      ROInewCell= {ROILabelNew(ROInew,1:wiN)}; %is ROInew a cell ?
16.02.2005
      [hiO wiO]=size(ListROIOld);
      while length(char(ROInewCell))<wiO  %--problem with length of
string??  16.02.2005
```

```
            ROInewCell=strcat({' '},ROInewCell);
        end  %while
        %ROInewCell   %debug
        %ListROIOld   %debug
        %ClassROInew=class(ROInewCell)   %debug
        %ClassListROIOld=class(ListROIOld)   %debug
        %strcmpOldNew=strcmp(ListROIOld, ROInewCell)   %debug
        if max(strcmp(ListROIOld, ROInewCell))  %added 16.02.2005; is
ismember the right function?
            disp('WARNING: You have already assigned this label to an
ROI. You may only assign each label once per image.');
            warning('You have already assigned this label to an ROI.  You
may only assign each label once per image.');
            Handle_error=errordlg('You have already assigned this label
to an ROI.  You may only assign each label once per image.','ROI Label
Assignment Error');
            bad_ROInew_flag=1;
        else
            bad_ROInew_flag=0;
        end  %if max(strcmp
        else
            bad_ROInew_flag=0;
        end %if ~(ROInew==LastLabel)

 case 'push_ROILabel'  %added 24.01.05
     if bad_ROInew_flag  %if it's an ROI already in use, reject it
         warning('You have already assigned this label to an ROI.  You
may only assign each label once per image.');
            Handle_error=errordlg('You have already assigned this label
to an ROI.  You may only assign each label once per image.','ROI Label
Assignment Error');
     else   %otherwise, proceed 17.02.2005
     %disp('push_ROILabel activated');
     run_viewResults('pop_ROIold');   %%%%%%update ROIold value added
04.05.2005
      ROIString=run_viewResults('getROIString');  %this and below added
25.01.05
      ROIString=num2str(ROIString);
      [h w]=size(ROIString);  %the dimensions of the old ROI String
      %the label will be as long as w/4
      %ROILabelOld=ROIString(ROIold,1); %not the old ROI Label, just
the
      %first character
      lenOld=w/4; %determine length of entry to replace it with same
length entry
      if ROInew==length(ROILabel_list)  %to be able to rename ROI's
ordinally
         % disp('ordinal ROI')
          ROILabelNew=num2str(ROIold) ;    %22.02.2005 this must be a
string
      %because line 702 dictates that it must be either a cell
      %array or a string.
      elseif ROInew==1
          disp('Please pull down menu to choose a valid new ROI
label.');
          errordlg('This is not a valid ROI label, please pull down the
menu to choose a valid ROI Label.')
```

```
          ROILabelNew=num2str(ROIold);  %%% 22.05.2005
      else
      ROILabelNew=get(pop_ROInew, 'String');  %whole string of new ROI
Labels
      [hi wi]=size(ROILabelNew);  %how wide is this matrix?
      %disp('named ROI')
      ROILabelNew=ROILabelNew(ROInew,1:wi) ; %chosen new ROI Label--get
whole width
      end %if ROInew
      lenNew=length(ROILabelNew);
      if lenNew<lenOld
      while lenNew<lenOld
          ROILabelNew=strcat({' '},ROILabelNew);
          lenNew=length(char(ROILabelNew));
      end %while
      end %if lenNew
      %lenNew
      %lenOld
      %classROIString=class(ROIString)  %25.01.05 debug
      ROILabelNew=char(ROILabelNew);
      %classROILabelNew=class(ROILabelNew)  %25.01.05 debug
      ROIString(ROIold,1:lenOld)=ROILabelNew;
      %ROI String is not a matrix of entries, it's a matrix of
characters
      set(list_ROIlist,'String',ROIString);
      %ROIold
      %ROILabelNew
      viewParams.ROILabels{ROIold}=ROILabelNew;  %26.01.05 update
global variable
      run_viewResults('update_pop_ROIold');  %25.01.05 added to update
oldROILabel list
     end %if bad_ROInew_flag

 case 'push_saveImgPpt' %added 02.02.2005
      disp('push save image ppt activated')
      prompt={'Slide Title'};
      dlgname='MRI Mapper image PowerPoint slide title';
      pptFile=inputdlg(prompt,dlgname);
      slideTitle=pptFile{1};
      figureN='-f2';
      [fileName, pathName, affirmed] = uiputfile('*.ppt', 'MRI Mapper
image .ppt file name');
     fileName=fullfile(pathName,fileName);  %changed from strcat to
fullfile 07.02.2005
      saveppt(fileName,slideTitle,figureN);
      disp(['Image saved to: ' fileName]);

%%push_close button removed; viewResults not closed independently of
Main
    %    case 'push_close'      % save view parameter changes first
%
%     set( fhm, 'Pointer', 'watch' );
%     disp( ['Saving viewParams to ' saveFile] );
%     try
%       save( saveFile, 'viewParams', '-append' );
%     end;
%     set( fhm, 'Pointer', 'arrow' );
```

```
%     run_viewResults( 'close' );

      otherwise %try action in run_main  %added 13.01.2005
          disp('action is unrecognized by run_viewResults, I will try
it in run_main')
          run_main(action);

  end %switch


% 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;


% updates titles on current figure
function ret = updateTitles( title_ah );
  global viewParams;
  persistent th1 th2 th3;  %22.02.2005 what are th1, th2, th3?

  new_handles = 0;
  if( isempty(title_ah) )  %22.02.2005 what is title_ah?
    new_handles = 1;
  elseif( ~ishandle(title_ah) )
    new_handles = 1;
  end;

  if( new_handles )  %these are obviously title, xlabel, and ylabel
    orig_ah = gca;
    title_ah = axes;
    set( title_ah, 'Visible', 'off' );
    th1 = text( 0, 1, '', 'HorizontalAlignment', 'left', ...
        'VerticalAlignment', 'top');
    th2 = text( 0, .5, '', 'HorizontalAlignment', 'right' );
    th3 = text( 0, .05, '', 'HorizontalAlignment', 'left', ...
%changed from .05 to .03 to try not to hide data beneath image
        'VerticalAlignment', 'top');
    axes( orig_ah );
  end;

  set(th1, 'String', viewParams.figTitle );  %22.02.2005 what is string
here?
  set(th2, 'String', viewParams.figylabel );
  set(th3, 'String', viewParams.figxlabel );

  ret = title_ah;
```

A.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],
...
     ...
    ...
    '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],
...
    ...
    ...
    'MenuBar','none', ...
    'Name','Analyze Parameters', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[301.5 339 291 177.75], ...
    'Tag','Fig2', ...
    'Visible','on');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Min',1, ...
    'Position',[123.75 142.5 101.25 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',[11.25 146.25 101.25 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],
...
    ...
    ...
    'MenuBar','none', ...
    'Name','Data Source', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[21 132.75 282 345], ...
    'Tag','Fig3', ...
    'Visible','on');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[5.25 36.75 270 276], ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 281.25 67.5 22.5], ...
    'String','Source:', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Min',1, ...
```

```
    'Position',[78.75 281.25 101.25 22.5], ...
    'String',' ', ...
    'Style','popupmenu', ...
    'Tag','pop_source', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 180 258.75 22.5], ...
    'String','Data:', ...
    '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',[180 90 90 22.5], ...
    'String','Reset', ...
    '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',[140.25 320.25 135 18.75], ...
    'String','Get params from file...', ...
    'Tag','push_loadParam');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontName','courier', ...
    'HorizontalAlignment','left', ...
    'Position',[11.25 213.75 258.75 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',[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 67.5], ...
    'String',' ', ...
    'Style','listbox', ...
    'Tag','list_dataParams', ...
    'TooltipString','Click on a parameter to edit it', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Min',1, ...
    'Position',[78.75 180 101.25 22.5], ...
    'String',' ', ...
    'Style','popupmenu', ...
    'Tag','pop_data', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[180 281.25 90 22.5], ...
    'String','Select files...', ...
    'Tag','push_sourceDir');
if nargout > 0, fig = h0; end



function fig = ui_editSource()
% 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], ...
     ...
    ...
    'MenuBar','none', ...
    'Name','Edit Source 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_filterPixels()
% 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], ...
    ...
    ...
    'MenuBar','none', ...
    'Name','Filter Pixels', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[80 68 452 326], ...
    'Tag','Fig3', ...
    'Visible','on');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[6 61.5 326.25 123.75], ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[22.5 135 78.75 22.5], ...
    'String',['Remove     ';'Interpolate'], ...
    'Style','popupmenu', ...
    'Tag','list_tMin', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[157.5 138.75 33.75 18.75], ...
    'Style','edit', ...
    'Tag','edit_tMin');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
```

```
    'ListboxTop',0, ...
    'Position',[157.5 116.25 33.75 18.75], ...
    'Style','edit', ...
    'Tag','edit_tMax', ...
    'TooltipString','Enter a number or enter ''auto''');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[135 112.5 22.5 22.5], ...
    'String','>', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[22.5 112.5 78.75 22.5], ...
    'String',['Remove     ';'Interpolate'], ...
    'Style','popupmenu', ...
    'Tag','list_tMax', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[135 135 22.5 22.5], ...
    'String','<', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[101.25 67.5 33.75 22.5], ...
    'String','error', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[101.25 112.5 33.75 22.5], ...
    'String','T1', ...
    'Style','text', ...
    'Tag','txt_mapParam1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[101.25 135 33.75 22.5], ...
    'String','T1', ...
```

```
    'Style','text', ...
    'Tag','txt_mapParam2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[135 67.5 22.5 22.5], ...
    'String','>', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[157.5 71.25 33.75 18.75], ...
    'Style','edit', ...
    'Tag','edit_errorMax');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[22.5 67.5 78.75 22.5], ...
    'String',['Remove     ';'Interpolate'], ...
    'Style','popupmenu', ...
    'Tag','list_errorMax', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 157.5 123.75 22.5], ...
    'Style','text', ...
    'Tag','txt_t1range');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 90 123.75 22.5], ...
    'Style','text', ...
    'Tag','txt_errorRange');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[236.25 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',[11.25 213.75 146.25 22.5], ...
    'String','Relaxation equation:', ...
    'Style','text', ...
    'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 191.25 315 22.5], ...
    'Style','text', ...
    'Tag','txt_equation');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 33.75 101.25 22.5], ...
    'String','Interpolation radius:', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[112.5 37.5 33.75 18.75], ...
    'Style','edit', ...
    'Tag','edit_interRadius');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[146.25 33.75 67.5 22.5], ...
    'String',' pixels', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[202.5 135 123.75 22.5], ...
    'Style','text', ...
    'Tag','txt_TminAffected');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
```

```
    'Position',[202.5 67.5 123.75 22.5], ...
    'Style','text', ...
    'Tag','txt_PminAffected');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[202.5 112.5 123.75 22.5], ...
    'Style','text', ...
    'Tag','txt_TmaxAffected');
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.


%----------------------------Old Main window----------------------------
--
shiftUp=50;
shiftDown=-2;
%location vectors are as [left bottom width height]
h0 = figure('Units','points', ...
    'Color',[0.752941176470588 0.752941176470588 0.752941176470588],
...
     ...
    ...
    'MenuBar','none', ...
    'Name','Main', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576.0000000000001 432.0000000000002], ...
    'PaperUnits','points', ...
    'Position',[17.25 27 417 153.75+329.25+shiftUp], ...
    'Tag','Fig1', ...
    'Visible','on');
h1 = uicontrol('Parent',h0, ...
```

```
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[277.5 330+7.5 131.25 142.5+shiftUp], ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[7.5 330+7.5 131.25 142.5+shiftUp], ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[142.5 330+7.5 131.25 142.5+shiftUp], ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[11.25 330+101.25 123.75 22.5], ...
    'String','Load Source Data...', ...
    '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(fhm);',
...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 330+49 123.75 22.5], ...  % 07.01.2005
    'Style','text', ...
    'Tag','txt_magnet');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
```

```
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 330+11.25+5 123.75 43.75], ...  %07.01.05
    'Style','text', ...
    'Tag','txt_sourceDir');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[146.25 330+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(fhm);',
...
    'ListboxTop',0, ...
    'Position',[281.25+37.5 2.5 90 22.5], ...
    'String','Close MRIMapper', ...
    'Tag','push_close', ...
    'TooltipString','Close MRIMapper');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[146.25 330+31.25 123.75 56.25], ...  %move text up
10.01.2005
    'Style','text', ...
    'Tag','txt_ROImsg');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[281.25 330+11.25 123.75 22.5], ...
    'String','Load Results from File...', ...
    'Tag','push_loadResults', ...
    'TooltipString','Load previously saved calculation results');
```

```
h1 = uicontrol('Parent',h0, ...  %add button to browse through slices
in a series 07.03.2005
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[11.25 330+101.25+26.25 123.75 22.5], ...
    'Enable','On', ...
    'String','Browse Series Images', ...
    'Tag','push_browseImages', ...
    'TooltipString','Browse images in one series to aid in slice
selection');

h1 = uicontrol('Parent',h0, ...  %add button to reorder and name eFilm
DICOM data 17.05.2005
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[11.25 330+101.25+26.25+26.25 123.75 22.5], ...
    'Enable','On', ...
    'String','Re-order eFILM/DICOM data', ...
    'Tag','push_orderImages', ...
    'TooltipString','Order data files in one eFILM directory according
to anatomical location.');


%re: enable set to off:  can't load comparison image until
    %regular image has been loaded. This is helpful if non-dicom image
is
    %chosen, then msize and endian type are already known.  11.01.2005
h1 = uicontrol('Parent',h0, ...  %add button to bring up comparison
image 10.01.2005
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[146.25 330+11.25 123.75 22.5], ...
    'Enable','On', ...
    'String','Open Comparison Image', ...
    'Tag','push_openImage', ...
    'TooltipString','Open image for comparison in segmentation');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'FontWeight','bold', ...
```

```
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 330+123.75+shiftUp+4 123.75 18.5], ...
    '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(fhm);',
...
    'FontWeight','bold', ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[146.25 330+123.75+shiftUp+4 123.75 18.5], ...
    'String','Calculate Map', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'Enable','off', ...
    'ListboxTop',0, ...
    'Position',[281.25 330+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(fhm);',
...
    'FontWeight','bold', ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[281.25 330+123.75+shiftUp+4 123.75 18.5], ...
    'String','Results', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ForegroundColor',[0 0 0.627450980392157], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[281.25 330+33.75+20 123.75 40.75], ... 06.01.05
    'Style','text', ...
    'Tag','txt_resultsSaved');
h1 = uicontrol('Parent',h0, ...
```

```
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[11.25 330+75 123.75 22.5], ...
    'String','Edit Source Data...', ...
    'Tag','push_editSource', ...
    'TooltipString','Select and align data for calculation; set
background');

%-------------------------Old View Results window---------------------
----

% h0 = figure('Units','points', ...
%   'Color',[0.752941176470588 0.752941176470588 0.752941176470588],
...
%   ...
%   ...
%   'MenuBar','none', ...
%   'Name','Results', ...
%   'NumberTitle','off', ...
%   'PaperPosition',[18 180 576 432], ...
%   'PaperUnits','points', ...
%   'Position',[36 159.75 417 329.25], ...
%   'Tag','Fig1', ...
%   'Visible','on');
h1 = uicontrol('Parent',h0, ...  %added 20.05.2005
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'Callback','include_globals; activeFig=''main''; uiresume(fhm);',
...
    'ListboxTop',0, ...
    'Position',[2 29 413 304], ...
    'Style','frame', ...
    'Tag','Frame1');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[116.25 41.25+shiftDown 292.5 93.75], ...
    'Style','frame', ...
    'Tag','Frame2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[206.25 138.75+shiftDown 202.5 90], ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
```

```
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[206.25 232.5+shiftDown 202.5 90], ...
    'Style','frame', ...
    'Tag','Frame1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'FontWeight','bold', ...
    'FontSize',12,...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 302.5+shiftDown 150.25 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',[11.25 142.5+shiftDown 151.25 123.75], ...  %make it
narrower 24.01.05
    'String',' ', ...                                %151.25 was 191.25
    'Style','listbox', ...
    'Tag','list_ROIlist', ...
    'Value',1);
%24.01.05  add popup menus to change ROI Labels from digits to
description
%codes.
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[165 230+shiftDown 40 35], ...
    'String','Change ROI Label from:', ...
    'Style','text', ...
    'Tag','StaticTextROIold');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Min',1, ...
    'Position',[165 210+shiftDown 40 22.5], ...
    'String',' ', ...
    'Style','popupmenu', ...
    'Tag','pop_ROIold', ...
    'Enable','off', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
```

```
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[165 195+shiftDown 40 10], ...
    'String','to:', ...
    'Style','text', ...
    'Tag','StaticTextROInew');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Min',1, ...
    'Position',[165 170+shiftDown 40 22.5], ...
    'String',' ', ...
    'Style','popupmenu', ...
    'TooltipString', 'pFC = posterior Femoral Condyle', ...
    'Tag','pop_ROInew', ...
    'Enable','off', ...
    'Value',1);
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[165 142.5+shiftDown 40 22.5], ...
    'String','Change', ...
    'Enable','off', ...
    'Tag','push_ROILabel');
%end new popup menus 24.01.05
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[11.25 67.5+shiftDown 90 22.5], ...
    'String','Select all', ...
    'Style','checkbox', ...
    'Enable','off', ...
    'Tag','chk_selectAll');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'FontWeight','bold', ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[213.75 202.5+shiftDown 150 22.5], ...
    'String','Display', ...
    'Style','text', ...
    'Tag','txt_display');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[315 45+shiftDown 90 22.5], ...
    'String','Save figure to file...', ...
    'Enable','off', ...
```

```
    'Tag','push_saveImg');

h1 = uicontrol('Parent',h0, ...   %added 02.02.2005 for save to PPT
functionality
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[315-90 45+shiftDown 90 22.5], ...
    'String','Save figure to ppt...', ...
    'Enable','off', ...
    'Tag','push_saveImgPpt');

h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[123.75 90+shiftDown 90 22.5], ...
    'String','Color bar', ...
    'Style','checkbox', ...
    'Enable','off', ...
    'Tag','chk_colorbar');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[213.75 292.5+shiftDown 123.75 22.5], ...
    'String','Equation:', ...
    'Style','text', ...
    'Tag','StaticText3');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[213.75 180+shiftDown 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','right', ...
    'ListboxTop',0, ...
    'Position',[315 180+shiftDown 33.75 22.5], ...
    'String','to ', ...
    'Style','text', ...
    'Tag','StaticText8');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
```

```
    'ListboxTop',0, ...
    'Position',[348.75 101.25+shiftDown 56.25 22.5], ...
    'Style','edit', ...
    'Tag','edit_bgMax', ...
    'Enable','off', ...
    'TooltipString','Enter a number or ''auto''');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[348.75 180+shiftDown 56.25 22.5], ...
    'Style','edit', ...
    'Tag','edit_tMax', ...
    'Enable','off', ...
    'TooltipString', 'Enter a number or enter ''auto''' );
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','right', ...
    'ListboxTop',0, ...
    'Position',[315 101.25+shiftDown 33.75 22.5], ...
    'String','0 to ', ...
    'Style','text', ...
    'Tag','StaticText9');
% viewResults combined in one window with main, close button
unnecessary
% h1 = uicontrol('Parent',h0, ...
%   'Units','points', ...
%   'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
%   'ListboxTop',0, ...
%   'Position',[315 11.25 90 22.5], ...
%   'String','Close', ...
%     'Enable','off', ...
%   'Tag','push_close');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 275+shiftDown 191.25 22.5], ...
    'String','Select ROI(s)  (Ctrl-click to select multiple)', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...   %added 06.01.2005
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[11.25 266+shiftDown 191.25 10.5], ...
    'String','ROI Label   pixels   mean    std. dev.', ...
    'Style','text', ...
    'Tag','StaticText4');
h1 = uicontrol('Parent',h0, ...
```

```
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[123.75 67.5+shiftDown 90 22.5], ...
    'String','ROI labels', ...
    'Style','checkbox', ...
    'Enable','off', ...
    'TooltipString', 'Tip:  In Image window, select Tools / Enable Plot
Editing to move text around', ...
    'Tag','chk_labels');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[11.25 45+shiftDown 90 22.5], ...
    'String','Add ROIs...', ...
    'Enable','off', ...
    'Tag','push_ROIs');
h1 = uicontrol('Parent',h0, ...         %Export stats to Excel added
04.01.2005
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[11.25 110+shiftDown 90 22.5], ...
    'String','Save stats as .xls', ...
    'Enable','off', ...
    'Tag','push_saveStats');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[213.75 258.75+shiftDown 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',[315 236.25+shiftDown 90 22.5], ...
    'String','Plot curve fit...', ...
    'Enable','off', ...
    'Tag','push_analyzeCurve');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[210 236.25+shiftDown 90 22.5], ...
    'String','View details...', ...
    'Enable','off', ...
```

```
    'Tag','push_analyzeParams');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[315 67.5+shiftDown 90 22.5], ...
    'String','Edit title...', ...
    'Enable','off', ...
    'Tag','push_figTitle');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[123.75 45+shiftDown 90 22.5], ...
    'String','ROI details', ...
    'Style','checkbox', ...
    'Enable','off', ...
    'TooltipString', 'Tip:  In Image window, select Tools / Enable Plot
Editing to move text around', ...
    'Tag','chk_ROIdetails');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[281.25 180+shiftDown 56.25 22.5], ...
    'Style','edit', ...
    'Enable','off', ...
    'Tag','edit_tMin');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'FontWeight','bold', ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[123.75 112+shiftDown 150 15], ...
    'String','Map', ...
    'Style','text', ...
    'Tag','txt_mapSettingsBox');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[213.75 101.25+shiftDown 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','left', ...
```

```
    'ListboxTop',0, ...
    'Position',[213.75 78.75+shiftDown 90 22.5], ...
    'String','Parameters', ...
    'Style','checkbox', ...
    'Enable','off', ...
    'TooltipString', 'Tip:  In Image window, select Tools / Enable Plot
Editing to move text around', ...
    'Tag','chk_paramsTitle');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[315 146.25+shiftDown 90 22.5], ...
    'String','Advanced...', ...
    'Enable','off', ...
    'Tag','push_filterPixels');

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],
...
    ...
    ...
    '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',[18.75 247.5 101.25 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');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'ListboxTop',0, ...
    'Position',[123.75 257.5 101.25 12.5], ...
    'String','Highlight ROIs', ...
    'Style','checkbox', ...
    'Tag','chk_highlight');
h1 = uicontrol('Parent',h0, ...   %added 06.01.2005
    'Units','points', ...
    'BackgroundColor',[0.752941176470588 0.752941176470588
0.752941176470588], ...
    'HorizontalAlignment','left', ...
    'ListboxTop',0, ...
    'Position',[117.75 245 146.25 10.5], ...
    'String','ROI Label   pixels', ...
    'Style','text', ...
    'Tag','StaticText4');
if nargout > 0, fig = h0; end



function ret = enable_viewResults()

if nargout > 0, ret = 0; end
```

A.3 Other supporting Files

```
%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.

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

function imgi = alignData( img0, img1, varargin )
% ALIGNDATA

  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
       disp('use entire image for allignment?')  %for debug, added
22.02.2005  Ashley doesn't have the locate a good test block line
printing out...
      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).

  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).


  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.

  [ 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 )

  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;
```

```
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 ret = confirmChange
  % returns 1 if it's okay to make changes
  %made into a seperate .m file 23.05.2005
  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' );  %no need because it is attached to
main
  %13.01.2005
    else ret = 0;
    end; %if
  else
    ret = 1;
  end; %if
ret_of_conCh=ret;  %%for debug 23.05.2005
  if( ret==1 )
      %pop_source  %%this function doesn't know pop_source
      %get(pop_source,'Value')
      %disp('some vars being cleared')
      clear viewParams thisROIlist dataParams tempMap
sourceParams.msize;  %cannot clear all variables  23.05.2005 --
pop_source is needed
      clear img_background theMapSave theMapi ROIlist
      if exist('thisROIlist')
          thisROIlist
          thisROIlist='';
      end %if exist(thisROIlist)
      if exist('tempMap')
```

```
          sizeTEmpMap_f=size(tempMap)  %tempMap is cleared, so this
doesn't
      end %if exist(tempMap)
      run_viewResults('setup_list_ROIlist');  %needs to then be updated
update main?
      ret2=disable_viewResults;  %starting with new data, so there's
nothing to view yet.
      ret=1;  %re-set ret to be one after it is cleared.
      %pop_source
      %get(pop_source,'Value')  %%debug 23.05.2005
    %sourceValid = 0;
    theMapSaveValid = 0;
    ROIlistValid = 0;
    figsValid = 0;
    saveFile = '';
    run_main( 'update' );
  end;




function ret = def_bruker( action, varargin )
%DEF_BRUKER Definitions file for Bruker imaging system.

%   Edits the value of sourceParams

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset sourceParams to defaults for Bruker
    sourceParams = struct( ...
    'sourceDir', '', ...
    'sourceFile', '2DSEQ', ...
    'dataScale', 1/5e6, ...
    'msize', [128 128], ...
    'no_of_slices', 1, ...
    'slice', 1, ...
    'offset', 0, ...
    'noise_level', 1, ...
    'median_filter', 'no', ...
    'byte_ordering', 'big endian', ...
    'type', 'bruker' );

    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
   case 'display_struct'
    ret = rmfield( dataParams, {'type'} );

   case 'input'
    paramPrompts = struct ( ...
    'no_of_slices', 'Total number of slices in scan:', ...
    'slice', 'Slice number to analyze:', ...
    'msize', 'Matrix size [h,w]:', ...
```

```
    'offset', 'Number of pixels in y to offset:', ...
    'median_filter', 'Apply median filter to data? [yes/no]',...
    'byte_ordering', 'big endian (default) or little endian' );

    fieldname = varargin{1};
    switch( fieldname )
     case 'uigetfile'
      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 { 'sourceDir', 'sourceFile' }
      a = inputdlg( {'Enter the source directory:', ...
             'Enter the source file name:'}, ...
            'Source files', [1; 1], ...
            {sourceParams.sourceDir, sourceParams.sourceFile} );
      if( isempty( a )) ret = 0; return;
      else
    sourceParams.sourceDir = a{1};
    sourceParams.sourceFiles = a{2};
    feval(['def_' dataParams.type], 'set_no_of_timeVar', size(a{2},1));
    %dataParams.no_of_timeVar = size(a{2},1);
    ret = 1;
      end; %if

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

    end; %switch( fieldname )

    % fix up the user input if necessary
    switch( fieldname )
     case 'median_filter'
      if( ~strcmp(sourceParams.median_filter,'yes' ))
    sourceParams.median_filter = 'no';
      end;
     case 'msize'
      a = sourceParams.msize;
      if( length(a)>1 )
    sourceParams.msize = [a(1) a(2)];
      else
```

```
    sourceParams.msize = [a(1) a(1)];
      end;
     case {'slice', 'no_of_slices'}
      if( sourceParams.slice > sourceParams.no_of_slices )
    sourceParams.slice = sourceParams.no_of_slices;
      end;
     case 'offset'
      if( sourceParams.offset<0 )
    sourceParams.offset = 0;
      else
    sourceParams.offset = round(sourceParams.offset);
      end;

      case 'byte_ordering'
      ordering = sourceParams.byte_ordering;
      big_str = 'big endian';
      little_str = 'little endian';
      default_str = big_str;
      if( isempty(ordering) )
    sourceParams.byte_ordering = default_str;
      else
    if( ordering(1)=='l' | ordering(1)=='L' )
      sourceParams.byte_ordering = little_str;
    else
      sourceParams.byte_ordering = default_str;
    end;
      end;

    end;

    end; %switch( fieldname )

  end %switch( action )



function ret = def_despot( action, varargin )
%DEF_DESPOT Definitions file for struct dataParams.

%06.02.2005, used code from def_ir

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset dataParams and cm to default values
    dataParams = struct( ...
    'TR', 6.688, ...
    'Flip_Angle', [0.3665 0.0524], ...
    'no_of_FA', 2, ...
    'data_sequence', 1:2, ...
    'initial_guess', [15000 6.688/500], ...
    'type', 'despot', ...
    'timeVar', 'TI' );
```

```
    %for despot fit there is only one TR value, independent of number
of
    %flip angles.  06.02.2005
    %flip angle should be in radians, based on charlie's despot1fit.m
code
    %06.02.2005

    ret = '';

   case 'display_struct'
    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
    ret = rmfield( dataParams, {'type', 'timeVar'} );

    % Rename fields when displaying
    ren_struct = '';

   case 'input'
    paramPrompts = struct ( ...
    'Flip_Angle', 'Flip angles (radians):', ...
    'TR', 'TR time (milliseconds):', ...
    'initial_guess', 'Initial guess for M0 and T1 (ms):', ...
    'data_sequence', ['Which data points to use (can be edited' ...
            ' after loading):'] );

    fieldname = varargin{1};
    switch(fieldname)

     case 'no_of_FA'
      msgbox( strvcat('This field is updated automatically.', ...
                'Edit Flip Angle instead.'), 'Note', 'modal');
      ret = 0;

     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;
    end % switch( fieldnames )

    % fix up the user input if necessary
    switch( fieldname )
     case {'Flip_Angle','TR'}
    % Reset some other parameters (to be safe)
    dataParams.no_of_FA = length(dataParams.Flip_Angle);
    dataParams.data_sequence = 1:dataParams.no_of_FA;
    if( length(dataParams.TR)~=1 &
length(dataParams.TR)~=dataParams.no_of_FA )
      warndlg( 'Number of TR does not match number of TI', 'Warning',
'modal' );
```

```
    end;

     case 'data_sequence'
      seq = dataParams.data_sequence;
      if( ~sum(seq<=0) & ~sum(seq>dataParams.no_of_FA) )
    dataParams.data_sequence = round(seq);
      else
    dataParams.data_sequence = 1:dataParams.no_of_FA;
      end;
     case 'initial_guess'  %this is the x0 from the despot fit code
06.02.2005
      if( length(dataParams.initial_guess)~=2 )
    dataParams.initial_guess = [15000 6.688/500];  %guessing that
TR=6.688 (from 1.5T example)
      end;
    end;

   case 'set_no_of_timeVar'
    val = varargin{1};
    dataParams.no_of_FA = val;

   case 'get_no_of_timeVar'
    ret = dataParams.no_of_FA;

   case 'set_backgroundImg'
    img_background = figs(:,:,dataParams.no_of_FA);

   case 'get_timeaxis'
    ret = dataParams.Flip_Angle;

    %06.02.2005: the displayed equation definitely needs to be changed
   case 'showEqn'
    ret = 'abs( M0 * ( 1 - 2*a*exp(-TI/T1) + exp(-TR/T1) ))';

   case 'get_fitParam'
    ret = 3;        % 3rd column of pfittedSave holds T1 data

   case 'get_errorParam'
    ret = 4;        % 4th column of pfittedSave holds error

   case 'get_paramList'
    ret = strvcat( 'M0', 'a', 'T1', 'Fit error' );

   case 'get_commonLabel'
    ret = 'T1';

  end %switch( action )



function ret = def_diffusion( action, varargin )
%DEF_DIFFUSION Definitions file for struct dataParams.

  include_globals;
  ret = '';
```

```
  switch( action )

   case 'default'   % reset dataParams and cm to default values
    dataParams = struct( ...
    'b', [16.252, 250.304, 750.765, 1000.514, 1500.73], ...
    'no_of_b', 5, ...
    'data_sequence', 1:5, ...
    'initial_guess', [300 .001], ...    % guess for D
    'type', 'diffusion', ...
    'timeVar', 'b' );

    ret = '';

   case 'display_struct'
    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
    ret = rmfield( dataParams, {'type', 'timeVar'} );

    % Rename fields when displaying
    ren_struct = '';

   case 'input'
    paramPrompts = struct ( ...
    'b', 'b factor (s/mm^2):', ...
    'no_of_b', 'Number of b factors', ...
    'initial_guess', 'Initial guess [a D] (D is mm^2/s):', ...
    'data_sequence', ['Which data points to use (can be edited' ...
            ' after loading):'] );

    fieldname = varargin{1};
    switch(fieldname)

     case 'no_of_b'
      msgbox( strvcat('This field is updated automatically.', ...
              'Edit b instead.'), 'Note', 'modal');
      ret = 0;

     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;
    end % switch( fieldnames )

    % fix up the user input if necessary
    switch( fieldname )
     case 'b'
    % reset some other parameters (to be safe)
    dataParams.no_of_b = length(dataParams.b);
    dataParams.data_sequence = 1:dataParams.no_of_b;
     case 'data_sequence'
```

```
      seq = dataParams.data_sequence;
      if( ~sum(seq<=0) & ~sum(seq>dataParams.no_of_b) )
    dataParams.data_sequence = round(seq);
      else
    dataParams.data_sequence = 1:dataParams.no_of_b;
      end;
     case 'initial_guess'
      if( length(dataParams.initial_guess)~=2 )
    dataParams.initial_guess = [300 .001];  % **
      end;
    end;

   case 'set_no_of_timeVar'
    val = varargin{1};
    dataParams.no_of_b = val;

   case 'get_no_of_timeVar'
    ret = dataParams.no_of_b;

   case 'set_backgroundImg'
    img_background = figs(:,:,1);

   case 'get_timeaxis'
    ret = dataParams.b;

   case 'showEqn'
    ret = 'S = a*exp(-b * D)';

   case 'get_fitParam'
    ret = 2;        % 2st column of pfittedSave holds diffusion data

   case 'get_errorParam'
    ret = 3;        % 3rd column of pfittedSave holds error

   case 'get_paramList'
    ret = strvcat( 'a', 'Diffusion', 'Fit error' );

   case 'get_commonLabel'
    ret = 'Diffusion';

  end %switch( action )



function ret = def_ge( action, varargin )
%DEF_GEIR Definitions file GE imaging system.

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset dataParams and cm to default values
    sourceParams = struct( ...
    'sourceDir', '', ...
```

```
    'sourceFiles', '', ...
    'dataScale', 1, ...
    'msize', [512 512], ...
    'noise_level', 1, ...
    'median_filter', 'no', ...
    'byte_ordering', 'big endian', ...
    'type', 'ge' );

    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
   case 'display_struct'
    ret = rmfield( dataParams, {'type'} );

   case 'input'
    paramPrompts = struct ( ...
    'msize', 'Matrix size [h,w]:', ...
    'median_filter', 'Apply median filter to data? [yes/no]', ...
    'byte_ordering', 'big endian (default) or little endian' );

    fieldname = varargin{1};

    switch( fieldname )

     case 'uigetfile'
       ret = def_siemens( 'input', 'uigetfile', '*' );
%       [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 = 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};
    feval(['def_' dataParams.type], 'set_no_of_timeVar',size(a(2),1));
    %dataParams.no_of_timeVar = size(a{2},1);
    ret = 1;
      end; %if

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

    end; %switch( fieldname )

    % fix up the user input if necessary
    switch( fieldname )
     case 'msize'
      a = sourceParams.msize;
      if( length(a)>1 )
    sourceParams.msize = [a(1) a(2)];
      else
    sourceParams.msize = [a(1) a(1)];
      end;
     case 'byte_ordering'
      ordering = sourceParams.byte_ordering;
      big_str = 'big endian';
      little_str = 'little endian';
      default_str = big_str;
      if( isempty(ordering) )
    sourceParams.byte_ordering = default_str;
      else
    if( ordering(1)=='l' | ordering(1)=='L' )
      sourceParams.byte_ordering = little_str;
    else
      sourceParams.byte_ordering = default_str;
    end;
      end;

    end;

  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_ir( action, varargin )
%DEF_IR Definitions file for struct dataParams.

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset dataParams and cm to default values
    dataParams = struct( ...
    'TR', 15, ...
    'TI', [0.025 0.075 0.175 0.3 0.45 0.6 0.75 1 3 7 15], ...
    'no_of_TI', 11, ...
    'data_sequence', 1:11, ...
    'initial_guess', [1 500], ...
    'type', 'ir', ...
    'timeVar', 'TI', ...
    'section_number', 0);

    ret = '';

   case 'display_struct'
    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
    ret = rmfield( dataParams, {'type', 'timeVar'} );

    % Rename fields when displaying
    ren_struct = '';

   case 'input'
    paramPrompts = struct ( ...
    'TI', 'TI times (seconds):', ...
    'TR', 'TR time(s) (seconds):', ...
```

```
    'initial_guess', 'Initial guess for M0 and T1 (ms):', ...
    'data_sequence', ['Which data points to use (can be edited' ...
            ' after loading):'] );

    fieldname = varargin{1};
    switch(fieldname)

     case 'no_of_TI'
      msgbox( strvcat('This field is updated automatically.', ...
                'Edit TI instead.'), 'Note', 'modal');
      ret = 0;

     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;
    end % switch( fieldnames )

    % fix up the user input if necessary
    switch( fieldname )
     case {'TI','TR'}
    % Reset some other parameters (to be safe)
    dataParams.no_of_TI = length(dataParams.TI);
    dataParams.data_sequence = 1:dataParams.no_of_TI;
    if( length(dataParams.TR)~=1 &
length(dataParams.TR)~=dataParams.no_of_TI )
      warndlg( 'Number of TR does not match number of TI', 'Warning',
'modal' );
    end;

     case 'data_sequence'
      seq = dataParams.data_sequence;
      if( ~sum(seq<=0) & ~sum(seq>dataParams.no_of_TI) )
    dataParams.data_sequence = round(seq);
      else
    dataParams.data_sequence = 1:dataParams.no_of_TI;
      end;
     case 'initial_guess'
      if( length(dataParams.initial_guess)~=2 )
    dataParams.initial_guess = [1 500];
      end;
    end;

   case 'set_no_of_timeVar'
    val = varargin{1};
    dataParams.no_of_TI = val;

   case 'get_no_of_timeVar'
    ret = dataParams.no_of_TI;

   case 'set_backgroundImg'
```

```
    img_background = figs(:,:,dataParams.no_of_TI);

   case 'get_timeaxis'
    ret = dataParams.TI;

   case 'showEqn'
    ret = 'abs( M0 * ( 1 - 2*a*exp(-TI/T1) + exp(-TR/T1) ))';

   case 'get_fitParam'
    ret = 3;        % 3rd column of pfittedSave holds T1 data

   case 'get_errorParam'
    ret = 4;        % 4th column of pfittedSave holds error

   case 'get_paramList'
    ret = strvcat( 'M0', 'a', 'T1', 'Fit error' );

   case 'get_commonLabel'
    ret = 'T1';

  end %switch( action )



function ret = def_perfusion( action, varargin )
%DEF_PERFUSION Definitions file for struct dataParams.

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset dataParams and cm to default values
    dataParams = struct( ...
    'T1map_source', '', ...
    'alpha', 0.8, ...
    'lambda', 0.9, ...
    'data_sequence', [2 1 3], ...
    'type', 'perfusion', ...
    'timeVar', 'n', ...     % not really a "time variable"
    'n', 1:3, ...
    'no_of_n', 3 );

    ret = '';

   case 'display_struct'
    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
    ret = rmfield( dataParams, {'type', 'timeVar', 'n', 'no_of_n'} );

    % Rename fields when displaying
    ren_struct = struct( ...
    'data_sequence', 'Input sequence' );

   case 'input'
    paramPrompts = struct ( ...
```

```
    'lambda', 'Value for lambda (mL/g):', ...
    'data_sequence', ...
    'Enter read order of images in this format: [Sc Si T1map]' );

    fieldname = varargin{1};
    switch(fieldname)

     case 'T1map_source'
      [filen, pathn] = ...
      uigetfile('*.mat', 'Select T1 results file' );
      if( filen==0 )
        ret = 0;
        return;  %07.01.2005 changed break to return to allow pcode
compilation
      else
        s = load( [pathn filen], 'sourceParams', 'dataParams' );
        if( ~isfield(s,'sourceParams') | ~isfield(s,'dataParams'))
        errordlg( [filen ' is not a valid results file'], ...
            'Error', 'modal' );
        ret = 0;
        return;  %07.01.2005 changed break to return to allow pcode
compilation
        end;
        if(~(strcmp(s.dataParams.type,'sr') |
strcmp(s.dataParams.type,'ir')))
        errordlg( [filen ' is not a T1 results file'], ...
            'Error', 'modal' );
        ret = 0;
        return;  %07.01.2005 changed break to return to allow pcode
compilation
        end;

     sourceParams.msize = s.sourceParams.msize;
     sourceParams.no_of_slices = s.sourceParams.no_of_slices;
     sourceParams.slice = s.sourceParams.slice;
        if( isfield(s.sourceParams,'offset') )
        sourceParams.offset = s.sourceParams.offset;
        end;
     dataParams.T1map_source = [pathn filen];

     run_dataSource( 'update' );
     ret = 1;
     return;  %07.01.2005 changed break to return to allow pcode
compilation
     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;
    end % switch( fieldnames )
```

```
    % fix up the user input if necessary
    switch( fieldname )
     case 'data_sequence'
      seq = dataParams.data_sequence;
      if( isequal( seq, [1 2 3] ) | isequal( seq, [2 1 3] ))
    dataParams.data_sequence = seq;
      else
    dataParams.data_sequence = [2 1 3];
      end;
    end;

   case 'ok_to_load'
    ret = ~isempty(dataParams.T1map_source);

   case 'set_no_of_timeVar'
    % Do nothing
    %val = varargin{1};
    %dataParams.no_of_n = val;

   case 'get_no_of_timeVar'
    ret = dataParams.no_of_n;

   case 'set_backgroundImg'
    img_background = figs(:,:,1);

   case 'get_timeaxis'
    ret = dataParams.n;

   case 'showEqn'
    ret = 'F = 6000 * l * (Sc - Si)/ ( 2* a* T1 * Sc)';

   case 'get_fitParam'
    ret = 3;        % 3rd column of pfittedSave holds perfusion data

   case 'get_errorParam'
    ret = 4;        % 4th column of pfittedSave holds error

   case 'get_paramList'
    ret = strvcat( 'Sc-Si', 'Perfusion', 'T1', 'Fit error' );

   case 'get_commonLabel'
    ret = 'Perfusion';

  end %switch( action )



function ret = def_siemens( action, varargin )
%DEF_SIEMENS Definitions file for Siemens imaging system.

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset dataParams and cm to default values
```

```
    sourceParams = struct( ...
    'sourceDir', '', ...
    'sourceFiles', '', ...
    'dataScale', 1, ...
    'msize', [512 512], ...
    'noise_level', 1, ...
    'median_filter', 'no', ...
    'byte_ordering', 'big endian', ...
    'type', 'siemens' );

    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
   case 'display_struct'
    ret = rmfield( dataParams, {'type'} );

   case 'input'
    paramPrompts = struct ( ...
    'msize', 'Matrix size [h,w]:', ...
    'median_filter', 'Apply median filter to data? [yes/no]' );

    fieldname = varargin{1};

    switch( fieldname )

        case 'uigetfile'    % user file open dialog box
            conCh=confirmChange;   %%added/changed 23.05.2005
            if conCh
                if( strmatch('*', varargin))
                    if( strcmp(computer, 'PCWIN' )), mask = '*.*';
                    else mask = '*';
                    end;
                else
                    mask = '*.ima';
                end;
                %[filen pathn] = uigetfile( mask, ['Select a data
file'] );
                yesString='Yes' ;
                noString='No'  ;
                default=yesString;
                cr=sprintf('\n');
                qstring=['Will you select all inversion delay files
manually?' ...
                    '  Choosing "No" means that your data are DICOM
format and that'...
                    ' data for each inversion delay are in different
folders.' ...
                    '  In this case, you will select one file and
MRIMapper will automatically load'...
                    ' the corresponding slices for the remaining four
inversion delays.'...
                    cr cr 'Yes: I will select all files for all
inversion delays.'...
                    cr cr 'No: I will select one DICOM file in a folder
that contains data for one inversion delay.'];
                button=questdlg(qstring,'Will you select all files
manually?',yesString,noString,default);
```

```
                if strcmp(button(1),'N')
                    [filen pathn] = uigetfile( mask, ['Select a data
file'] );
                    fullFileName=fullfile(pathn,filen);
                    run_dataSource('init_from_browse',fullFileName)

                else  %else all files must be selected
                    pathn=uigetdir('Select a data directory containing
all desired files');  %%%changed 17.05.2005

                    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 )
                            feval(['def_' dataParams.type],
'set_no_of_timeVar', length(sel));
                            %dataParams.no_of_timeVar = length(sel);
                            sourceParams.sourceDir = pathn;
                            sourceParams.sourceFiles =
fileNames(sel,:);
                            ret = 1;
                        else
                            ret = 0;
                        end; %if(ok)

                        % 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 pathn
                        ret = 0;
                    end; %if  pathn
                end %if strcmp
            end %if confirmChange

    case { 'sourceDir', 'sourceFiles' } % these fields are for
                                        % manually setting the source
%       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(double(isspace(fileNames(i,:))) ))  % non-empty items
only
        sourceParams.sourceFiles = strvcat(sourceParams.sourceFiles,
...
                            fileNames(i,:));
      end;
    end;
    feval( ['def_' dataParams.type], 'set_no_of_timeVar', ...
           size(sourceParams.sourceFiles,1) );
    %dataParams.no_of_timeVar = size(sourceParams.sourceFiles,1);
    ret = 1;
      end; %if

     otherwise
      % use the default dialog box for changing the field value
      %fieldname = action;  %why is this here?
      tempParams = inputFieldDlg( sourceParams, fieldname, '',
paramPrompts );
      if( isstruct(tempParams) )
    sourceParams = tempParams;
    ret = 1;
      else
    ret = 0;
      end;

    end; %switch( fieldname )

    % fix up the user input if necessary
    switch( fieldname )
     case 'msize'
      a = sourceParams.msize;
      if( length(a)>1 )
    sourceParams.msize = [a(1) a(2)];
      else
    sourceParams.msize = [a(1) a(1)];
      end;
     case 'byte_ordering'
      ordering = sourceParams.byte_ordering;
      big_str = 'big endian';
      little_str = 'little endian';
      default_str = big_str;
      if( isempty(ordering) )
    sourceParams.byte_ordering = default_str;
      else
```

```
    if( ordering(1)=='1' | ordering(1)=='L' )
      sourceParams.byte_ordering = little_str;
    else
      sourceParams.byte_ordering = default_str;
    end;
      end;

    end; % switch( fieldname )

  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;



function ret = def_sr( action, varargin )
% DEF_SR Definitions file.

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset dataParams to its default for SR
    dataParams = struct( ...
    'TR', [0.04 0.1 0.25 0.5 0.8 1.5], ...
    'no_of_TR', 6, ...
    'data_sequence', 1:6, ...
    'initial_guess', [30 3], ...
    'type', 'sr', ...
    'timeVar', 'TR' );

   case 'display_struct'
    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
    ret = rmfield( dataParams, {'type', 'timeVar'} );

    % Rename fields when displaying
    ren_struct = '';

   case 'input'
    paramPrompts = struct ( ...
    'TR', 'TR times (seconds):', ...
    'data_sequence', ['Which data points to use (can be edited' ...
```

```
            ' after loading):'], ...
    'initial_guess', 'Initial guesses for M0 and T1 (seconds):' );

    fieldname = varargin{1};
    switch( fieldname )
     case 'no_of_TR'
      msgbox( strvcat('This field is updated automatically.', ...
              'Edit TR instead.'), 'Note', 'modal');
      ret = 0;

     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;

    end % switch( fieldname )

    % fix up the user input if necessary
    switch( fieldname )
     case 'TR'
      % reset some other parameters (to be safe)
      dataParams.no_of_TR = length(dataParams.TR);
      dataParams.data_sequence = 1:dataParams.no_of_TR;
     case 'data_sequence'
      seq = dataParams.data_sequence;
      if( ~sum(seq<=0) & ~sum(seq>dataParams.no_of_TR) )
    dataParams.data_sequence = round(seq);
      else
    dataParams.data_sequence = 1:dataParams.no_of_TR;
      end;
    end; % switch( fieldname )

   case 'set_no_of_timeVar'
    val = varargin{1};
    dataParams.no_of_TR = val;

   case 'get_no_of_timeVar'
    ret = dataParams.no_of_TR;

   case 'set_backgroundImg'
    img_background = figs(:,:,dataParams.no_of_TR);

   case 'get_timeaxis'
    ret = dataParams.TR;

   case 'showEqn'
    ret = 'M0 * (1 - exp(-TR/T1))';

   case 'get_fitParam'
    ret = 2;            % 2nd column of pfittedSave holds T1 data
```

```
   case 'get_errorParam'
    ret = 3;            % 3rd column of pfittedSave holds error

   case 'get_paramList'
    ret = strvcat( 'M0', 'T1', 'Fit error' );

   case 'get_commonLabel'
    ret = 'T1';

  end; %switch( action )



function ret = def_t2( action, varargin )
% DEF_T2 Definitions file.

  include_globals;
  ret = '';

  switch( action )

   case 'default'   % reset dataParams to its default for T2
    dataParams = struct( ...
    'TR', 10, ...
    'TE', 20:20:640, ...
    'no_of_TE', 32, ...
    'data_sequence', 2:32, ...
    'initial_guess', [50 300], ...
    'type', 't2', ...
    'timeVar', 'TE' );

   case 'display_struct'
    % Returns structure with certain fields omitted.  Note that
    % omitted fields should be the last fields in the original
structure.
    ret = rmfield( dataParams, {'type', 'timeVar'} );

    % Rename fields when displaying
    ren_struct = '';

   case 'input'
    paramPrompts = struct ( ...
    'TR', 'TR time (seconds):', ...
    'TE', 'TE times (ms):', ...
    'data_sequence', ['Which data points to use (can be edited' ...
           ' after loading):'], ...
    'initial_guess', 'Initial guesses for M0 and T2 (ms):' );

    fieldname = varargin{1};
    switch( fieldname )

     case 'no_of_TE'
      msgbox( strvcat('This field is updated automatically.', ...
              'Edit TE instead.'), 'Note', 'modal');
      ret = 0;
```

```
      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;

    end % switch( fieldname )

    % fix up the user input if necessary
    switch( fieldname )
     case 'TE'
      % reset some other parameters (to be safe)
      dataParams.no_of_TE = length(dataParams.TE);
      dataParams.data_sequence = 1:dataParams.no_of_TE;
     case 'data_sequence'
      seq = dataParams.data_sequence;
      if( ~sum(seq<=0) & ~sum(seq>dataParams.no_of_TE) )
    dataParams.data_sequence = round(seq);
      else
    dataParams.data_sequence = 1:dataParams.no_of_TE;
      end;
    end; % switch( fieldname )

   case 'set_no_of_timeVar'
    val = varargin{1};
    dataParams.no_of_TE = val;

   case 'get_no_of_timeVar'
    ret = dataParams.no_of_TE;

   case 'set_backgroundImg'
    img_background = figs(:,:,dataParams.data_sequence(1));

   case 'get_timeaxis'
    ret = dataParams.TE;

   case 'showEqn'
    ret = 'abs( M0*exp(-TE/T2 )';

   case 'get_fitParam'
    ret = 2;            % 2nd column of pfittedSave holds T1 data

   case 'get_errorParam'
    ret = 3;            % 3rd column of pfittedSave holds error

   case 'get_paramList'
    ret = strvcat( 'M0', 'T2', 'Fit error' );

   case 'get_commonLabel'
    ret = 'T2';
```

```
  end; %switch( action )


% DEF_VIEW Sets defaults for viewParams

%26.01.05  added ROI list labels
function ret = def_view( action )

  ret = '';

  switch( action )
    case 'default'
     viewParams = struct( ...
     'maxImgVal', 'auto', ...
     'maxTval', 'auto', ...
     'minTval', 0, ...
     'maxErrorVal', 0, ...
     'maxTvalAction', 'Remove', ...
     'minTvalAction', 'Remove', ...
     'maxErrorValAction', 'Remove', ...
     'interpolationRadius', 1.5, ...
     'figTitle', '', ...
     'figxlabel', '', ...
     'figylabel', '', ...
     'figColorbar', 1, ...
     'figROIlabels', 0, ...
     'ROIdetails', 0, ...
     'paramsTitle', 0);
     ret = viewParams;

      case 'new'   %added 26.01.05
     viewParams = struct( ...
     'maxImgVal', 'auto', ...
     'maxTval', 'auto', ...
     'minTval', 0, ...
     'maxErrorVal', 0, ...
     'maxTvalAction', 'Remove', ...
     'minTvalAction', 'Remove', ...
     'maxErrorValAction', 'Remove', ...
     'interpolationRadius', 1.5, ...
     'figTitle', '', ...
     'figxlabel', '', ...
     'figylabel', '', ...
     'figColorbar', 1, ...
     'figROIlabels', 0, ...
     'ROIdetails', 0, ...
     'paramsTitle', 0, ...
     'ROILabels', {[]});  %26.01.05  does this still work as an array?
just {} is not enough.
     ret = viewParams;
  end; %switch( action )
```

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


source_list = { 'bruker' ; 'siemens'; 'ge' };
sourceLabel_list = { 'Bruker' ; 'Siemens' ; 'General Electric' };
data_list = { 'ir' ; 'sr' ; 't2' ; 'diffusion'; 'perfusion' ; 'despot'
};
%look locker added 21.01.2005
%changed from look locker to despot  06.02.2005
dataLabel_list = { 'Inversion recovery' ; 'Saturation recovery' ; ...
          'T2 spin echo' ; 'Diffusion' ; 'Perfusion' ; 'Despot'};
      %look locker added 21.01.2005
      %changed from look locker to despot 06.02.2005
outRange_list = { 'Remove', 'Interpolate' };
outRange_list2 = { 'Clip', 'Remove', 'Interpolate' };
ROILabel_list = {'---';'   FC' ; '  aFC' ; '  cFC' ; '  pFc' ; '   TP'
; '  aTP' ; '  cTP' ; '  pTP' ; '    #'}; %added 24.01.05
%%%%%finalized 04.05.2005
%definitions as indicated by Nomenclature document produced by Felix
%Eckstein
%following make sense for sagittal view:
%T=total cartilage in image
%TF=total femoral cartilage
%TT=total tibial cartilage
%Fc=femoral central cartilage
%Tc=tibial central cartilage
%following make sense for coronal view:
%M = medial cartilage
%L = lateral cartilage



% Files:
%   'source' is from source_list
%   'data' is from data_list
%
%   def_source.m
%   def_data.m
%   load_source.m
%   exe_data.m
%   plot_data.m




function f = diff_fn( p, S, b )
% DIFF_FN  Diffusion equation

% p is parameter to estimate
% equation: S = a*exp(-b*D)
%   where p(1) is a, p(2) is D
err = S - p(1)*exp( -b * p(2) );
f = sum(err.^2);
```

```
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)
      %disp('in for loop')
    if( ~fileList(i).isdir)  %for files which are not directories
        %disp('getting individual file info')
      fileName = strvcat( fileName, fileList(i).name );
      fileInfo = strvcat( fileInfo, [fileList(i).name '  ' ...
            num2str(round(fileList(i).bytes/1024)) 'K'] );
    elseif (~strcmp('.',fileList(i).name)) &
(~strcmp('..',fileList(i).name))
        %disp('getting directory info')
        %for (sub)directories  added 17.05.2005

[subFileName,subFileInfo]=dirFiles(fullfile(pathn,fileList(i).name));
        %length(subFileName)
        fileName=strvcat(fileName,['----' fileList(i).name]);
        fileInfo=strvcat(fileInfo,['----' fileList(i).name]);
        [hn wn]=size(subFileName);
        [hi wi]=size(subFileInfo);
        for j = 1:hn
            fileName = strvcat( fileName,
fullfile(fileList(i).name,subFileName(j,1:wn)));
            fileInfo = strvcat(fileInfo,
strcat(fileList(i).name,filesep,subFileInfo(j,1:wi)));
        end;  %for j
    end;  %if
  end; %for i


function ret = disable_viewResults(handle)
% enables the view results area of the main window

global fhm childs;

handle=fhm;  %only want to enable features within main figure handle
15.12.2004
axis off;  %added 20.05.2005  don't want axis in background of figure
%fhm = ui_main; <-- this creates an unneeded new copy of ui_main

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

    % Set up event callbacks.

    % set up UI controls
    %childs = allchild(fhm);  %childs is global and should already be
    %generated  13.01.2005
    if isempty(childs)
        childs=allchild(fhm);
```

```
    end    %just in case childs is empty

    chk_selectAll = findobj( childs, 'Tag', 'chk_selectAll' );
    push_saveImg = findobj( childs, 'Tag', 'push_saveImg' );
    push_saveStats = findobj( childs, 'Tag', 'push_saveStats' );
    chk_colorbar = findobj( childs, 'Tag', 'chk_colorbar' );
    edit_bgMax = findobj( childs, 'Tag', 'edit_bgMax' );
    edit_tMax = findobj( childs, 'Tag', 'edit_tMax' );
    push_close = findobj( childs, 'Tag', 'push_close' );
    chk_labels = findobj( childs, 'Tag', 'chk_labels' );
    push_ROIs = findobj( childs, 'Tag', 'push_ROIs' );
    push_analyzeCurve = findobj( childs, 'Tag', 'push_analyzeCurve' );
    push_analyzeParams = findobj( childs, 'Tag', 'push_analyzeParams'
);
    push_figTitle = findobj( childs, 'Tag', 'push_figTitle' );
    chk_ROIdetails = findobj( childs, 'Tag', 'chk_ROIdetails' );
    edit_tMin = findobj( childs, 'Tag', 'edit_tMin' );
    chk_paramsTitle = findobj( childs, 'Tag', 'chk_paramsTitle' );
    push_filterPixels = findobj( childs, 'Tag', 'push_filterPixels' );
    push_ROILabel = findobj( childs, 'Tag', 'push_ROILabel' ); %added
24.01.05
    pop_ROInew = findobj( childs, 'Tag', 'pop_ROInew' ); %added
24.01.05
    pop_ROIold = findobj( childs, 'Tag', 'pop_ROIold' ); %added
24.01.05
    push_saveImgPpt = findobj( childs, 'Tag', 'push_saveImgPpt' );
%added 02.02.2005

    set( chk_selectAll, 'Enable', 'Off' );
    set( push_saveImg, 'Enable', 'Off' );
    set( push_saveStats, 'Enable', 'Off' );
    set( chk_colorbar, 'Enable', 'Off' );
    set( edit_bgMax, 'Enable', 'Off' );
    set( edit_tMax, 'Enable', 'Off' );
    %set( push_close, 'Enable', 'Off' );  %push_close should always be
    %enabled
    set( chk_labels, 'Enable', 'Off' );
    set( push_ROIs, 'Enable', 'Off' );
    set( push_analyzeCurve, 'Enable', 'Off' );
    set( push_analyzeParams, 'Enable', 'Off' );
    set( push_figTitle, 'Enable', 'Off' );
    set( chk_ROIdetails, 'Enable', 'Off' );
    set( edit_tMin, 'Enable', 'Off' );
    set( chk_paramsTitle, 'Enable', 'Off' );
    set( push_filterPixels, 'Enable', 'Off' );
    set( push_ROILabel, 'Enable', 'Off' ); %added 24.01.05
    set( pop_ROInew, 'Enable', 'Off' ); %added 24.01.05
    set( pop_ROIold, 'Enable', 'Off' ); %added 24.01.05
    set( push_saveImgPpt, 'Enable', 'Off' ); %added 02.02.2005

ret=1;
if nargout > 1, ret = 0; 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 = enable_viewResults(handle)
% enables the view results area of the main window

global fhm childs;

handle=fhm;  %only want to enable features within main figure handle
15.12.2004
axis off;  %added 20.05.2005  don't want axis in background of figure
%fhm = ui_main; <-- this creates an unneeded new copy of ui_main

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

    % Set up event callbacks.

    % set up UI controls
    %childs = allchild(fhm);  %childs is global and should already be
    %generated  13.01.2005
    if isempty(childs)
        childs=allchild(fhm);
    end    %just in case childs is empty

    chk_selectAll = findobj( childs, 'Tag', 'chk_selectAll' );
    push_saveImg = findobj( childs, 'Tag', 'push_saveImg' );
    push_saveStats = findobj( childs, 'Tag', 'push_saveStats' );
    chk_colorbar = findobj( childs, 'Tag', 'chk_colorbar' );
    edit_bgMax = findobj( childs, 'Tag', 'edit_bgMax' );
    edit_tMax = findobj( childs, 'Tag', 'edit_tMax' );
    push_close = findobj( childs, 'Tag', 'push_close' );
    chk_labels = findobj( childs, 'Tag', 'chk_labels' );
    push_ROIs = findobj( childs, 'Tag', 'push_ROIs' );
    push_analyzeCurve = findobj( childs, 'Tag', 'push_analyzeCurve' );
    push_analyzeParams = findobj( childs, 'Tag', 'push_analyzeParams'
);
    push_figTitle = findobj( childs, 'Tag', 'push_figTitle' );
    chk_ROIdetails = findobj( childs, 'Tag', 'chk_ROIdetails' );
    edit_tMin = findobj( childs, 'Tag', 'edit_tMin' );
    chk_paramsTitle = findobj( childs, 'Tag', 'chk_paramsTitle' );
    push_filterPixels = findobj( childs, 'Tag', 'push_filterPixels' );
    push_ROILabel = findobj( childs, 'Tag', 'push_ROILabel' ); %added
24.01.05
    pop_ROInew = findobj( childs, 'Tag', 'pop_ROInew' ); %added
24.01.05
    pop_ROIold = findobj( childs, 'Tag', 'pop_ROIold' ); %added
24.01.05
    push_saveImgPpt = findobj( childs, 'Tag', 'push_saveImgPpt' );
%added 02.02.2005

    set( chk_selectAll, 'Enable', 'On' );
    set( push_saveImg, 'Enable', 'On' );
    set( push_saveStats, 'Enable', 'On' );
    set( chk_colorbar, 'Enable', 'On' );
    set( edit_bgMax, 'Enable', 'On' );
```

```
    set( edit_tMax, 'Enable', 'On' );
    set( push_close, 'Enable', 'On' );
    set( chk_labels, 'Enable', 'On' );
    set( push_ROIs, 'Enable', 'On' );
    set( push_analyzeCurve, 'Enable', 'On' );
    set( push_analyzeParams, 'Enable', 'On' );
    set( push_figTitle, 'Enable', 'On' );
    set( chk_ROIdetails, 'Enable', 'On' );
    set( edit_tMin, 'Enable', 'On' );
    set( chk_paramsTitle, 'Enable', 'On' );
    set( push_filterPixels, 'Enable', 'On' );
    set( push_ROILabel, 'Enable', 'On' ); %added 24.01.05
    set( pop_ROInew, 'Enable', 'On' ); %added 24.01.05
    set( pop_ROIold, 'Enable', 'On' ); %added 24.01.05
    set( push_saveImgPpt, 'Enable', 'On' ); %added 02.02.2005

ret=1;
if nargout > 1, ret = 0; end



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 [t1map, t1mapi, pfitted] = exe_despot( mask )
%EXE_DESPOT

%code taken from exe_ir.  06.02.2005
%modified to do despot fit.

  global figs sourceParams dataParams

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

  % for lsqcurvefit
  %parameters from Charlie's despot1fit code
  OPTIONS = optimset( 'lsqcurvefit' );
  OPTIONS = optimset( OPTIONS, 'Display', 'off');

  mask = double(mask);
  a = dataParams.data_sequence(length(dataParams.data_sequence));
  roind = find( figs(:,:,a).*mask > sourceParams.noise_level);
  if( isempty(roind) )
    t1map = ''; t1mapi = ''; pfitted = '';
    return;          % abort because nothing to do
  end;

  % create array of data to be processed array
  nn = 1;
  for n = dataParams.data_sequence
```

```
    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 = [dataParams.initial_guess(1) dataParams.initial_guess(2)];

  %TR = dataParams.TR * 1000;
  %I don't think that re-scaling is necessary.
  %the value from the header should be good
 TR = dataParams.TR;  %TR should be in milliseconds

 %readjust iniguess based on TR value:
 iniguess(2)=iniguess(2)*TR/6.688;  %6.688 is value in def_despot
06.02.2005

 %06.02.2005 -- only one TR value necessary per slice
 % if( length(TR)==1 )          % single TR
  %  TR = ones(1,length(dataParams.data_sequence)) * TR;
 % else                  % multiple TRs
  %  TR = TR(dataParams.data_sequence);
 % end;

  Flip_Angle = dataParams.Flip_Angle(dataParams.data_sequence);  %flip
angles should be in radians
Flip_Angle=pi*Flip_Angle/180;  %flip angle is entered in degrees

    %str1=[' of total' int2str(s) ' pixels'];
    h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
    a = length(dataParams.data_sequence);
    %s=3;  %debug 08.02.2005
    for n = 1:s,        % index through pixels
      SI = Data(n,:);       % for a pixel, get from all images
      [pfitted(n,1:2) resnorm(n) residual(n,:) exitflag(n)] = ...
      lsqcurvefit(@Meqfun,iniguess,Flip_Angle,SI,[0 0],[1e9
1e9],OPTIONS);
      %fit line taken from Charlie's despot1fit.m code
      %changed upperbound from 1e9 to 1e4 <==changed scale of
answers???

      %not appropriate for this fit:
      %pfitted(n,3) = fval/mean(SI);    % MSE normalized
      %pfitted(n,4) = output.iterations;

      if( mod(n,10)==0 )
    waitbar( n/s, h );
      end
    end
    delete( h );  %changed from close to delete 31.12.2004
%class_pfitted1=class(pfitted(:,1))
%class_pfitted2=class(pfitted(:,2))
%class_pfitted3=class(pfitted(:,3))
```

```
%class_pfitted4=class(pfitted(:,4))
%class_pfitted5=class(pfitted(:,5))
%pfitted100=pfitted(100,:)
%%this did not come out as an integer.

%disp('resnorm from exe_despot: ')%debug 07.02.2005
 %     resnorm%debug 07.02.2005
      disp('exitflag from exe_despot: ')%debug 07.02.2005
      exitflag%debug 07.02.2005
  %    disp('residual from exe_despot: ')%debug 07.02.2005
  %    residual   %debug 07.02.2005

maxPFitted=max(pfitted(:,1:2))  %for debug 08.02.2005

%make TR into a vector
%TR=TR*ones(length(pfitted(:,2));
%or use dot divide--TR needs to divide all of the things.
    t1_values = TR./pfitted(:,2);  %from Charlie's despot1fit.m code
    MO_values = pfitted(:,1); %from Charlie's despot1fit.m code

    %put t1 values into pfitted(3)
    pfitted(:,3)=t1_values;
    %error values shoudl be in pfitted(4)
    %residuals are the difference between the data and the fit.
    %take the error for each pixel and sum the squares
    pfitted(:,4)=sum((residual.^2),2);  %09.02.2005 sum across the
pairs of errors
    %pfittedSize=size(pfitted) %09.02.2005 debug
    %dataSize=size(Data) %09.02.2005 debug
    pfitted(:,4)=pfitted(:,4)./mean(Data,2);  %normalize by dividing by
average pixel strength

maxT1values=max(t1_values)  %for debug 08.02.2005
    %scale down t1 values to fit 256 matrix
    %t1_values = t1_values./10;

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

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

    t1map = t1_ima;
    t1mapi = roind;

maxT1map=max(t1map)  %for debug 08.02.2005
maxT1mapi=max(t1mapi)  %for debug 08.02.2005


function [diffmap, diffmapi, pfitted] = exe_diffusion( mask )
%EXE_IR

  global figs sourceParams dataParams
```

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

% for fminsearch
OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );

a = dataParams.data_sequence(length(dataParams.data_sequence));
roind = find( figs(:,:,a).*mask > sourceParams.noise_level);
if( isempty(roind) )
  diffmap = ''; diffmapi = ''; pfitted = '';
  return;          % abort because nothing to do
end;

% create array of data to be processed
nn = 1;
for n = dataParams.data_sequence
  temp = figs(:,:,n);
  temp = temp(:);
  Data(:,nn) = temp(roind);
  nn = nn + 1;
end
clear temp

%Initial values and curve fitting
b = dataParams.b;
s = size(Data,1);
pfitted = zeros(s,3);
iniguess = dataParams.initial_guess;

h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
for n = 1:s,       % index through pixels

  S = Data(n,:);       % for a pixel, get from all images
  [pfitted(n,1:2) fval exitflag output] = ...
  fminsearch( 'diff_fn', iniguess, OPTIONS, S, b );
  pfitted(n,3) = fval/mean(S);     % MSE normalized
  pfitted(n,4) = output.iterations;
  iniguess = pfitted(n,1:2);

  if( mod(n,10)==0 )
    waitbar( n/s, h );
  end
end
delete( h );  %changed from close to delete 31.12.2004

diff_values = pfitted(:,2);

%create image of these values
ima = zeros(sourceParams.msize(1)*sourceParams.msize(2),1);
ima(roind) = diff_values;

diffmap = reshape(ima, sourceParams.msize(1), sourceParams.msize(2));
diffmapi = roind;
```

```
function [t1map, t1mapi, pfitted] = exe_ir( mask )
%EXE_IR

  global figs sourceParams dataParams

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

  % for fminsearch
  OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );

  mask = double(mask);
  a = dataParams.data_sequence(length(dataParams.data_sequence));
  roind = find( figs(:,:,a).*mask > sourceParams.noise_level);
  if( isempty(roind) )
    t1map = ''; t1mapi = ''; pfitted = '';
    return;          % abort because nothing to do
  end;

  % create array of data to be processed array
  nn = 1;
  for n = dataParams.data_sequence
    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 = [dataParams.initial_guess(1) 1
dataParams.initial_guess(2)];
  TR = dataParams.TR * 1000;
  if( length(TR)==1 )            % single TR
    TR = ones(1,length(dataParams.data_sequence)) * TR;
  else                    % multiple TRs
    TR = TR(dataParams.data_sequence);
  end;
  TI = dataParams.TI(dataParams.data_sequence) * 1000;

    %str1=[' of total' int2str(s) ' pixels'];
    h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
    a = length(dataParams.data_sequence);
    for n = 1:s,         % index through pixels
      SI = Data(n,:);        % for a pixel, get from all images
      iniguess(1) = max(SI);
      [sim,simi] = min(SI); % sim is minimum value, simi is index of
min.
      iniguess(3) = TI(simi)/0.6;
      [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  %normalized error
      pfitted(n,5) = output.iterations;
```

```
      if( mod(n,10)==0 )
    waitbar( n/s, h );
      end
    end
    delete( h );  %changed from close to delete 31.12.2004
%class_pfitted1=class(pfitted(:,1))
%class_pfitted2=class(pfitted(:,2))
%class_pfitted3=class(pfitted(:,3))
%class_pfitted4=class(pfitted(:,4))
%class_pfitted5=class(pfitted(:,5))
%pfitted100=pfitted(100,:)
%%this did not come out as an integer.

    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(sourceParams.msize(1)*sourceParams.msize(2),1);
    t1_ima(roind) = t1_values;

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

    t1map = t1_ima;
    t1mapi = roind;



function [Fmap, Fmapi, pfitted] = exe_perfusion( mask )
%EXE_PERFUSION

  global figs sourceParams dataParams


  OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );

  mask = double(mask);

  % Perfusion source data images are Sc, Si, then T1 images.


    % mask out the pixels that are < noise threshold
    a = dataParams.data_sequence(1);
    img = figs(:,:,a).*mask;
    [i j v] = find(img < sourceParams.noise_level);
    mask(i) = 0;          % new mask
    img = figs(:,:,a).*mask;     % re-apply mask
    img = img(:);         % flatten
    [Fmapi j v] = find(img);     % indices of pixels that matter
    if( isempty(Fmapi) )         % abort because nothing to do?
      Fmap = ''; Fmapi = ''; pfitted = '';
      return;
    end;
```

```
    % Get Sc
    img = figs(:,:,dataParams.data_sequence(1));
    img = img(:);
    Sc_src = img(Fmapi);

    % Get Si
    img = figs(:,:,dataParams.data_sequence(2));
    img = img(:);
    Si_src = img(Fmapi);

    % Get T1map
    img = figs(:,:,3);
    img = img(:);
    t1_src = img(Fmapi);

    % T1 map may not be available for all pixels in the ROIs
    [t1void_i j v] = find(t1_src==0);
    t1_src2 = t1_src;
    t1_src2(t1void_i) = 1;  % to avoid divide by zero

    clear img a i j v mask

    pfitted = zeros(length(Fmapi),4);

    % Do the calculations
    pfitted(:,1) = Sc_src - Si_src;
    pfitted(:,2) = 6000*dataParams.lambda*(Sc_src-Si_src) ./ ...
    (2*dataParams.alpha*(t1_src2 .* Sc_src));
    pfitted(t1void_i,2) = 0;
    pfitted(:,3) = t1_src;
    pfitted(:,4) = 0;       % Error map (set to zero)

    % save the data

    F_values = pfitted(:,2);
    Fmap = zeros(sourceParams.msize(1)*sourceParams.msize(2),1);
    Fmap(Fmapi) = F_values;
    Fmap = reshape(Fmap, sourceParams.msize(1), sourceParams.msize(2));
```

```
function [t1map, t1mapi, pfitted] = exe_sr( mask )
%EXE_SR

  global figs sourceParams dataParams

  % for fminsearch
  %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);
```

```
    % mask out the pixels that are < noise threshold
    a = dataParams.data_sequence(end);
    temp1 = figs(:,:,a).*mask;
    [Im Jm Vm] = find(temp1 < sourceParams.noise_level);

    mask( Im ) = 0;

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

    clear mask

    [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:length(dataParams.data_sequence)
      Data(:,n) = Imasked(I,n);
    end

    clear Imasked J V

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

    h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
    for n = 1:s,
      SI = Data(n,:);
      [pfitted(n,1:2) fval exitflag output] = ...
      fminsearch( 'excised_t1_fun_sr', dataParams.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
delete( h );  %changed from close to delete 31.12.2004

    % save the data

    t1_values = pfitted(:,2);
    t1map = zeros(sourceParams.msize(1)*sourceParams.msize(2),1);
    t1map(I) = t1_values;
    t1map = reshape(t1map, sourceParams.msize(1),
sourceParams.msize(2));
    t1mapi = I;     % index: pixels in the image which correspond to
                        % t1map and pfitted data
```

```
function [t2map, t2mapi, pfitted] = exe_t2( mask )
%EXE_T2

  global figs sourceParams dataParams

  % for fminsearch
  OPTIONS = optimset( 'TolFun', .02, 'TolX', .02, 'Display', 'off' );

  % pick out regions that are masked and above the noise theshold
  %mask = double(mask(:));
  mask = double(mask);
  a = dataParams.data_sequence(length(dataParams.data_sequence));
  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
  nn = 1;
  for n = dataParams.data_sequence
    temp = figs(:,:,n);
    temp = temp(:);
    Data(:,nn) = temp(roind);
    nn = nn+1;
  end;

  TE = dataParams.TE(dataParams.data_sequence);

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

  h = waitbar( 0, ['Calculating region (' int2str(s) ' pixels)'] );
  for n = 1:s,
    SI = Data(n,:);
    [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,1:2); % use last results as initial guess
    if( mod(n,10)==0 )
      waitbar( n/s, h );
    end;
  end
delete( h );  %changed from close to delete 31.12.2004
  clear Data;

  t2_values = pfitted(:,2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    % save the data

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

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

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



function ret = exportFig( h, varargin )
% EXPORTFIG Dialog box prompting user what format to save figure window
to.
%
%   RET = EXPORTFIG( H ) first brings up a dialog box asking the user
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.
%
%   RET = EXPORTFIG( H, MAPDATA ) to allow saving to text file

  formatSwitch = { '-djpeg75', ...
           '-djpeg90', ...
           '-dtiff', ...
           '-dpng', ...
           '-dps', ...
           '-dpsc', ...
           '-dps2', ...
           '-dpsc2', ...
           '-deps', ...
           '-depsc', ...
           '-deps2', ...
           '-depsc2', ...
           '-dmfile', ...
           'void' ...
         };
  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', ...
        '*.txt' ...
          };


  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', ...
      'Raw text file'
    };

  if( nargin==1 )
    str = str{1:length(str)-1};     % Remove "Raw text file" option
  end;

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

  if( ok )

    if( strcmp( str{selection}, 'Raw text file' ))
      result = questdlg( 'Export what data?', 'Raw text export', ...
             'Just the map', 'Entire matrix', 'Just the map' );
      saveData = varargin{1};
      if( strcmp( result, 'Just the map' ))
    saveData = find( saveData(:) );
      end;
      [filen, pathn] = uiputfile( formatMask{selection}, ...
                  ['Save ' str{selection}] );
      if( filen~=0 )
    saveFile = fullfile(pathn, filen);
    save( saveFile, 'saveData', '-ascii' );
      else
    ok = 0;
      end;

    else
      [filen, pathn] = uiputfile( formatMask{selection}, ...  %standard
filesave dialog box
                  ['Save ' str{selection}] );
      if( filen~=0 )
    saveFile = fullfile(pathn, filen);
    print( h, formatSwitch{selection}, saveFile );
```

```
      else
    ok = 0;
      end;
    end;
  end;

  ret = ok;
```

```
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.

  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));
  if( isempty(a) )
    maxVal = binSize;   % degenerate case
  else
    a = a(length(a));           % index of bin where cutoff is
    maxVal = binSize*a;
  end;
  if( isempty(maxVal) )
    maxVal = 0;
  end;
  % ...............................................................

  maxVal = maxVal * 1.2;        % add an upper cushion
```

```
function [y,x] = getyx( i )
% GETYX Converts linear index to row,col
%
%   [Y,X] = GETYX(I)
```

```
  global sourceParams;

  msize = sourceParams.msize;
  y = mod( i, msize(1));
  if( y==0 ) y = msize(1); end;
  x = floor((i-1)/msize(1))+1;
```

```
% INCLUDE_GLOBALS

global figs sourceParams dataParams ren_struct viewParams
img_background ...
    ROIlist theMapSave theMapi pfittedSave fhm
global sourceValid figsValid ROIlistValid theMapSaveValid ...
    saveFile doSaveSrcLocal
```

```
function ret = inputFieldDlg( theStruct, fieldname, varargin )
% INPUTFIELDDLG Get user input for a structure's field.


  % 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 = ['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 H = intFilter( radius )
% INTFILTER  Interpolation filter
%   H = INTFILTER( RADIUS )
%   The interpolation filter is a 2-D Gaussian PDF with the origin's
%   value zeroed and distributed evenly.  The standard deviation is
%   RADIUS and the filter is calculated over the range -RADIUS to
%   RADIUS on both axes.

  r2 = round(radius*2);
  if( r2==0 )
    r2=1;
  end;
  x = (-r2):1:(r2);

  base = normpdf( x, 0, radius );
  H = zeros(length(x));
  for y=x
    index = y+r2+1;
    H(index,:) = base(index)*base;
  end;

  a = H(r2+1,r2+1);
  H = H + a/((r2*2+1)^2-1);
  H(r2+1,r2+1) = 0;
```

```
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 ret = load_bruker
%LOAD_BRUKER
```

```
  include_globals;

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

  msize = sourceParams.msize;
  no_of_timeVar = feval(['def_' dataParams.type],'get_no_of_timeVar');
  filen = fullfile(sourceParams.sourceDir, sourceParams.sourceFile);

  endian = sourceParams.byte_ordering(1);

  [data, msg] = fopen(filen, 'r', endian);
  if( data==-1 )
delete( h );  %changed from close to delete 31.12.2004
    lasterr( ['Could not open ' filen '.  ' msg] );
    ret = 0;
    return;
  end;

  if( isfield(sourceParams,'offset') )
    if( sourceParams.offset>0 )
      advance = sourceParams.offset*msize(2);
      fread( data, advance, 'long' );
    end;
  end;

  figs = zeros(msize(1),msize(2), no_of_timeVar);
  if( strcmp(dataParams.type,'perfusion' ))
    no_of_timeVar = 2;
  end;

  switch( dataParams.type )
   case 't2'
    for slice_no = 1:sourceParams.slice
      temp = fread(data,[msize(1)*msize(2) no_of_timeVar], 'long');
    end
    for te_num = 1:no_of_timeVar
      ima = reshape(temp(:,te_num), msize(2), msize(1));
      ima = ima'*sourceParams.dataScale;
      figs(:,:,te_num) = ima;
    end;
   otherwise
    for num = 1:no_of_timeVar
      temp = fread(data,[msize(1)*msize(2) sourceParams.no_of_slices],
'long');
      ima = reshape(temp(:,sourceParams.slice), msize(2), msize(1));
      ima = ima'*sourceParams.dataScale;
      figs(:,:,num) = ima;
    end;
  end;
  fclose(data);

  % For perfusion, load T1map source and ROIlist
  if( strcmp(dataParams.type,'perfusion' ))
    figs(:,:,3) = getfield( load( dataParams.T1map_source,
'theMapSave'), ...
                'theMapSave' );
    theMapi = getfield(load( dataParams.T1map_source, 'theMapi' ), ...
```

```
            'theMapi' );
    ROIlist = getfield(load( dataParams.T1map_source, 'ROIlist' ), ...
                'ROIlist' );
    ROIlistValid = 2;
    for( i=1:length(ROIlist))       % set dirty
      ROIlist(i).modified = 1;
    end;
  end;

  if( ishandle(h) ), delete(h);, end;   %change from close to delete
06.01.2005

  ret = 1;



unction ret = load_ge
% LOAD_GE

% GE and Siemens imaging systems have file formats that are very
similar.

  ret = load_siemens( 'ge' );



function ret = load_siemens( 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;

  msize = sourceParams.msize;
  no_of_timeVar = feval(['def_' dataParams.type],'get_no_of_timeVar');
  figs=zeros(msize(1),msize(2),no_of_timeVar);

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

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

    endian = sourceParams.byte_ordering(1);
    [fid, msg] = fopen( filen, 'r', endian);
    if( fid==-1 )
      lasterr( ['Could not open ' filen '.  ' msg] );
delete( h );  %changed from close to delete 31.12.2004
      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(filen,'r');
      waitbar( n/no_of_timeVar );
      figs(:,:,n) = dicomread(filen);
    else
    %end;

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

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

  figs = figs .* sourceParams.dataScale;

  % For perfusion, load T1map source and ROIlist
  %  ** Haven't tested perfusion and Siemens data yet
  if( strcmp(dataParams.type,'perfusion' ))
    no_of_timeVar = 3;
    figs(:,:,3) = getfield( load( dataParams.T1map_source,
'theMapSave'), ...
                'theMapSave' );
    theMapi = load( dataParams.T1map_source, 'theMapi' );
    ROIlist = load( dataParams.T1map_source, 'ROIlist' );
    ROIlistValid = 2;
  end;

delete( h );  %changed from close to delete 31.12.2004

  ret = 1;


function loadoldfile( action )
% LOADOLDFILE Utility program
%
%  This program converts older .MAT results files into files that can
%  be used by current versions of the mappin program.  Specifically,
```

```
%  the names of variables and fields have changed, so this program
%  maps the old variables and fields to new ones.

%  Old variables:
%   'sourceParams', 'dataParams', 'ROIlist', 't1mapSave', ...
%     't1mapi', 'pfittedSave', 'img_background'
%
%  Current variables:
%   'sourceParams', 'dataParams', 'viewParams', 'ROIlist',
'theMapSave',
%   'theMapi', 'pfittedSave', 'img_background'

  persistent dataSource_h new_saveFile;

  include_globals;

  if( nargin>0 )
    switch( action )
     case 'proceed'  % When run_dataSource is all done
      delete( dataSource_h );    %30.12.2004  changed close to delete

      % save to new saveFile
      saveFile = new_saveFile;
      disp( [' Saving to ' saveFile] );
      save( saveFile, 'sourceParams', 'dataParams', 'viewParams', ...
        'ROIlist', 'theMapSave', 'theMapi', 'pfittedSave', ...
        'img_background' );
      sourceValid = 1;
      ROIlistValid = 2;
      theMapSaveValid = 2;
      run_main( 'update' );
      run_viewResults( 'init' );

      msgbox( ['The new save file is ' saveFile], 'Note', 'modal' );

    end;
    return;
  end;

  run_dataSource( 'close' );        % going to use this dialog box
  run_ROIlist( 'close' );
  run_editSource( 'close' );
  %run_viewResults( 'close' );  %no need because it is attached to main
  %13.01.2005
  run_image( 'close' );

  [filen dirn] = uigetfile( '*.mat', 'Open Calculation Results' );
  if( filen==0 ) return; end;
  loadFile = fullfile(dirn,filen);
  s = load( loadFile );
  [p,n,e,v] = fileparts(filen);     % make new file
  new_filen = [n '_new' e];
  new_saveFile = fullfile(dirn,new_filen);

  disp( 'Original sourceParams:' );
  disp( s.sourceParams );
  disp( 'Original dataParams:' );
```

```
  disp( s.dataParams );

  % retrieve other variables ----------------------------------------
-
  ROIlist = s.ROIlist;
  if( isfield(s,'theMapSave') )
    theMapSave = s.theMapSave;
    theMapi = s.theMapi;
  elseif( isfield(s,'t1MapSave') )
    theMapSave = s.t1MapSave;
    theMapi = s.t1Mapi;
  end;
  pfittedSave = double(s.pfittedSave);
  img_background = s.img_background;

  if( isfield( s.sourceParams, 'magnet' ))

    % Figure out source type (bruker,biospec,ge,siemens) -------------
-
    magnet = s.sourceParams.magnet;
    if( strcmp(magnet(1:6),'bruker' ))
      sourceType = 'bruker';
    elseif( strcmp(magnet(1:7),'biospec' ))
      sourceType = 'bruker';
    elseif( strcmp(magnet(1:2),'ge'))
      sourceType = 'ge';
    elseif( strcmp(magnet(1:7),'siemens' ))
      sourceType = 'siemens';
    elseif( strcmp(magnet(1:7),'siemans' ))
      sourceType = 'siemens';
    elseif( strcmp(magnet(1:7),'seimens' ))
      sourceType = 'siemens';
    elseif( strcmp(magnet(1:7),'seimans' ))
      sourceType = 'siemens';
    else
      error( ['Unrecognized magnet type ' magnet] );
    end;

    % Figure out data type (IR, SR, T2) ------------------------------
-
    dataType = lower(magnet((length(magnet)-1):length(magnet)));

  else
    try
      sourceType = s.sourceParams.type;
      dataType = s.dataParams.type;
    catch
      error( ['Could not identify source data type'] )
    end;
  end;

  % use defaults as templates ----------------------------------------
-
  feval( ['def_' sourceType], 'default' );
  feval( ['def_', dataType], 'default' );
  viewParams = def_view( 'new' );  %07.02.2005 changed from 'default'
to 'new'
```

```
  % match up the fields in sourceParams and dataParams ----------------
-
  fields = fieldnames(s.sourceParams);
  for i = 1:length(fields)
    field = fields{i};
    if( ~strcmp( field, 'type' ))
      data = getfield( s.sourceParams, field );
      if( isfield(sourceParams,field) )
    sourceParams = setfield( sourceParams, field, data );
      elseif( isfield(dataParams,field) )
    dataParams = setfield( dataParams, field, data );
      else
    disp( ['Not matched: old sourceParams.' field ':'] );
    disp( data );
      end;
    end; %if
  end; %for

  fields = fieldnames(s.dataParams);
  for i = 1:length(fields)
    field = fields{i};
    if( ~strcmp( field, 'type' ))
      data = getfield( s.dataParams, field );
      if( isfield(sourceParams,field) )
    sourceParams = setfield( sourceParams, field, data );
      elseif( isfield(dataParams,field) )
    dataParams = setfield( dataParams, field, data );
      else
    disp( ['Not matched: old dataParams.' field ':'] );
    disp( data );
      end;
    end; %if
  end; %for

  % fix up some of the fields  ---------------------------------------
-
  % fix no_of_TI,TR,TE
  switch( dataParams.type )
   case 'ir'
    dataParams.no_of_TI = length(dataParams.TI);
   case 'sr'
    dataParams.no_of_TR = length(dataParams.TR);
   case 't2'
    dataParams.no_of_TE = length(dataParams.TE);
  end;

  % fix msize
  sourceParams.msize = size(img_background);

  disp( 'new sourceParams:' );
  disp( sourceParams );
  disp( 'new dataParams:' );
  disp( dataParams );

  sourceValid = 1;
  figsValid = 0;
```

```
  ROIlistValid = 0;
  theMapSaveValid = 0;

  % use dataSource dialog box to edit the values
  dataSource_h = run_dataSource( 'init' );
  push_close = findobj( allchild(dataSource_h), 'Tag', 'push_close' );
  set( push_close, 'Callback', 'loadoldfile(''proceed'');' );
  push_load = findobj( allchild(dataSource_h), 'Tag', 'push_load' );
  set( push_load, 'Visible', 'off' );

  msgbox( strvcat('Use the Data Source dialog box to edit any
parameters', ...
          'as needed.  Click Close when done.'), ...
      'Instructions', 'modal');



function ret = loadSourceData( handle, varargin )
% LOADSOURCEDATA Loads MRI source data.
%
%   RET = LOADSOURCEDATA( handle )
%   RET = LOADSOURCEDATA( handle, ERRORMESSAGE )
%
%   RET=1 on success, otherwise 0.

  global sourceParams fhm
%the handle here is the main window handle  (better to just not have
input
%to this function???)
  handle=fhm;
  axis off;  %added 20.05.2005  don't want axis in background of figure
  if( nargin>1 )
    errortext = varargin{1};
  else
    errortext = 'Error loading MRI source data';
  end;

  set(fhm,'Pointer','watch');
  try
  lasterr( '' );
    ret = feval( ['load_' sourceParams.type] );
  catch
    ret = 0;
  end;
  set(fhm,'Pointer','arrow');
  if( ret==0 )
    closeWaitBars;
    errordlg( strvcat(errortext, lasterr), 'Error', 'modal' );
  end;


function closeWaitBars
% CLOSEWAITBARS Closes all waitbar figures
%

  hlist = allchild(0);
```

```
  for i = 1:length(hlist)
    h = hlist(i);
    if( strcmp(get(h,'Tag'),'TMWWaitbar' ))
delete( h );  %changed from close to delete 31.12.2004
    end;
  end;
```

```
function F= Meqfun(x,flip_angles)
F=(x(1)*(1-exp(-x(2))).*sin(flip_angles))./(1-exp(-
x(2)).*cos(flip_angles));
```

```
% MRIMAPPER  Program for quantitative mapping of MRI parameters
%
% While the program is running, the following variables are available
% for analyzing at the command line:
%
%   sourceParams   - Source file parameters
%   dataParams     - MRI data parameters
%   viewParams     - Viewing parameters (do not affect analysis)
%   figs           - MRI source data
%   theMapSave     - MRI calculated map (subset of pfittedSave)
%   pfittedSave    - MRI calculated data

% Log
%
% 7/14/02 Release
%       Data types: IR, SR, T2, diffusion*, perfusion*
%       Source types: Bruker, Siemens, GE
%       Features: image alignment, results analysis
%       * Not completely validated yet
%       Bruce Po - igan@mit.edu

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

```
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.

  if( nargin==0 )
    h = gcf;
  else
```

```
    h = varargin{1};
  end;

  figure( h );
  colorbar;

  figSettings = get( h, 'Userdata' );
  if( isempty(figSettings) )
    cbScale = 1;
    cbMin = 0;
  else
    cbScale = figSettings.colorbarScale;
    cbMin = figSettings.colorbarStartVal;
  end;

  ah = colorbar;
  % The colorbar command by default spaces ticks 50 units apart within
  % the range of 0 to 255.
  set( ah, 'YTicklabel', strvcat( num2str( cbMin,3 ), ...
                  num2str( 1/cbScale*50+cbMin,3 ), ...
                  num2str( 1/cbScale*100+cbMin,3 ), ...
                  num2str( 1/cbScale*150+cbMin,3 ), ...
                  num2str( 1/cbScale*200+cbMin,3 ), ...
                  num2str( 1/cbScale*250+cbMin,3 )));
  set( ah, 'YTick', [1 50 100 150 200 250 300] );



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 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_despot( dataPoints, p, varargin )
  include_globals

  %code taken from plot_ir  09.02.2005

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

  TI = dataParams.TI(dataParams.data_sequence);
  timeIndex = 0:.01:max(TI);
  timeIndex = reshape(timeIndex, length(timeIndex), 1);     % columnize
it

  if( isempty(p) )
    plotIt( logScale, TI, dataPoints, '.' );
  else
    % generate the fitted curve
    %timeIndex = 0:.01:dataParams.TR;   % [seconds]
    curve = despot_fn( timeIndex, p );  %09.02.2005
    %for testing
    %fake1 = ir_fn( timeIndex, [8968.3226 0.86986509 398.69779
0.80531766 111]);
    %fake2 = ir_fn( timeIndex, [8968 0 398 0 111]);
    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 = despot_fn( timeindex, p )   %09.02.2005
  % t is time, p is parameters
  global dataParams;

  %print out parameters and their class to see what type they are
  %integer precision is not enough to give an accurate fit
  %parameters=p
  %classParameters=class(p)
```

```
  TI = dataParams.TI;     % If TR is multi-TR, then should be same size
as TI
  TR = dataParams.TR;

  % If TR is multi, then make a interpolate to get a TR the same size
as t
  TI = dataParams.TI;     % If TR is multi-TR, then should be same size
as TI
  TR = dataParams.TR;

  % If TR is multi, then make a continuous time "stepped" TR
  if( length(TR)>1 )
    % Create linearly interpolated new_TR based on multi-TR and TI

    TR_index = 1;
    TR_interp_index = 1;
    new_TR = zeros(length(timeindex),1);

    % initial
    m = (TR(2)-TR(1))/(TI(2)-TI(1));
    b = TR(1) - m*TI(1);
    prev_TI = 0;
    next_TI = TI(1);
    next_TR = TR(1);

    for i = 1:length(timeindex)
      t = timeindex(i);     % Something weird in Matlab preventing me
from doing 'for t = timeindex'
      if( t>=next_TI )
    TR_index = TR_index+1;
    prev_TR = next_TR;
    prev_TI = next_TI;
    if( TR_index<length(TR) )
      next_TR = TR(TR_index);
      next_TI = TI(TR_index);
    else
      next_TR = TR(end);
      next_TI = TI(end);
    end;

    % line equation
    m = (next_TR - prev_TR) / (next_TI - prev_TI );
    b = prev_TR;
      end;

      x = t-prev_TI;
      new_TR(TR_interp_index) = m*x + b;
      TR_interp_index = TR_interp_index + 1;
    end;
    TR = new_TR;
  end;

  TR = TR*1000;          % convert to miliseconds
  t = timeindex*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_diffusion( dataPoints, p, varargin )
  include_globals

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

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

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

function ret = fn(b, p)
  % b is sort of time, p(2) is the parameter D

  ret = p(1)*exp( -b * p(2));



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



function plot_ir( 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);
  timeIndex = reshape(timeIndex, length(timeIndex), 1);     % columnize
it

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

```
    curve = ir_fn( timeIndex, p );
    %for testing
    %fake1 = ir_fn( timeIndex, [8968.3226 0.86986509 398.69779
0.80531766 111]);
    %fake2 = ir_fn( timeIndex, [8968 0 398 0 111]);
    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 = ir_fn( timeindex, p )
  % t is time, p is parameters
  global dataParams;

  %print out parameters and their class to see what type they are
  %integer precision is not enough to give an accurate fit
  %parameters=p
  %classParameters=class(p)

  TI = dataParams.TI;     % If TR is multi-TR, then should be same size
as TI
  TR = dataParams.TR;

  % If TR is multi, then make a interpolate to get a TR the same size
as t
  TI = dataParams.TI;     % If TR is multi-TR, then should be same size
as TI
  TR = dataParams.TR;

  % If TR is multi, then make a continuous time "stepped" TR
  if( length(TR)>1 )
    % Create linearly interpolated new_TR based on multi-TR and TI

    TR_index = 1;
    TR_interp_index = 1;
    new_TR = zeros(length(timeindex),1);

    % initial
    m = (TR(2)-TR(1))/(TI(2)-TI(1));
    b = TR(1) - m*TI(1);
    prev_TI = 0;
    next_TI = TI(1);
    next_TR = TR(1);

    for i = 1:length(timeindex)
      t = timeindex(i);     % Something weird in Matlab preventing me
from doing 'for t = timeindex'
      if( t>=next_TI )
    TR_index = TR_index+1;
    prev_TR = next_TR;
    prev_TI = next_TI;
    if( TR_index<length(TR) )
      next_TR = TR(TR_index);
```

```
      next_TI = TI(TR_index);
    else
      next_TR = TR(end);
      next_TI = TI(end);
    end;

    % line equation
    m = (next_TR - prev_TR) / (next_TI - prev_TI );
    b = prev_TR;
      end;

      x = t-prev_TI;
      new_TR(TR_interp_index) = m*x + b;
      TR_interp_index = TR_interp_index + 1;
    end;
    TR = new_TR;
  end;

  TR = TR*1000;          % convert to miliseconds
  t = timeindex*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_perfusion( dataPoints, p, varargin )
  persistent ok
  global dataParams

  axis off
  if( ~isempty(dataPoints) )
    text( 0, 1, ['Sc:  ' num2str(dataPoints(1))] );
    text( 0, .9,    ['Si:  ' num2str(dataPoints(2))] );
  end;
  if( ~isempty(p))
    text( 0, .8, ['Sc-Si:  ' num2str(p(1))] );
    text( 0, .7, ['T1:  ' num2str(p(3))] );
    text( 0, .6, ['Perfusion:  ' num2str(p(2))] );
  end

  if( isempty( ok ))
    msgbox( ['Graphical plot function is unavailable for ' ...
         'perfusion calculation.'], ...
        'Note', 'modal');
  end;
  ok = 1;



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))));



function plot_sr( dataPoints, p, varargin )
  global dataParams

  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 = sr_eqn( timeIndex, p );
    if( isempty(dataPoints) )
      plotIt( logScale, timeIndex, curve, '-' );
    else
      plotIt( logScale, TR, dataPoints, '.', timeIndex, curve, '-' );
    end;
  end; %if

function ret = sr_eqn(t, p)
  % t is time, p is parameters
```

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


function plot_t2( 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 = t2_eqn( 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 = t2_eqn(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 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;



% Rename_DICOM.m
% Ashley, 3/23/05
% Linearly interpolate image files and save in DICOM format
% Note:  change the interp_factor to interpolate the image size
%        to keep the same resolution, type "1" at the prompt
```

```
echo off
%clear
%close all

%interp_factor = input('Interpolate by how much?  (Type "1" to keep
resolution the same, < 1 to decrease, > 1 to increase) ');
%interp_factor = 1;

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

    [filenames, pathname] = uigetfile('*.*', 'Pick the DICOM files to
be reordered', 'Multiselect', 'on');
    newPathName=uigetdir('','Select the destination folder for the re-
ordered files');
    number_of_files = size(filenames,(2));
    unranked_data = zeros(number_of_files,2);
    h = waitbar(0,'Please wait...eFILM files being re-ordered...');
    filenames=sort(filenames);
    for n=1:number_of_files;
       imagename = char(filenames(1,n));
       fullfilename = fullfile(pathname, imagename);

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Read in Data from a GE (ie. DICOM) file:

        %header=dicominfo(fullfilename);
        %matrix = header.Height;
        %input_file=double(dicomread(fullfilename));
        %input_file(:,:) = reshape(input_file,matrix, matrix);
        metadata(n) = dicominfo(fullfilename);
        position=metadata(n).InstanceNumber;    %slice location also
works
        unranked_data(n,1)=n;
        unranked_data(n,2)=position;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Linearly interpolate the input image and save as dicom with
the
        % same name

        %zipped_up_file(:,:) = imresize(input_file(:,:),
interp_factor);
        %uint16_zipped_up_file(:,:,n)=uint16(zipped_up_file(:,:));

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

        clear filename input_file matrix fullfilename position
        waitbar(n/(number_of_files*2),h);
    end  %for n

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now rank the data based on slice location
```

```
     ranked_data = sortrows(unranked_data,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Save the images with their headers in order of thier spatial
locations with a new filename
  for n=1:number_of_files;
       if n<10;
          sname=['Image','_00', num2str(n)];  %took out extension
18.05.2005
       elseif n<100;
         sname=['Image','_0', num2str(n)];  %took out extension
18.05.2005
       else
           sname=['Image','_', num2str(n)];
       end
          old_file_number = ranked_data(n,1);
          imagename = char(filenames(1,old_file_number));
          source=fullfile(pathname, imagename);
          destination=fullfile(newPathName,sname);
          copyfile(source,destination);
          %dicomwrite(uint16_zipped_up_file(:,:,old_file_number),
fullfile(newPathName,sname), metadata(old_file_number));
   waitbar((n+number_of_files)/(number_of_files*2),h);
  end
  close(h)



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);




% SAVEPPT saves plots to PowerPoint.
% function SAVEPPT(filespec,title,prnopt) saves the current Matlab
figure
%  window or Simulink model window to a PowerPoint file designated by
%  filespec.  If filespec is omitted, the user is prompted to enter
%  one via UIPUTFILE.  If the path is omitted from filespec, the
%  PowerPoint file is created in the current Matlab working directory.
%
%  Optional input argument title will add a title to the PowerPoint
slide.
%
%  Optional input argument prnopt is used to specify additional save
%  options:
%    -fHandle   Handle of figure window to save
%    -sName     Name of Simulink model window to save
%
%  Examples:
%  >> saveppt
%       Prompts user for valid filename and saves current figure
```

```
%  >> saveppt('junk.ppt')
%       Saves current figure to PowerPoint file called junk.ppt
%  >> saveppt('junk.ppt','Stock Price','-f3')
%       Saves figure #3 to PowerPoint file called junk.ppt with a
%       slide title of "Stock Price".
%  >> saveppt('models.ppt','System Block Diagram','-sMainBlock')
%       Saves Simulink model named "MainBlock" to file called
models.ppt
%       with a slide title of "System Block Diagram".
%
%  The command-line method of invoking SAVEPPT will also work:
%  >> saveppt models.ppt 'System Block Diagram' -sMainBlock
%
%  However, if you want to save a specific figure or simulink model
%  without title text, you must use the function-call method:
%  >> saveppt('models.ppt','','-f8')

%Ver 2.0, Copyright 2001, Mark W. Brown, mwbrown@ieee.org
%  changed slide type to include title.
%  added input parameter for title text.
%  added support for int32 and single data types for Matlab 6.0
% modified by Meredith Gerber, gerber@mit.edu, February 2, 2005
% does not close Powerpoint application when completed, but closes
specific
% file.
% opens powerpoint application earlier in program.

function saveppt(filespec,titletext,prnopt)

%02.02.2005 moved up from a bit later in the program.  Open ppt first
% Start an ActiveX session with PowerPoint:
ppt = actxserver('PowerPoint.Application');
ppt.Visible = 1;

% Establish valid file name:
if nargin<1 | isempty(filespec);
  [fname, fpath] = uiputfile('*.ppt');
  if fpath == 0; return; end
  filespec = fullfile(fpath,fname);
else
  [fpath,fname,fext] = fileparts(filespec);
  if isempty(fpath); fpath = pwd; end
  if isempty(fext); fext = '.ppt'; end
  filespec = fullfile(fpath,[fname,fext]);
end

% Default title text:
if nargin<2
  titletext = '';
end

% Capture current figure/model into clipboard:
if nargin<3
  print -dmeta
else
  print('-dmeta',prnopt)
end
```

```
if ~exist(filespec,'file');
  % Create new presentation:
  op = invoke(ppt.Presentations,'Add');
else
  % Open existing presentation:
  op = invoke(ppt.Presentations,'Open',filespec);
end

% Get current number of slides:
slide_count = get(op.Slides,'Count');

% Add a new slide (with title object):
slide_count = int32(double(slide_count)+1);
new_slide = invoke(op.Slides,'Add',slide_count,11);

% Insert text into the title object:
set(new_slide.Shapes.Title.TextFrame.TextRange,'Text',titletext);

% Get height and width of slide:
slide_H = op.PageSetup.SlideHeight;
slide_W = op.PageSetup.SlideWidth;

% Paste the contents of the Clipboard:
pic1 = invoke(new_slide.Shapes,'Paste');

% Get height and width of picture:
pic_H = get(pic1,'Height');
pic_W = get(pic1,'Width');

% Center picture on page:
set(pic1,'Left',single((double(slide_W) - double(pic_W))/2));
set(pic1,'Top',single(double(slide_H) - double(pic_H)));

if ~exist(filespec,'file')
  % Save file as new:
  invoke(op,'SaveAs',filespec,1);
else
  % Save existing file:
  invoke(op,'Save');
end

%02.02.2005  Powerpoint window needs to be open to save something to
%powerpoint, so don't close it.  But, if the same file is open that you
%to save to, there will be problems.

% Close the presentation window:
invoke(op,'Close');

% Quit PowerPoint
%invoke(ppt,'Quit');

% Close PowerPoint and terminate ActiveX:
%delete(ppt);

return
```

```
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.  CALLER_FUN is a
string
%   with the name of the callback function.  The callback function is
%   invoked as CALLER_FUN(tag), where tag is the tag of the UI control.
global childs

  set( fh, 'CloseRequestFcn', [callerFun '(''close'');']);
  %should this be changed to delete to avoid that anti-close message
from Matlab? 30.12.2004
  %no, because this just makes the fuction go to the case named 'close'
  %what matters is what happens there, whether the command 'close' is
  %called or not.
  objList = get( fh, 'Children' );
  for i=1:length(objList)
    obj = objList(i);
    %if ~ismember(obj,childs)  %if the handle is already in the childs
list
        %then do not re-define it. not going to work because all of the
        %handles are in childs, they just don't all have valid
callbacks.
    if( ~strcmp(get(obj,'Type'),'axes') )
      set( obj, 'Callback', ...
        [callerFun '(''' get(obj,'Tag') ''');'] );
    end;
  end;
  %end;



function ret = struct2string( src_struct, varargin )
%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.
%
%   X = STRUCT2STRING( S, 'no_' ) removes the underscore character
%   X = STRUCT2STRING( S, ren_struct ) renames fields when
%     printing.  To rename a field 'field1' to 'field one':
%     ren_struct.field1 = 'field one'.


  src_fields = fieldnames(src_struct);
  vals = '';
  for i=1:length(src_fields)
    value = getfield( src_struct, src_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 = [strvcat(src_fields) vals];

  % process optional arguments
  if( nargin>1 )

    % Rename field names
    for j=2:nargin
      if( isstruct(varargin{j-1} ))
    ren_struct = varargin{j-1};
    for i=1:size(ret,1)
      if( isfield( ren_struct, src_fields{i} ))
        src_fields{i} = getfield( ren_struct, src_fields{i} );
      end;
    end;
    ret = [strvcat(src_fields) vals ];
      end;
    end;

    % Remove underscore
    for j=2:nargin
      if( ischar( varargin{j-1} ))
    if( strcmp(varargin{j-1},'no_') )
      for i=1:size(ret,1)
        temp = ret(i,:);
        ret(i,:) = strrep(temp, '_', ' ');
      end;
    end;
      end;
    end;

  end;


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.
```

```
  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.

  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;



function update_displayMap( handle )
%UPDATE_TEMPMAP

% Creates temporary T1 (or T2) map for displaying and analysis.

% fhm - figure handle for ui_main and contains fields for selecting
what
%   parts to mask.
```

```
  include_globals;
  global fhm selectedMapi tempMapi tempMap thisROIlist textLabels;

  %disp('in update_displayMap--beginning')  %for debug 30.12.2004

  handle=fhm;
  axis off;  %added 20.05.2005  don't want axis in background of figure

  % get handles to UI controls
  obj_list = allchild(fhm);
  list_ROIlist = findobj( obj_list, 'Tag', 'list_ROIlist' );

  % ------------------------------------------------------------------
  % identify selected ROIs
  % ------------------------------------------------------------------
  % tempMap and tempMapi is a union of the selected ROIs
  selectedROIs = get( list_ROIlist, 'Value' );
  selectedMapi = unionall( thisROIlist(selectedROIs).maski, [] );

  % ------------------------------------------------------------------
  % process all ROIs
  % ------------------------------------------------------------------

  %fhm = gcf;  30.12.2004 commented out; unnecessary because fhm is
global.
  set( fhm, 'Pointer', 'watch' );

  map = theMapSave(:);
  savei = theMapi;
  interpolatei = [];

  % bad-fit pixels
  errorParam = feval( ['def_', dataParams.type], 'get_errorParam');
  errorMap = pfittedSave(:,errorParam);
  i = find(errorMap<=viewParams.maxErrorVal);
  run_filterPixels( 'pixelCount', 'Pmin', ...
            length(setdiff(selectedMapi, theMapi(i))));
  if( strcmpi(viewParams.maxErrorValAction,'remove') )
    savei = savei(i);
  elseif( strcmpi(viewParams.maxErrorValAction,'interpolate' ))
    interpolatei = union( interpolatei, setdiff(theMapi,theMapi(i)) );
  end;


  fitParam = feval( ['def_', dataParams.type], 'get_fitParam');

  % below T1 minimum pixels
  i = find(pfittedSave(:,fitParam)>=viewParams.minTval);
  i = theMapi(i);
  %i = find(map>=viewParams.minTval);
  run_filterPixels( 'pixelCount', 'Tmin', ...
            length( setdiff(selectedMapi, i)) );
  if( strcmpi(viewParams.minTvalAction,'remove') )
    savei = intersect(savei, i );
  elseif( strcmpi(viewParams.minTvalAction,'interpolate' ))
    interpolatei = union( interpolatei, setdiff(theMapi,i) );
```

```
  end;

  % above T1 maximum pixels
  i = find(pfittedSave(:,fitParam)<=viewParams.maxTval);
  i = theMapi(i);
  %i = find(map<=viewParams.maxTval);
  run_filterPixels( 'pixelCount', 'Tmax', ...
            length( setdiff(selectedMapi, i)) );
  if( strcmpi(viewParams.maxTvalAction,'remove') )
    savei = intersect(savei, i);
  elseif( strcmpi(viewParams.maxTvalAction,'interpolate' ))
    interpolatei = union( interpolatei, setdiff(theMapi,i) );
  end;

  % interpolate "bad" pixels
  if( ~isempty(interpolatei) )
    map(interpolatei) = 0;  % zero the bad pixels first
    X = reshape( map, sourceParams.msize );
    h = intFilter( viewParams.interpolationRadius );    % interpolation
filter
    A = conv2( X, h, 'same' );              % do it
    A = A(:);
    map(interpolatei) = A(interpolatei);        % updated bad pixels
  end;

  % update listbox showing list of ROIs
  oldStr = get( list_ROIlist, 'String' );
  num_ROIs = size(oldStr, 1);  %26.01.05  this use of size returns
number of rows.
  all_ROIs = 1:num_ROIs;
  %newStr = {'ROILabel' 'pixels' 'mean' 'std.dev.'};  %5.01.2005
  newStr = {};  %05.01.2005
  for ROILabel=1:num_ROIs
    indices = intersect(thisROIlist(ROILabel).maski,savei);
    itemMap = map(indices);
    %thisROILabel=horzcat(viewParams.ROILabels{ROILabel},'  ');
%%%%%added 03.05.2005
    %%%%%%above could be a big problem for ROI labeling menus
    thisROILabel=viewParams.ROILabels{ROILabel};
    itemStr= {thisROILabel length(itemMap)};  %26.01.05 references
global variable
    if( length(itemMap)>0 )
      itemStr = horzcat(itemStr, {int32(mean(itemMap))
int32(std(itemMap))});
      %changed to int 17.05.2005
    else
        itemStr = horzcat(itemStr, {0 0});
    end;  %if
    %09.02.2005 error here?  solved by adding else clause immediately
    %above?
    %newStr  %debug 09.02.2005
    %itemStr  %debug 09.02.2005
    newStr = vertcat( newStr, itemStr );
  end;  %for

   %added 06.01.05 to make sure that array displays properly in listbox
% Ensure that all elements of cell array are strings for display
```

```
for i=1:numel(newStr) % Goes through all elements
if ~iscellstr(newStr(i))
newStr{i}=num2str(newStr{i});
end  %if
charLength(i)=length(char(newStr{i}));
%newStr{i}=horzcat(newStr{i},'  ');  %%%%%%%added 03.05.2005
end  %for
% Insert spaces to separate the strings for display purposes
maxCharLength=max(charLength);
if maxCharLength>8   %26.01.05 labels were getting to be 30 long
    maxCharLength=8;
end
maxCharLength_withBuffer=maxCharLength+2;   %%%%%%%%%added 05.05.2005
for i=1:numel(newStr)
   if length(char(newStr(i))) > maxCharLength_withBuffer   %added to
deal with labels to long 27.01.05
        myEntry=char(newStr(i));
        len=length(myEntry);
        myEntry=myEntry(len-maxCharLength_withBuffer+1:len);  %use last
N characters, because space padding is at the beginning.
        newStr{i}=myEntry;
    end %if
    while length(char(newStr(i))) < maxCharLength_withBuffer
newStr(i)=strcat({' '},newStr(i));
    end  %while
end   %for
viewParams.ROILabels;
[h w]=size(newStr);
for i=1:h
viewParams.ROILabels{1,i}=newStr{i,1};  %updated ROILabels when
displayMap is updated
end
viewParams.ROILabels;
%newStr=strcat(newStr,{'  '});  27.01.05 commented out, causing
problems
% Concatenate the cell array of strings
newStr=cell2mat(newStr);

  set( list_ROIlist, 'String', newStr );

  tempMapi = intersect(selectedMapi, savei);
  %for debug  30.12.2004
  sizeTempMapi=size(tempMapi);


  map(setdiff(theMapi,tempMapi)) = 0;
  %sourceParams.msize  %%debug 23.05.2005
  tempMap = reshape( map, sourceParams.msize );


    %is tempMap empty??? 30.12.2004
    sizeTempMap2=size(tempMap);
    maxTempMap=max(max(tempMap));
    %h=sizeTempMap(1);
    %w=sizeTempMap(2);
    %tempMap(h/2:h/2+10,w/2:w/2+10)   %for debug purposes
    %disp('in update_displayMap--end')
```

```
  set( fhm, 'Pointer', 'arrow' );



function update_tempMap( fh )
%UPDATE_TEMPMAP

% Creates temporary T1 maps for displaying and analysis.  The temporary
% mask is a masked version of the original.

% fh - figure handle for ui_viewResults and contains fields for
selecting what
%   parts to mask.


  include_globals;
  global selectedMapi tempMapi tempMap thisROIlist textLabels;

  % get handles to UI controls
  obj_list = allchild(fh);
  list_ROIlist = findobj( obj_list, 'Tag', 'list_ROIlist' );

  % ------------------------------------------------------------------
  % identify selected ROIs
  % ------------------------------------------------------------------
  % tempMap and tempMapi is a union of the selected ROIs (with
  % out-of-range pixels removed or interpolated)
  selectedROIs = get( list_ROIlist, 'Value' );
  selectedMapi = unionall( thisROIlist(selectedROIs).maski, [] );

  % ------------------------------------------------------------------
  % process all ROIs
  % ------------------------------------------------------------------

  map = theMapSave(:);
  savei = theMapi;
  interpolatei = [];

  % bad-fit pixels
  errorParam = feval( ['def_', dataParams.type], 'errorParam');
  errorMap = pfittedSave(:,errorParam);
  i = find(errorMap<=viewParams.maxErrorVal);
  run_filterPixels( 'pixelCount', 'Pmin', ...
            length(setdiff(selectedMapi, theMapi(i))));
  if( strcmpi(viewParams.maxErrorValAction,'remove') )
    savei = savei(i);
  elseif( strcmpi(viewParams.maxErrorValAction,'interpolate' ))
    interpolatei = union( interpolatei, setdiff(theMapi,theMapi(i)) );
  end;

  % below T1 minimum pixels
  i = find(map>=viewParams.minTval);
  run_filterPixels( 'pixelCount', 'Tmin', ...
            length( setdiff(selectedMapi, i)) );
```

```
  if( strcmpi(viewParams.minTvalAction,'remove') )
    savei = intersect(savei, i );
  elseif( strcmpi(viewParams.minTvalAction,'interpolate' ))
    interpolatei = union( interpolatei, setdiff(theMapi,i) );
  end;

  % above T1 maximum pixels
  i = find(map<=viewParams.maxTval);
  run_filterPixels( 'pixelCount', 'Tmax', ...
            length( setdiff(selectedMapi, i)) );
  if( strcmpi(viewParams.maxTvalAction,'remove') )
    savei = intersect(savei, i);
  elseif( strcmpi(viewParams.maxTvalAction,'interpolate' ))
    interpolatei = union( interpolatei, setdiff(theMapi,i) );
  end;

  % interpolate "bad" pixels
  if( ~isempty(interpolatei) )
    map(interpolatei) = 0;  % zero the bad pixels first
    X = reshape( map, sourceParams.msize );
    h = intFilter( viewParams.interpolationRadius );    % interpolation
filter
    A = conv2( X, h, 'same' );              % do it
    A = A(:);
    map(interpolatei) = A(interpolatei);        % updated bad pixels
  end;

  % update listbox showing list of ROIs
  oldStr = get( list_ROIlist, 'String' );
  num_ROIs = size(oldStr, 1);
  all_ROIs = 1:num_ROIs;
  newStr = '';
  for i=1:num_ROIs
    indices = intersect(thisROIlist(i).maski,savei);
    itemMap = map(indices);
    itemStr = ['ROI ' num2str(i) ' -- ' ...
           num2str(length(itemMap)) ' pixels' ];
    if( length(itemMap)>0 )
      itemStr = [itemStr ...
         '   mean: ' num2str(int32(mean(itemMap))) ...
         '   std dev: ' num2str(int32(std(itemMap)))];  %changed to int
17.05.2005
    end;
    newStr = strvcat( newStr, itemStr );
  end;
  set( list_ROIlist, 'String', newStr );

  tempMapi = intersect(selectedMapi, savei);
  %for debug  30.12.2004
  sizeTempMapi=size(tempMapi);

  map(setdiff(theMapi,tempMapi)) = 0;
  %sourceParams.msize  %%debug 23.05.2005
  tempMap = reshape( map, sourceParams.msize );
    %is tempMap empty??? 30.12.2004
    sizeTempMap_c=size(tempMap);
    maxTempMap=max(max(tempMap));
```

```
  %h=sizeTempMap(1);
  %w=sizeTempMap(2);
  %tempMap(h/2:h/2+10,w/2:w/2+10)    %for debug purposes

disp('in update_tempMap')
```