

FMRLAB

Quick-Start Tutorial

Version 2.0
September 10, 2002

© Jeng-Ren Duann & Scott Makeig, 2002
Swartz Center for Computational Neuroscience
Institute for Neural Computation
University of California San Diego

Information and downloads: <http://sccn.ucsd.edu/fmrlab>
Questions and feedback: fmrlab@sccn.ucsd.edu.

FMRLAB Quick-Start Tutorial

Introduction

1. FMRLAB Installation

- 1.1 Download FMRLAB
- 1.2 Unzip and install FMRLAB
- 1.3 Add the FMRLAB path to the Matlab environment
- 1.4 Edit the FMRLAB settings file 'icadeefs.m' to set ICA defaults
- 1.5 Download the FMRLAB example data set

2. Functional Image Preprocessing and ICA Decomposition

- 2.1 Start FMRLAB
- 2.2 Quitting FMRLAB
- 2.3 Create an FMRLAB dataset
- 2.4 Save the FMRLAB dataset
- 2.5 Remove initial 'dummy' scans
- 2.6 Perform slice timing adjustment
- 2.7 Remove off-brain voxels
- 2.8 Decompose the data with ICA

3. Visualizing Results of ICA Decomposition

- 3.1 Visualize component regions of activity (ROAs)
- 3.2 Visualize component maps on structural images
- 3.3 Find dominant components by maximum Z value
- 3.4 Find dominant components by PVAF
- 3.5 Export a selected component
- 3.6 Spatially normalize the ROA maps
- 3.7 Produce a maximal intensity projection (MIP) display
- 3.8 Produce a 2-D slice-overlay display
- 3.9 Produce a 3-D dead-model rendered display

Appendix – Function List of FMRLAB

A.1 Main Files

A.2 Functions from ICA Toolbox

A.3 Functions from ICA Toolbox

A.4 Function from Supplement of SPM99 Toolbox

Introduction

This document provides step-by-step guidelines for those who want to use FMRLAB, a Matlab toolbox for fMRI data analysis using independent component analysis (ICA) available under the GNU public license from <http://sccn.ucsd.edu/fmrlab/>. This document first describes the procedures for installing the toolbox and then illustrates the procedure for using FMRLAB to analyze fMRI time series using a hands-on example. The example dataset used in this tutorial is also available at <http://www.sccn.ucsd.edu/fmrlab/example/>.

Theory and Background. For background information about ICA and its application to fMRI data analysis, please refer to the references available at <http://www.sccn.ucsd.edu/fmrlab/>. FMRLAB has a counterpart, EEGLAB, for analyzing EEG or MEG data using ICA. It also can be freely downloaded from <http://www.sccn.ucsd.edu/eeqlab/>. Some of the visualization functions FMRLAB uses to display ICA results are adapted from functions contained in SPM99, a Matlab-based program for brain imaging visualization and analysis which can be downloaded from <http://www.fil.ion.ucl.ac.uk/spm/>.

Requirements. FMRLAB runs under core MATLAB (The Math Works, Inc., Natick, MA), version 5.3 or higher. Currently, the (faster) binary version of our infomax ICA routine 'runica()' (run from within Matlab using 'binica()') has only been compiled under Linux (and FreeBSD). On the visualization side, the SPM99 display functions (e.g., max intensity projection or MIP, 2-D slice overlay, and 3-D rendering) run only under Linux. For other platforms, please refer to SPM99 website and download the proper version of related functions (see list in Appendix). FMRLAB requires at least 256 MB of RAM (more is better) and a Pentium III (or IV) CPU.

Processing Time. The amount of processing required by FMRLAB is relatively modest. For example, applied to a conventional fMRI dataset, FMRLAB requires less than 10 minutes to preprocess and run ICA training using a laptop running Linux with 1.6 GHz Pentium IV CPU and 1GB memory.

Image formats. The image format used in FMRLAB is generic raw (.img) without header and footer. Thus, the experimenter needs to enter the image acquisition parameters (e.g. image height and width, number of slices, FOV, slice thickness, TR, etc) as well as the experimental paradigm (e.g., total number of scans, stimulation onset asynchrony (SOA), etc). FMRLAB provides an editor allowing users to enter this information. To convert fMR images to the FMRLAB format, the FMRLAB distribution includes some MATLAB routines to convert images from different systems (Siemens Symphony, Siemens Magnetom, GE Signa 1.5/2.0 T and Bruker MedSpec S300 3T). Please refer to <http://www.sccn.ucsd.edu/fmrlab/> for further details. To display the results of regions of activity

(ROAs) using MIP, 2-D slice overlay, or 3-D rendering, the data need to be converted to ANALYZE format (Biomedical Imaging Resource, Mayo Foundation) used by SPM FMRLAB is equipped with a build-in converter function for this purpose.

FMRLAB Manual. This document gives a quick-start to FMRLAB only. It gives step-by-step instructions as to how to install the toolbox and get some hands-on experience with its use. Some available FMRLAB functions are not covered in this tutorial. For the details of these and other FMRLAB functions, please refer to the FMRLAB manual, which can also be downloaded from <http://sccn.ucsd.edu/fmrlab/>.

1. FMRLAB Installation

1.1 Download FMRLAB

The FMRLAB toolbox for fMRI data analysis using ICA can be downloaded at: <http://www.sccn.ucsd.edu/fmrlab/> as a file named **fmrlab.tgz**. Under Microsoft Explorer, click the right mouse button and select “Save link as” Under Netscape, press SHIFT + left mouse button to download the toolbox .tgz file and save it to disk.

1.2 Unzip and install FMRLAB

Copy the FMRLAB .tgz file into an FMRLAB directory, for example, “/home/ourlab/matlab/fmrlab”. Use “**tar xvzf fmrlab.tgz**” to unzip and untar the file. This will save all the necessary files for running FMRLAB in the fmrlab directory.

1.3 Add the FMRLAB path to the Matlab environment

Open the file **startup.m** using a text editor. Add the line “**path(path, FMRLAB_DIR);**” to the end of file. Replace **FMRLAB_DIR** here with the actual pathname of the FMRLAB directory (in our example, ‘/home/ourlab/matlab/fmrlab’)

1.4 Edit the FMRLAB settings file, ‘icadefs.m,’ to set ICA defaults

Open the file **icadefs.m** using a text editor. On **line 8**, replace **ICADIR** by your FMRLAB directory path (for example ‘/home/lab/matlab/fmrlab’). On **line 17**, set **ICABINARY** to ‘**FMRLAB_DIR/ica_linux**’, using the pathname of your **FMRLAB_DIR** directory (in our example,

'/home/ourlab/matlab/fmrlab/ica_linux').

1.5 Download the FMRLAB example data set

The example data set used in this tutorial can be downloaded from: <http://www.sccn.ucsd.edu/fmrlab/example/>. This data set contains two files, **2dseq_r1** and **2dseq_str**. The first is the file of functional images. The second contains the corresponding structural images. The functional images were acquired during a 5-minute experiment in which the subject was shown brief 8-Hz flickering-checkerboard stimulation lasting 0.5 s every 30 seconds (SOA). (See [Duann et al., 2002](#) for details).

The image acquisition parameters for the functional images were:

- ?? Image dimensions = 64 x 64 x 5
- ?? FOV = 250 mm x 250 mm
- ?? Slice thickness = 7 mm
- ?? TR = 0.5 sec
- ?? Total number of scans = 610 (600 time points)
- ?? Dummy scans = 10

The structural scans were T1-weighted images with the same slice positions, number of slices and FOV as the functional scans. However, they were acquired at 256 x 256 resolution to provide more structural details than the functional scans.

For this data set, the structural image acquisition parameters were:

- ?? Image dimensions = 256 x 256 x 5
- ?? FOV = 250 mm x 250 mm
- ?? Slice thickness = 7 mm

2. Functional Image Preprocessing and ICA Decomposition

Next we demonstrate, step-by-step, how to use FMRLAB to analyze the example fMRI data set.

2.1 Start FMRLAB

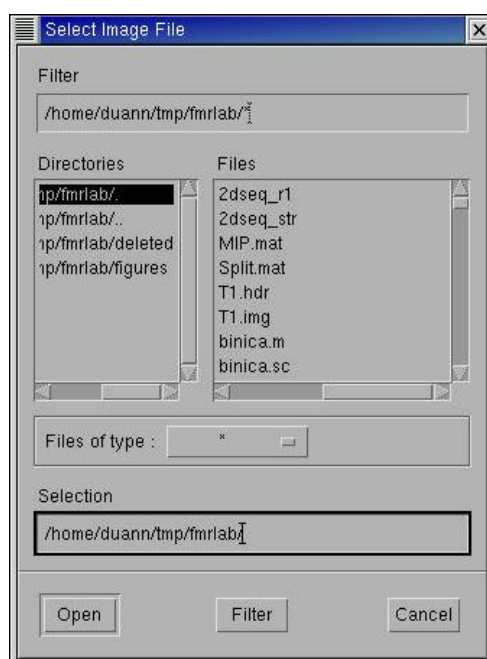
From the MATLAB command line, type **fmrlab**. If the environmental variables have been set properly, we recommend starting FMRLAB from the directory where the example image data are located.

2.2 Quitting FMRLAB

To exit from FMRLAB, select the FMRLAB menu item “**Dataset > Quit**”. This will clean the workspace, close any figures created by FMRLAB, and clear all the FMRLAB variables, including the FMRLAB global variable structure, **FMRI**.

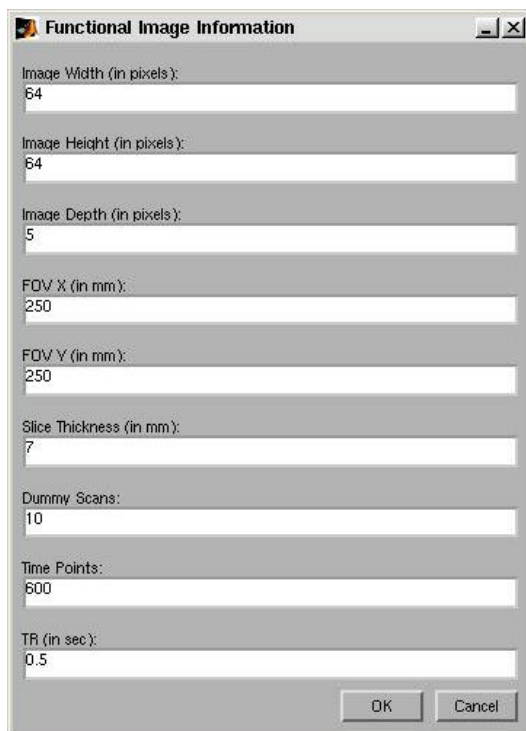
2.3 Create an FMRLAB dataset

The procedure for creating an FMRLAB dataset is: (1) Select a functional image file. (2) Enter the image acquisition parameters for the functional scans. (3) Enter the image information for the structural scans. Select **Dataset > Create Dataset** from the FMRLAB menu. A **Select Image File** window will pop up (as below) allowing you to select the file containing the functional scans.



Move the mouse cursor to **2dseq_r1** (the functional scan file for the sample data set) and click the **Open** button at the bottom of the figure. Next, a **Functional Image Information** window will pop up (as below) allowing you to enter the image acquisition parameters for the functional scans. Fill all the fields with the values given above (see Section 1.5).

After filling in the correct values in all the fields, press the **OK** button at the bottom of the window. Next, the **Structural Image Information** window will pop up (as below) to allow you to enter the necessary structural image parameters.



Functional Image Information

Image Width (in pixels): 64

Image Height (in pixels): 64

Image Depth (in pixels): 5

FOV X (in mm): 250

FOV Y (in mm): 250

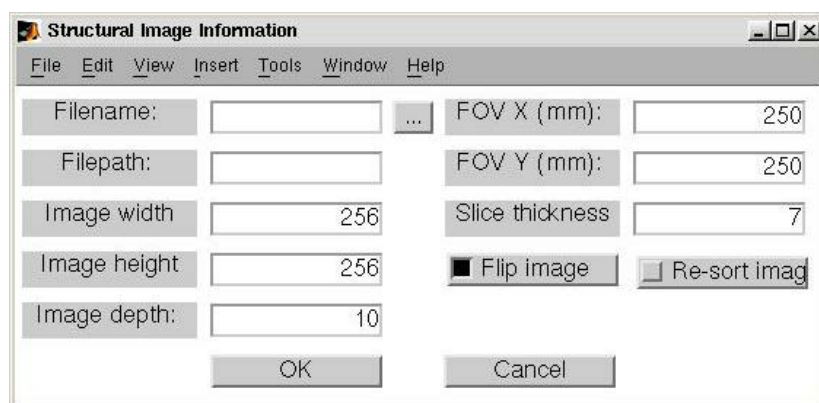
Slice Thickness (in mm): 7

Dummy Scans: 10

Time Points: 600

TR (in sec): 0.5

OK Cancel



Structural Image Information

File Edit View Insert Tools Window Help

Filename: [] ... FOV X (mm): 250

Filepath: [] FOV Y (mm): 250

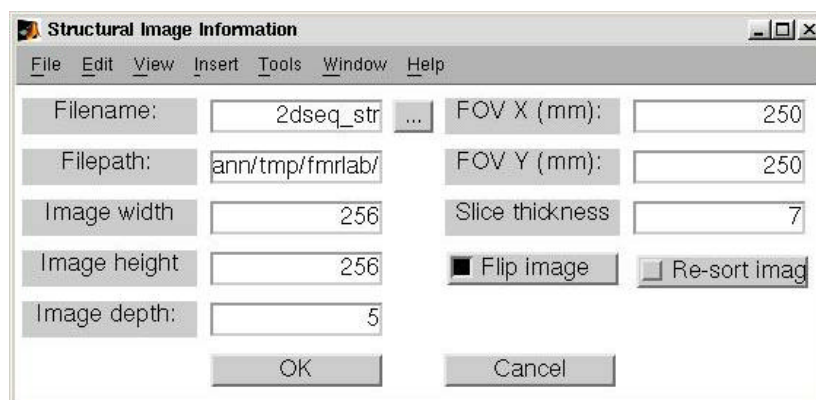
Image width: 256 Slice thickness: 7

Image height: 256 ☒ Flip image ☐ Re-sort image

Image depth: 10

OK Cancel

Press the (top center) continuation button [...], a **File Selection** dialog will be brought up for user to select file from the list, then it will fill in the filename as well as the pathname automatically. Refer to the structural image acquisition parameters (Section 1.5 above) and enter the correct values in the other fields of this window. Keep the **Flip Image** checkbox checked. This will flip the images to the neurology standard (“right is right”). If the sequence of structural images begins with the slice closest to the top of the head, check the **Re-sort Image** checkbox. This will convert the images to begin with bottom slice (as per the requirements of SPM99) for spatial normalization and visualization. The completed window should look like that shown below.



2.4 Save the FMRLAB dataset

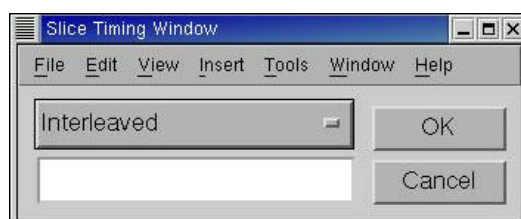
To save the dataset you have just created, select **Dataset > Save Dataset** from the FMRLAB menu. A file selection window will pop up allowing you to input the output filename. Be sure to append the extension **.fmr** to the output dataset file. Then, click the “**Save**” button to save the dataset to disk.

2.5 Remove initial ‘dummy’ scans

The number of dummy scans was specified during the create-dataset procedure (see 2.3 above). To remove those scans from the fMRI time series, select **Process > Remove Dummies** from the menu.

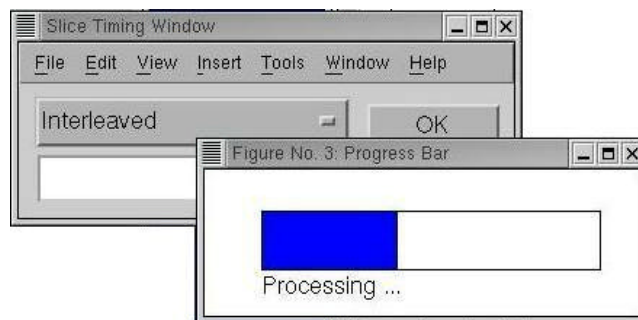
2.6 Perform slice timing adjustment

To adjust the slice timing (using interpolation to make the acquisition times for the slices as synchronous as possible), select **Process > Slice Timing**. A **Slice Timing Window** will pop up (as below) allowing you to specify the sequence in which the functional slices were acquired. There are four possible selections: **Interleaved**, **Ascending**, **Descending** and **User Defined**. The default is **Interleaved**.



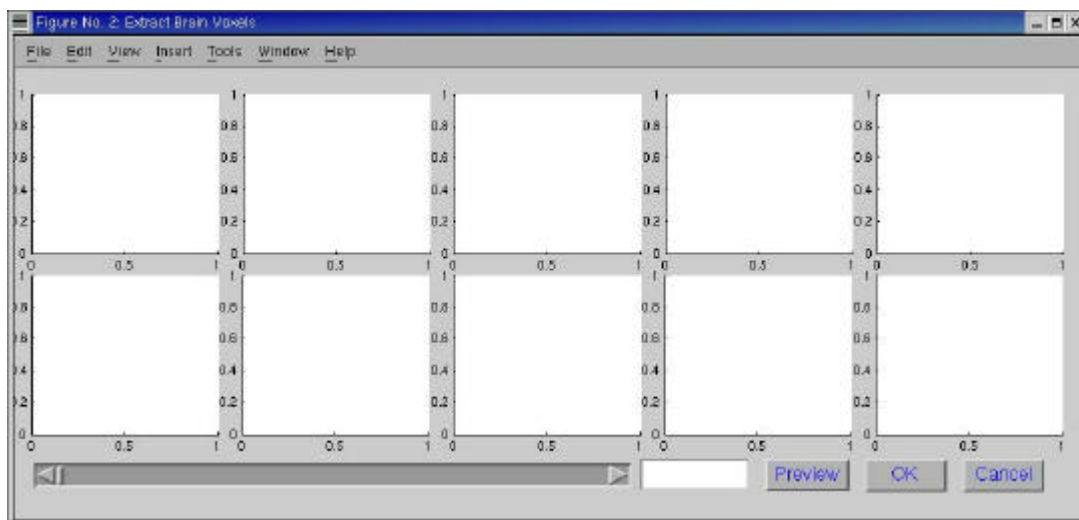
To use **User Defined** mode, enter the actual slice sequence, for example, “2 4 6 8 10 1 3 5 7 9,” into the text entry field of the **Slice Timing Window**. Then press **OK** to start the slice timing adjustment.

process. During this process, a progress-bar window will pop up to indicate its progress. When the timing adjustment process is done, FMRLAB will close both the **Slice Timing Window** and the progress bar.

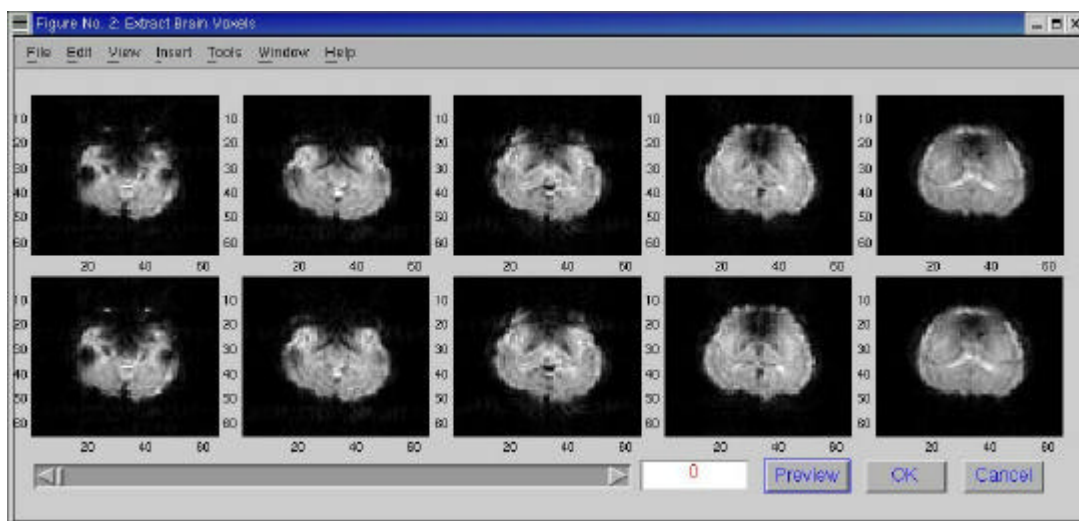


2.7 Remove Off-Brain Voxels

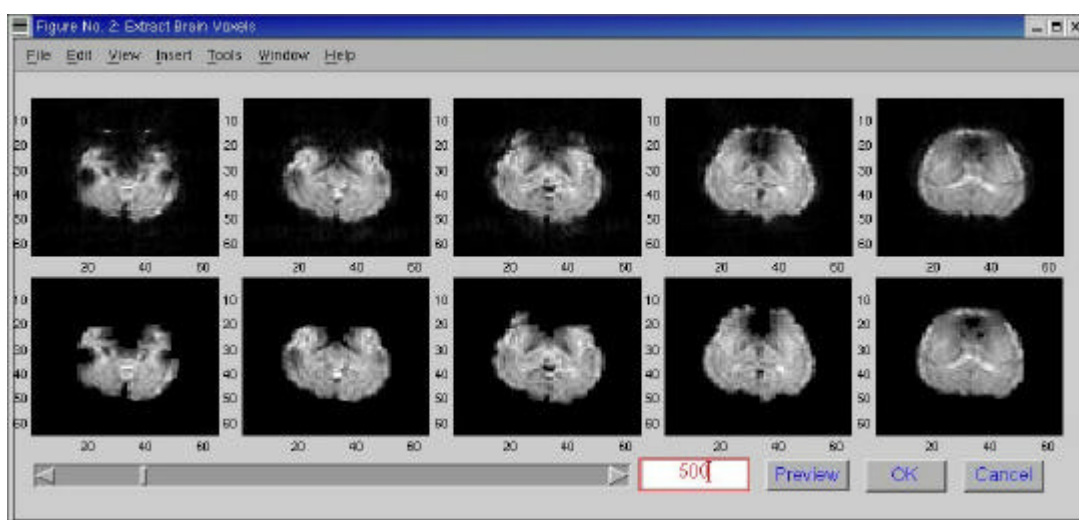
To remove off-brain voxels from the ICA training data, select **Process > Extract Brain** from the menu. The **Extract Brain Voxels** window will pop up. Before you start the voxel removal process, press the **Preview** button (bottom right) to load the functional images.



The images will be displayed in the top row of the window. The bottom row shows the results of using the (not-yet specified) voxel intensity threshold. The figure below shows the **Extract Brain Voxels** window after the functional images are loaded. The whole functional images are displayed in the first row. Because the threshold (appearing in the text entry field at the bottom of this window) is **0**, the thresholded images shown in the bottom row are identical to the whole (top) images.



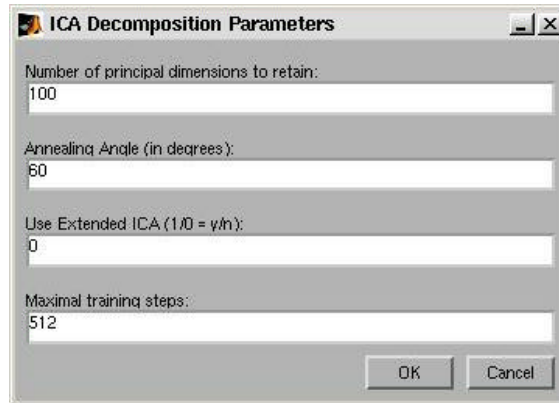
Set the voxel-removal intensity threshold by simply keying in a value, say, **500**, in the text entry field, or use the slider bar to the left of the editing field. To evaluate the results of the voxel selection threshold, compare the images in bottom and top rows. Once you have selected a suitable threshold, press **OK** to accept it and to remove the rejected voxels from the functional images.



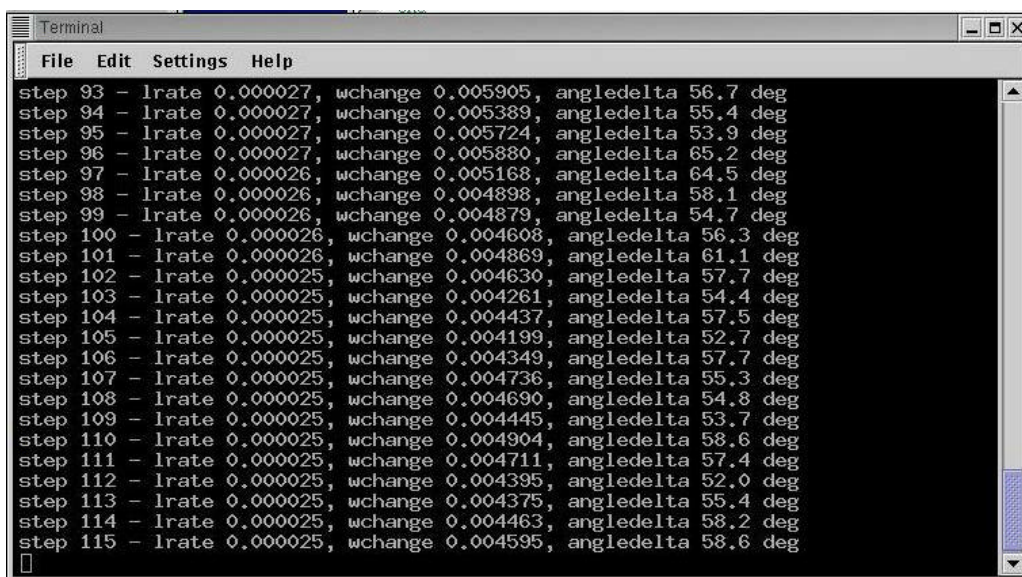
2.8 Decompose the data with ICA

Select **Process > ICA Training** in the menu to perform the ICA training. First, FMRLAB will pop up a window allowing you to tune several training parameters (PCA number, annealing angle, extended mode and maximum number of training steps). Enter **100** for **Principal component dimensions to retain** to reduce the dimensionality of the training dataset (here) from 600 to 100. (**Note**: Some papers

have recently appeared in which the number of principal dimensions retained for ICA training was as low as 4! Because of the complexity of brain hemodynamics, this is highly unrealistic and will not use ICA to best advantage. We recommend using as many dimensions as practical, given your RAM limits, etc.). All other training parameters may be left at the displayed default values. Press **OK** when ICA setup is complete.



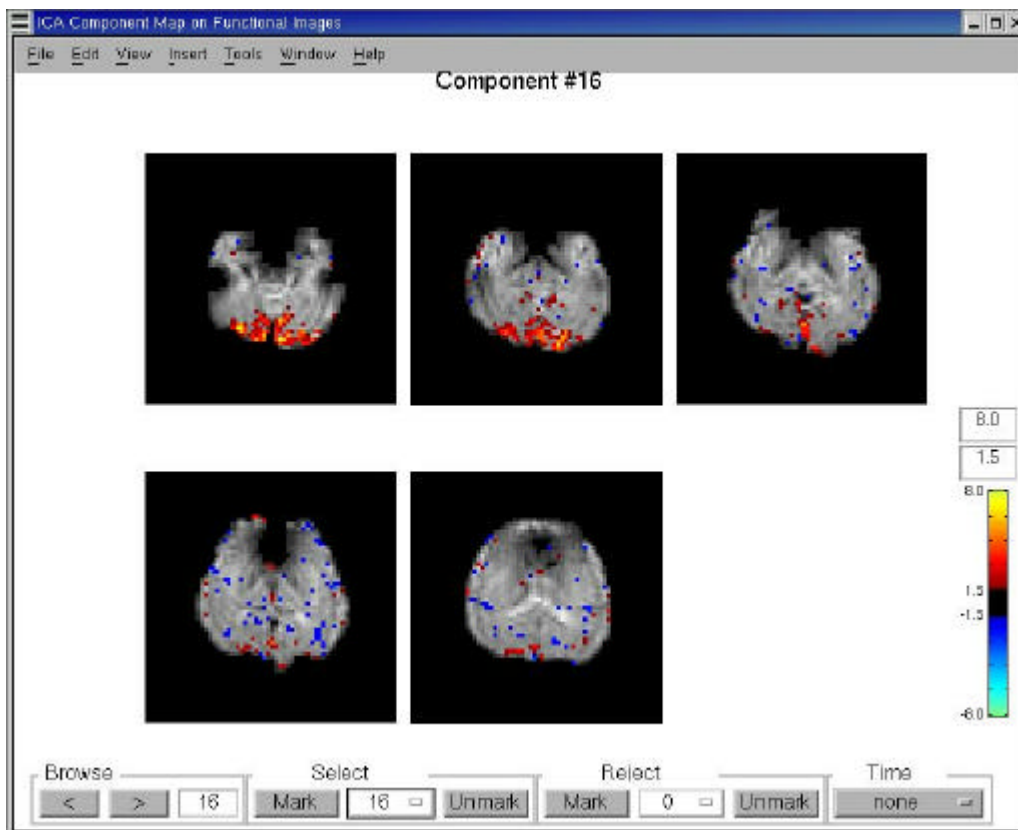
Once the ICA training begins, the MATLAB command line window (as below) will show the ICA training steps as well as the changes in the learning rate, weight values and annealing angle (this is the angle of deviation from the previous weight update to the most recent. For efficient training, this should be near 90 degrees). The process will converge when the change in weight values is less than the specified stopping point (default: 0.000001). If the process does not converge in this sense, it will be stopped after the specified maximum number of training steps.



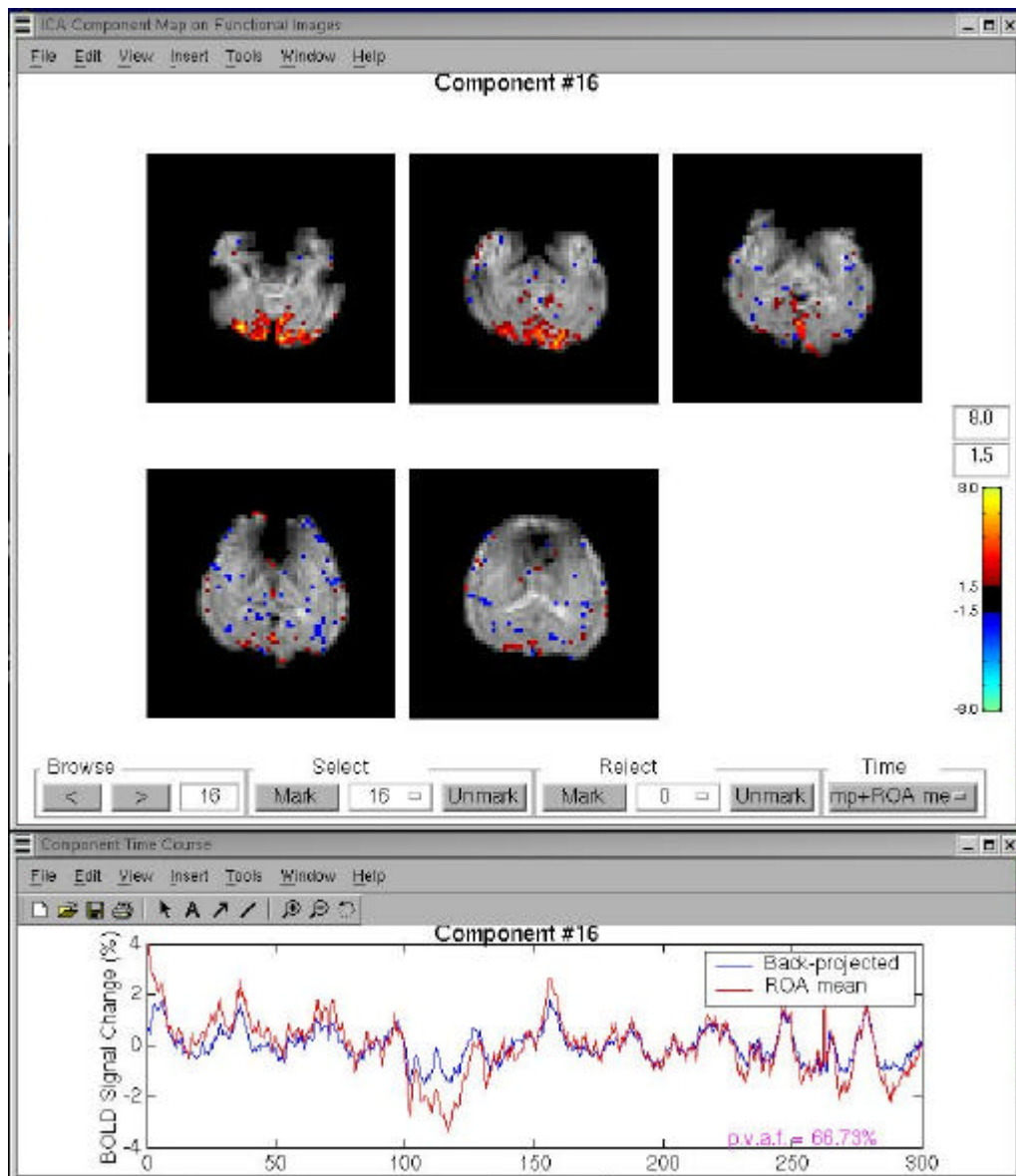
3. Visualizing Results of ICA Decomposition

3.1 Visualize component regions of activity (ROAs)

ICA decomposes the entire BOLD data into maximally spatially independent components. FMRLAB provides a function allowing the user to browse through the component region of activity (ROA) maps to determine which components are of interest for further analysis and which should be rejected as artifacts. Select **Visualize > Component ROAs > On Functional Images** from the FMRLAB menu to open the component map browser. As shown in the figure below, the activation ($\mathbf{u} = \mathbf{W} * \mathbf{x}$ where \mathbf{W} is the ICA unmixing matrix and \mathbf{x} is the (times,voxels) matrix of observed fMRI BOLD signals) is first convert to z values (standard deviations of the voxel weight distribution) and color coded (see the color bar at lower right) to form the colored map we call the component Region of Activity (ROA). Above the color bar, there are two editing fields that allow you to enter z-value thresholds The upper threshold gives the upper bound of the color coding (avoiding the skewing influence of extreme weight values). The lower threshold is the lower color bound of the displayed ROA. Setting this to 0 will fill the entire functional images with (mostly non-significant near-0) color values.



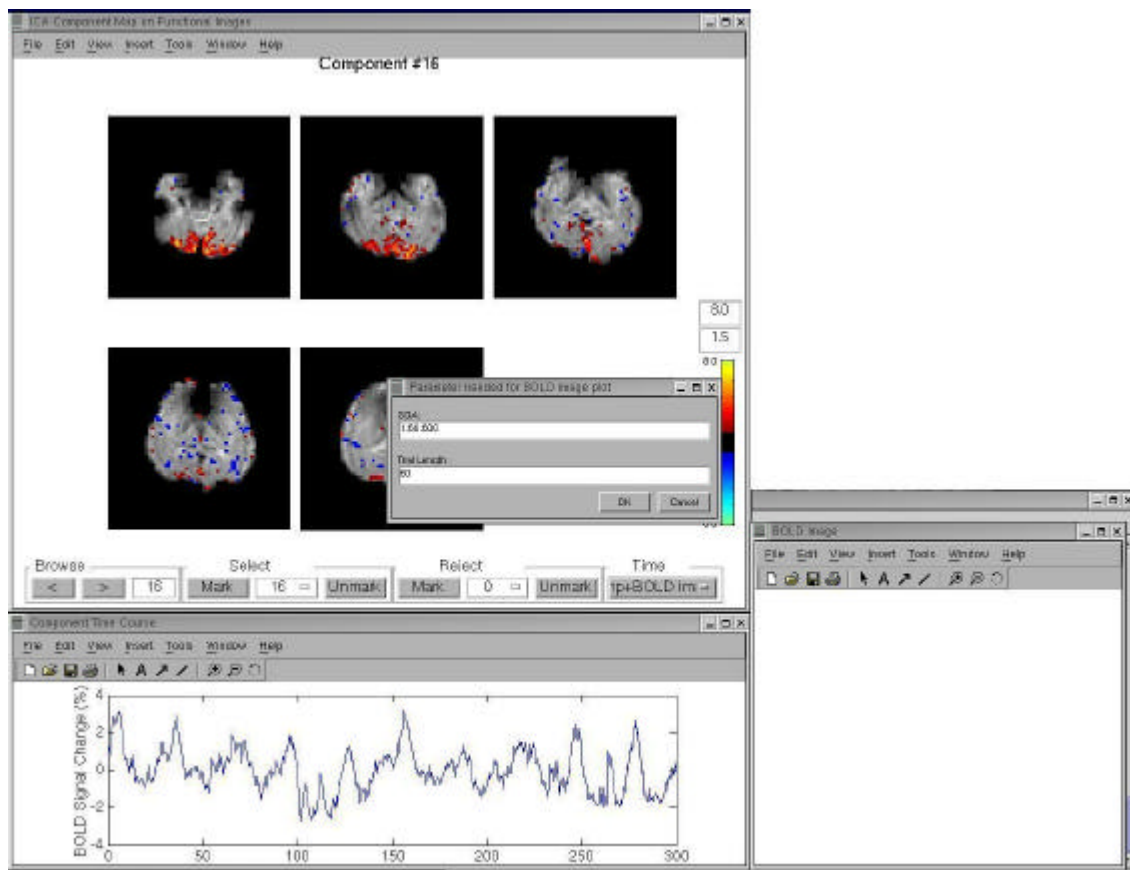
At the bottom of the **Component Map Browser** there are buttons to specify the next component number to review, to mark or unmark the current component either for selection (as a component of interest) or rejection (as a BOLD artifact), plus several time course display options (lower right).



To increase or decrease the component number by 1, simply click mouse button on the arrow [$<$] or [$>$] buttons. To move to a specific component number, type the component number in the text entry field to the right of the [$>$] button and press keyboard **ENTER**. Use the **Mark** and **Unmark** buttons to select components for further analysis or for rejection. To remove a component from the **Select** list, first browse to the component by selecting the component from the **Select** list (by clicking and holding

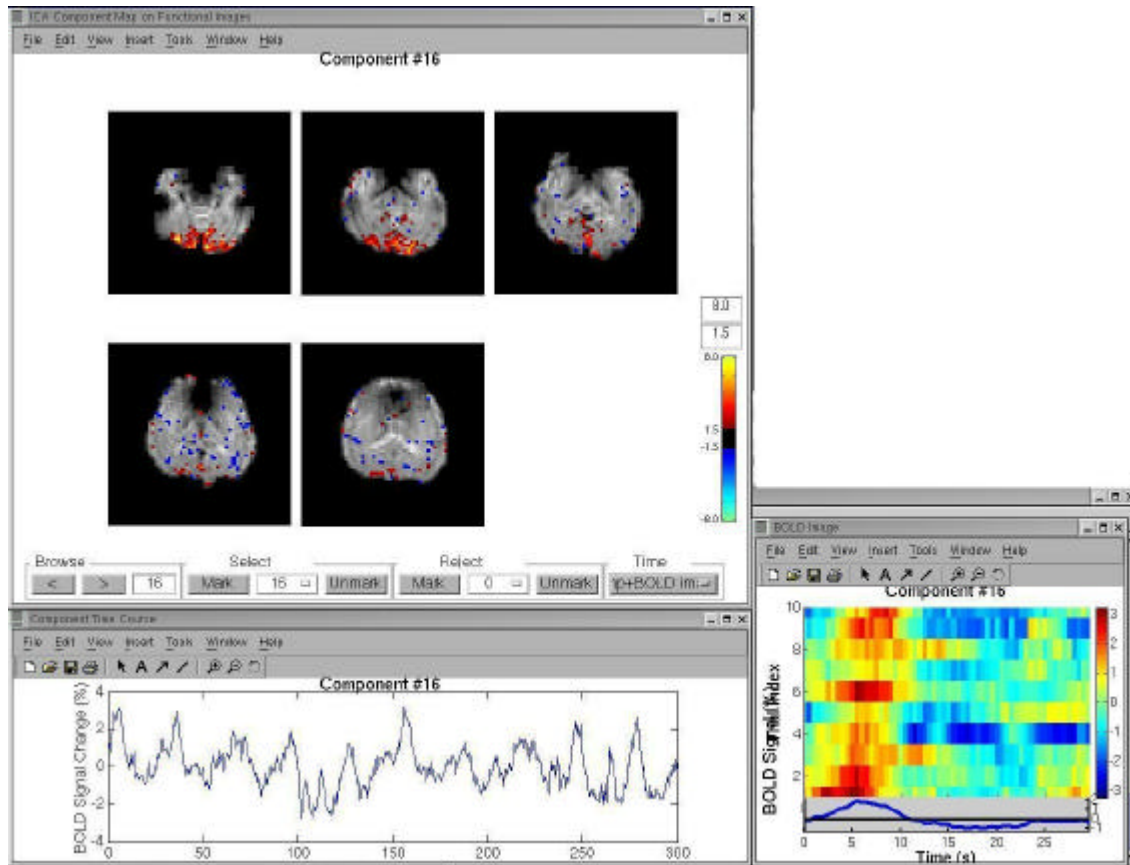
on the pull-down list box), then press the **Unmark** button. Follow the same procedure for unmarking a component marked for rejection.

The **Component Map Browser** can display component time courses in different ways (e.g., as back-projected component time courses, as ROA raw-mean time courses, and as BOLD images). For details, please refer to the FMRLAB manual (see <http://sccn.ucsd.edu/fmrlab/>). The figure above displays both the back-projected component time course (in blue) and the ROA whole-signal mean time course (in red) for component 16 from the decomposition of example fMRI dataset. To compare these two time courses, the FMRLAB graphic also gives the percentage of the whole-signal mean time course variance accounted for (p.v.a.f) by the back-projected component time course (66.73%).



The FMRLAB BOLD-image display function, **boldimage()**, is adapted from the **erpimage()** function of EEGLAB (Arnaud Delorme, Tzyy-Ping Jung and Scott Makeig, see <http://www.sccn.ucsd.edu/eeqlab/>). The **boldimage()** function converts the (1-D) time course associated with an fMRI ICA component to a 2-D image by segmenting the time course into experimental trials or epochs, then color coding the BOLD epoch time course, finally stacking the

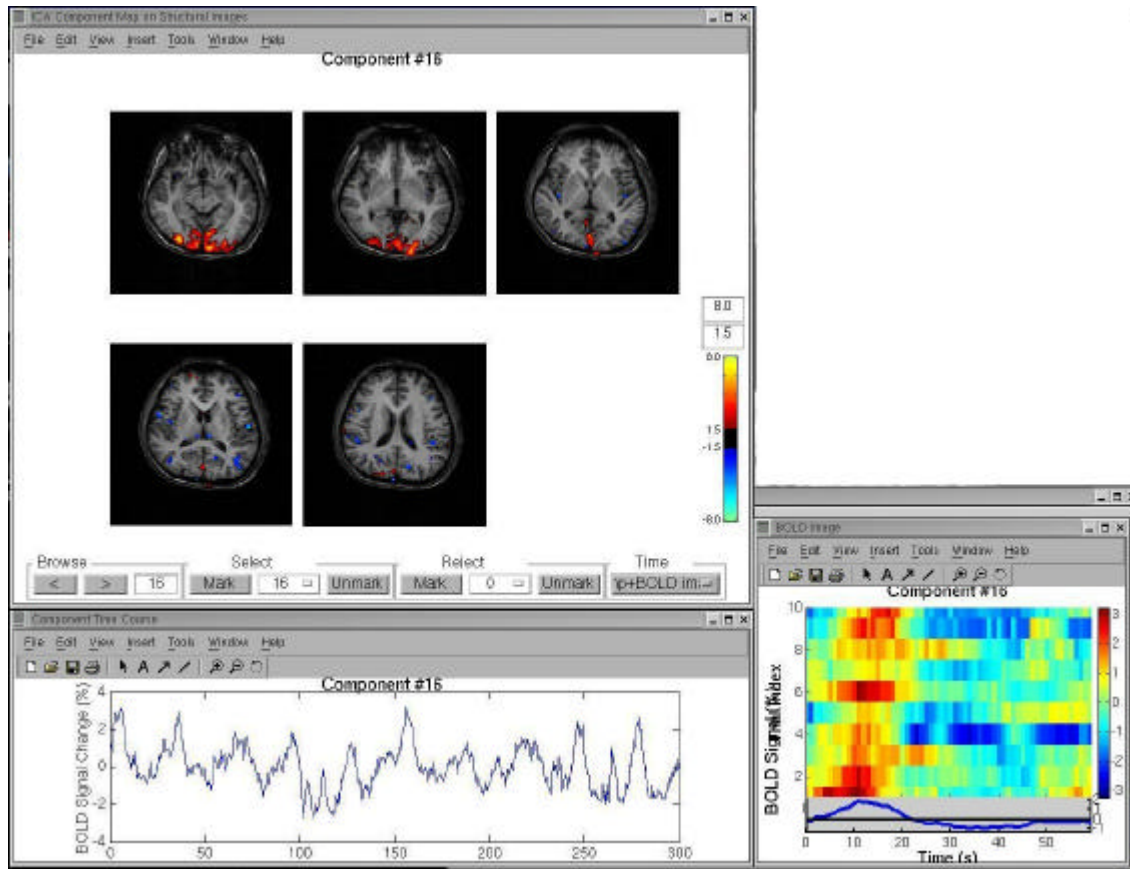
color-coded epoch time courses in acquisition order to form a color-coded 2-D BOLD image. Clicking on the **Time** text selection box and selecting the **Component + BOLD** item, a text entry window asking for the experiment epoch length and SOA will appear (as shown in the figure above). When these values are specified (followed by pressing **OK**), the BOLD image of the component time course will be displayed as the right lower panel below.



3.2 Visualize component maps on structural images

FMRLAB provides a function to overlay component ROA maps on the structural images. If you input structural images, you can also overlay the component ROAs on top of them by selecting **Visualize > Component ROAs > On Structural Images** from the FMRLAB menu. An example is shown in the figure below. The display features are the same as for mapping ROAs on the functional images (3.1 above). Although the user can also use this function to browse the component ROAs, the structural images must be available. Also, the structural maps appear much more slowly than maps on the functional images since the ROAs must be interpolated to fit the image dimensions of the structural scans. To save time, we recommend first mapping component ROAs on functional images, then

creating and saving ROA maps on structural images only for selected components of interest.



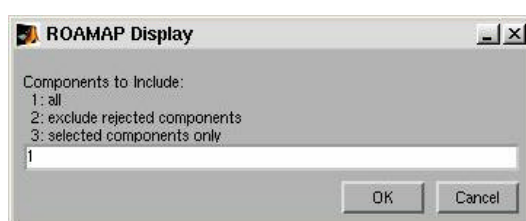
3.3 Find dominant components by maximum z value

Another FRMLAB component search method selects, for every voxel, the component having the maximum (z-value) weight at that voxel. The assigned component participates most strongly in generating the BOLD signal at the specific voxel. This component may also have the highest correlation coefficient between the back-projected component time course and the whole voxel time course, though this need not always be the case. Color-coding the dominant component for each voxel allows the user to graphically select voxel regions dominated by a single component. To bring up this search image, select **Visualize > Dominant Component > by Max Z**.

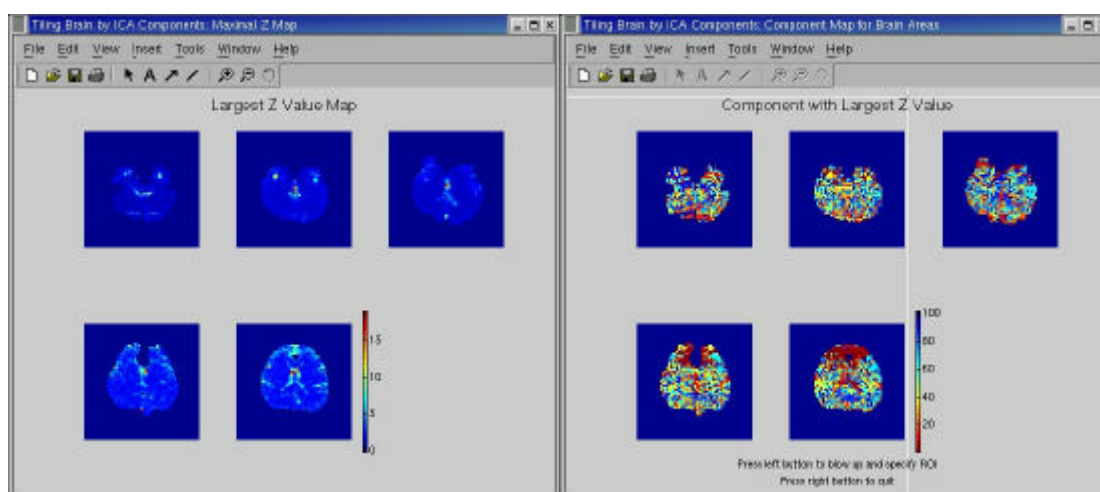
First, a **ROAMAP Display** window will pop up (as below) allowing you to input the lower-bound z value to use in the display. Entering a higher threshold will make the resulting figure simpler (with less “chickenpox” noise).



After the lower-bound threshold is set, a second **ROAMAP Display** window pops up to ask you if you want to show all components (option 1), or just the components selected as of interest in the component browser (option 2). You can also choose (option 3) to show all components except those on the “reject” list.

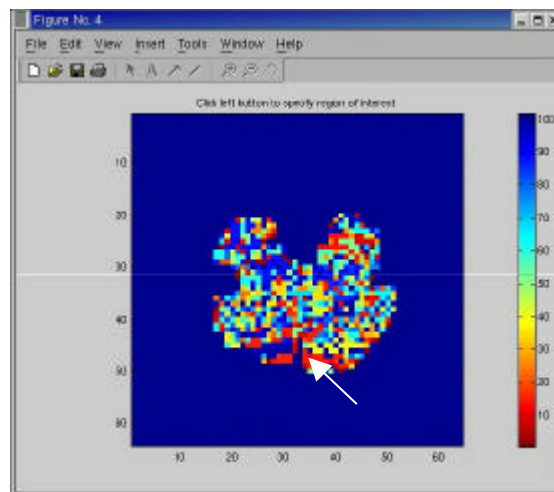


After the proper parameters are assigned, FMRLAB will calculate the z value of each voxel weights according to the activation ($u_{i,j} = W_{i,k} * x_{k,j}$, where i is the component number, j is the voxel index, and k is the time point of the fMRI time series) of each component, and will assign to every voxel a maximal z value (shown in the left panel of the figure below). To every voxel, the function will also assign a component having the maximum z value (as in the right panel below). We may call this component the “defining component” of the voxel.



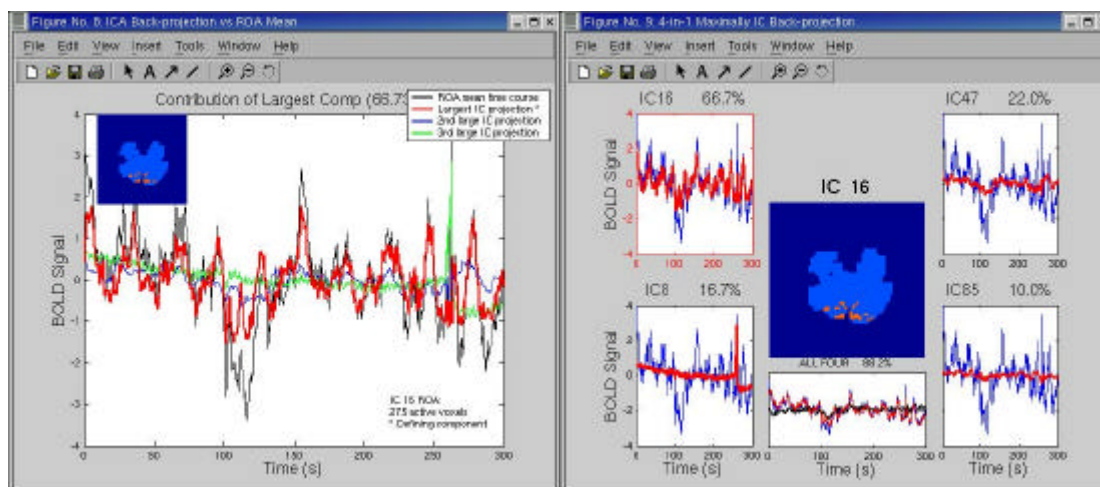
Clicking on one of the thumbnail slice images of this map (on right, above) will pop up an enlarged

slice image in another window (as below), allowing the user to select a voxel of interest. Normally, we click on a voxel in a color-connected region, for example, as shown by the white arrow below) since most relevant hemodynamic processes for cognitive research may be those that affect the BOLD signals of geometrically connected voxel regions rather than of isolated voxels. After the voxel component of interest is selected, the pop-up window will be closed. On the command line FMRLAB will indicate the number of the component selected.



FMRLAB then computes the Region of Activity (ROA) of the component by searching across all voxels and finding those whose z-normalized component weights are higher than the default threshold. The function then determines the mean whole-BOLD-signal time course for the voxels in the ROA. We call this the ROA raw-mean time course. In the left panel of the figure below, the black trace shows the ROA raw-mean time course, and the red trace the back-projected time course of the defining component. The blue and green traces show the back-projected component time courses of the 2nd and 3rd components accounting most strongly for the ROA raw-mean time course variance.

The right panel shows the user the ROA of the specified defining component and the four components that account maximally for the variance of the ROA raw-mean time course. This panel also gives the pvaf for these four components. For example, below the pvaf of the defining component (IC16) is 66.7%. The second-largest pvaf (IC 47) is 22.0%. the third (IC8), 16.7% and the fourth (IC85), 10%. Note that the ROA raw-mean time course pvaf by the sum of these four independent components is 88.2% (shown under the ROA map). **Note:** Since spatially independent components need *not* have orthogonal time courses, the pvaf of the backprojected sum of two or more components is *not* usually the same as the sum of the individual pvaf's of the individual component backprojections.

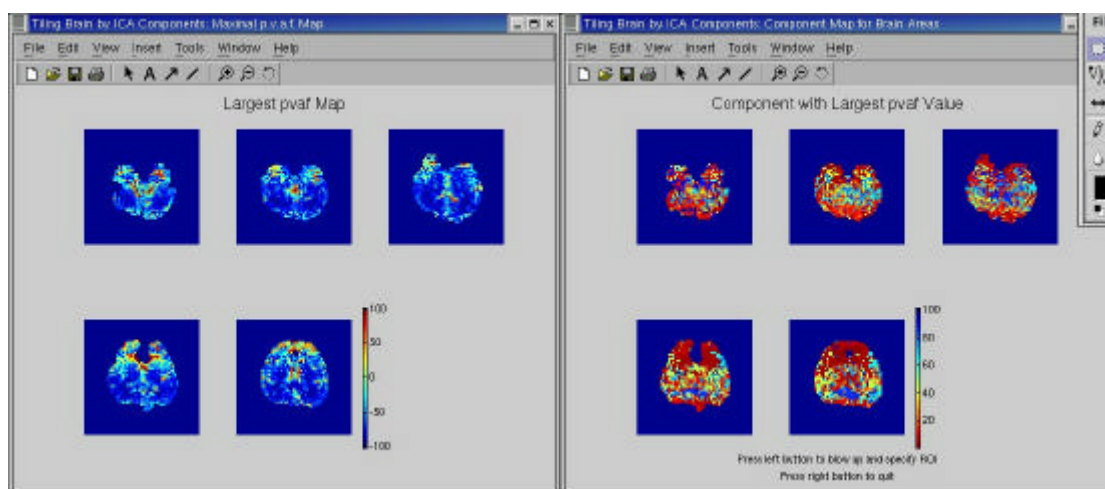


3.4 Find dominant components by PVAf

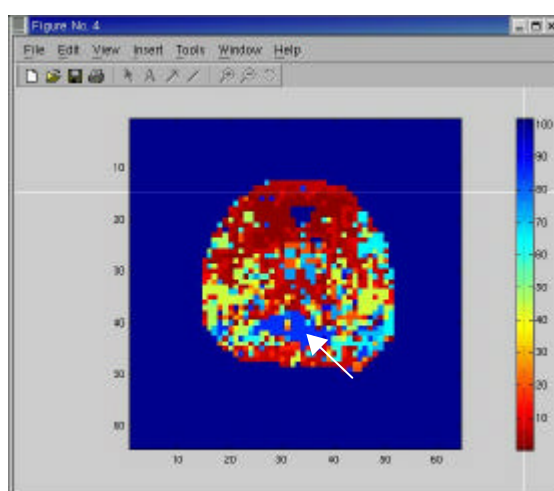
Another way to find dominant components is by ranking components by pvaf by assigning to every pixel the component maximally accounting for the variance of its raw time course. Select **Visualize > Dominant Component > By PVAf** from the menu. The system will first bring up a window allowing you to select the ICA components you want to consider in constructing the map. There are three options: Entering **1** will use all components in the analysis. If **2**, only the browser-selected components will be considered. Entering **3** will cause the function to consider all components except those on the component-browser “Reject” list.



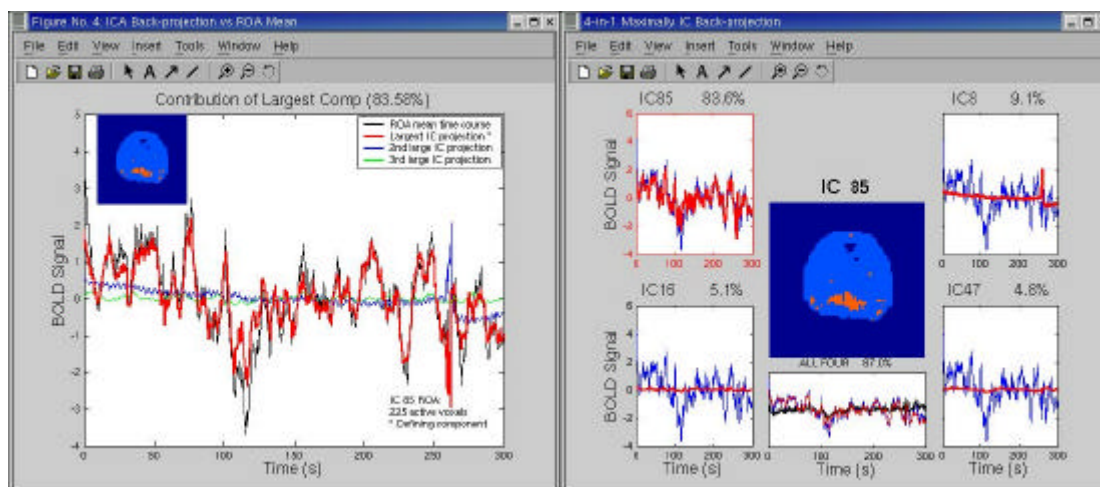
The two figures below show a maximal pvaf map (left) and a maximal z-value map (right). In the z-value map color shows the indices of the most highly weighted components. Here, dark red corresponds to the first (and largest) component, dark blue to the 100th (and smallest) component. These maps show some patches of connected voxels that are dominated by the same ICA component. These are often areas (and components) of functional interest.



Clicking on a thumbnail slice image in the maximal-component map (on the right above) will pop up the map of the selected slice map in a new window (as below), allowing the user to select voxel component of interest. In the figure below, user moves the cursor and clicks the left mouse button on a voxel of interest (at the tip of the white arrow).



When a voxel of interest is selected, FMRLAB will plot the region of activity (ROA) of the selected maximal-pvaf component. The function will then derive the raw-mean BOLD-signal time course for voxels in the ROA, and will calculate the pvaf of the mean component time course of the back-projected ROA voxels, producing figures like those below.



3.5 Export selected components

To display results obtained by ICA and compare them with other components from the same or different subjects, FMRLAB must convert the ROA map to standard coordinates and display the result in standard plots (such as 2-D slice-overlay and 3-D brain-rendering). For this purpose, FMRLAB uses some functions from SPM99 (see <http://www.fil.ion.ucl.ac.uk/spm>).

First, FMRLAB needs to export the ROA maps and save them in standard ANALYZE format. The function **Visualize > Export Result** is made for this. Selecting **Visualize > Export Result** will export the ROA maps of components in the selected-component list from the component browser (see Section 3.1 for how to create this list). The Matlab command line window will record the progress of component saving (as below).

```

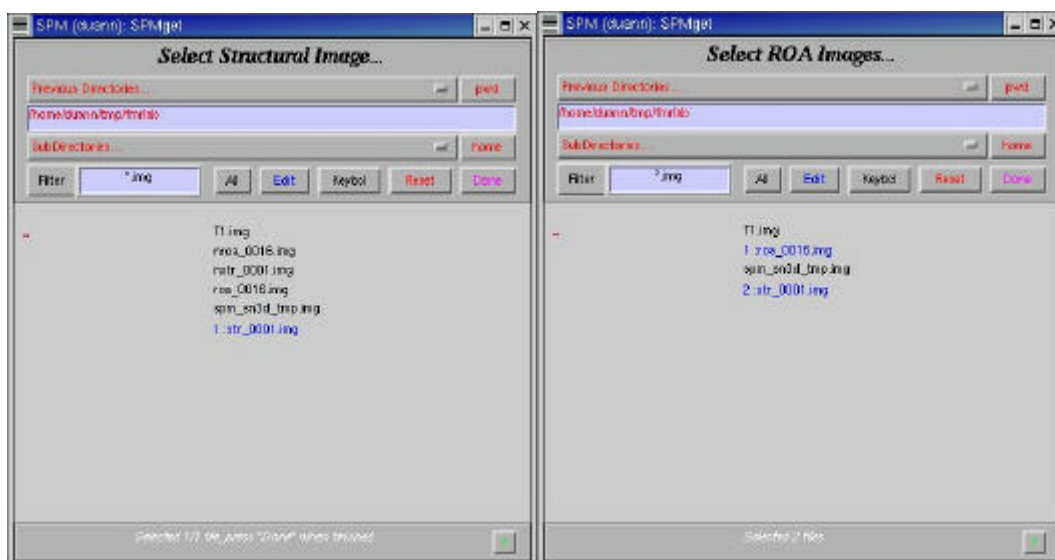
Temins|
File Edit Settings Help
b =
    Empty matrix: 0-by-1
>> clear_workspace
b =
    Empty matrix: 0-by-1
>> clear_workspace
b =
    1
>> farlab
>> Converting structural images...
Writing str_0001.img ...
Writing str_0001.hdr ...
Writing roa_0016.img ...
Writing roa_0016.hdr ...

```


If the structural brain images exist, the export function will also export these images. Note: Including the structural images will allow FMRLAB to spatially normalize the resulting ROA maps to the standard Talairach space. Without structural images, FMRLAB will not be able to perform the spatial normalization or visualize the data in MIP, 2-D slice-overlay, or 3-D rendered displays.

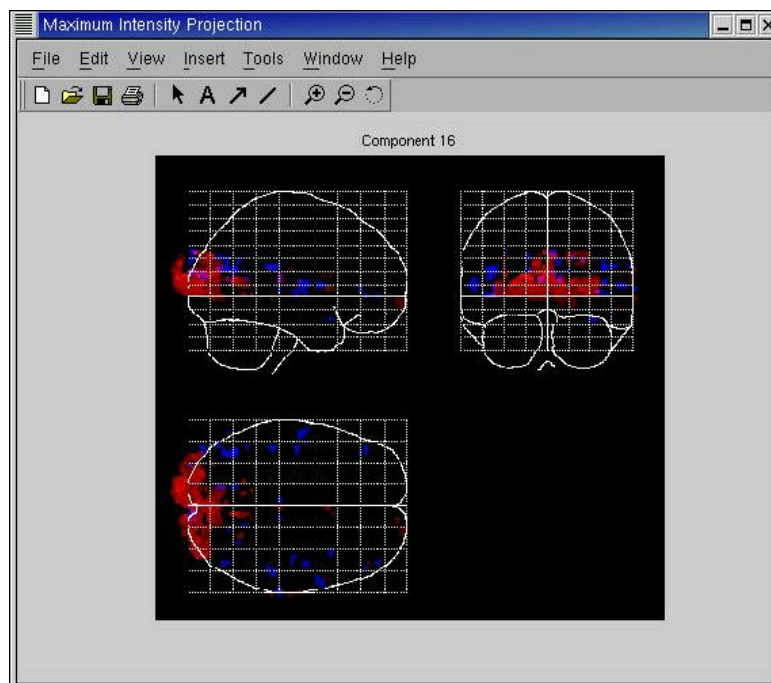
3.6 Spatially normalize the component maps

The spatial normalization used in FMRLAB is adapted from that provided in SPM99. The process requires structural images with higher spatial resolution than normal functional scans. The structural images provide more anatomic details for (a) accurately estimating the Talaraich transformation parameters, and (b) applying these parameters to the resulting ROA maps. Thus, in the **Select Structural Image** window (the left side of the figure below) select **str_0001.img**, the standard filename used by FMRLAB for the first exported structural image in ANALYZE format. Then, in the **Select ROA Images** window (right side below) select all those ANALYZE-format ROA maps of interest (having names like **roa_005.img** meaning ROA map of component 55), as well as the structural image itself. FMRLAB will determine the Talaraich transformation parameters from the structural images, and will apply those parameters to spatially normalize the ROA maps. The structural brain images will also be normalized for visualization purposes. The resulting spatially normalized files will have names like **nstr_0001.img** (normalized structural slice number 1) and **nroa_0055.img** (normalized ROA map 55).



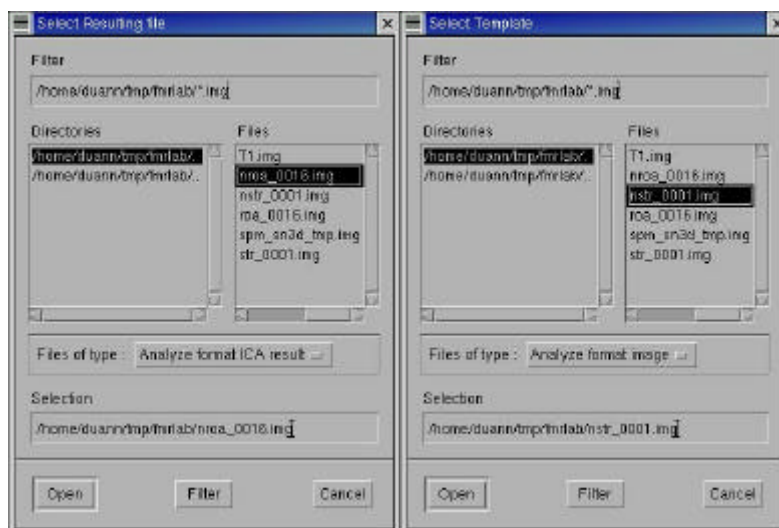
3.7 Produce a maximal intensity projection (MIP) display

By selecting **Visualize > MIP Display** and picking the desired spatially-normalized ROA map (be sure to select one that has been normalized), a maximum-intensity projection (MIP) display will be created (as below).



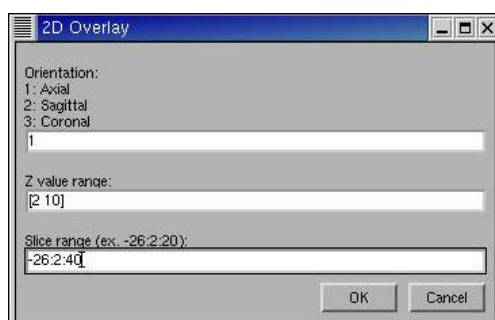
3.8 Produce a 2-D slice-overlay display

Selecting **Visualize > 2D Display** will pop up two file-selection windows (as below). The first (left) allows you to select the spatially-normalized ROA map to display. The second (right) allows you to select the template on which you want to overlay the ROA map.

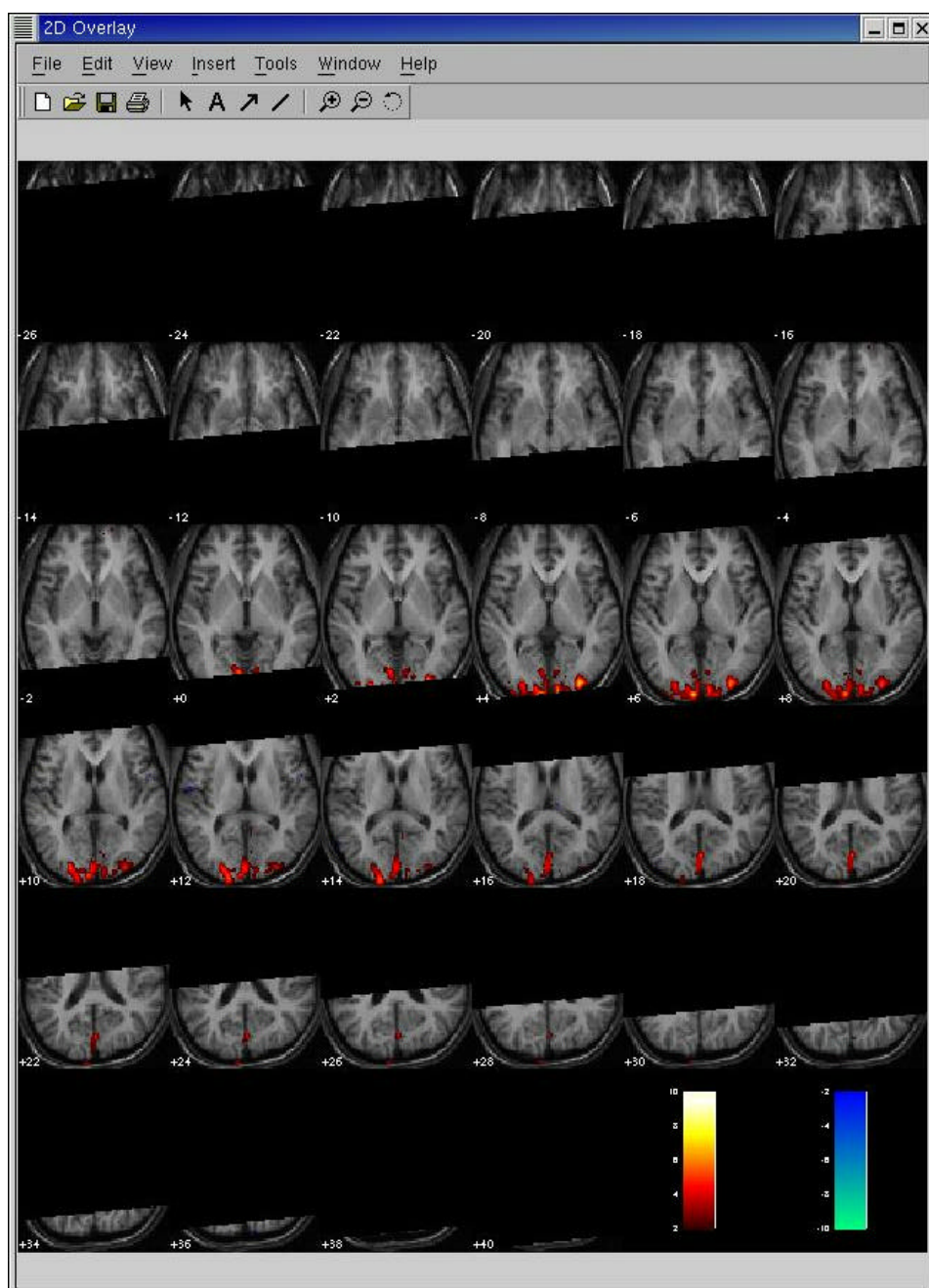


FMRLAB provides two options for this template. One is the smoothed standard brain template provided in SPM99 (“**T1.img**.”) The second is the spatially normalized individual brain structural image exported by FMRLAB, normally “**nstr_0001.img**.” The individual subject’s structural image provides more detail about the brain structures associated with a component than the smoothed standard brain template. However, the individual brain template can only be used if the subject’s structural images are available and are used in the spatial normalization process.

After the ROA map file and the plot template are specified, three more parameters must be specified for the 2-D slice-overlay display. The first is the slice orientation to display. This can be either (1) axial, (2) sagittal, or (3) coronal. The z-value range gives the lower and upper bounds for thresholding the ROA z-values. For example, entering [2 10] will ignore voxels with small z values between 2 and –2, and will not differentiate between z values greater than 10 (or between those with 2 values less than –10).



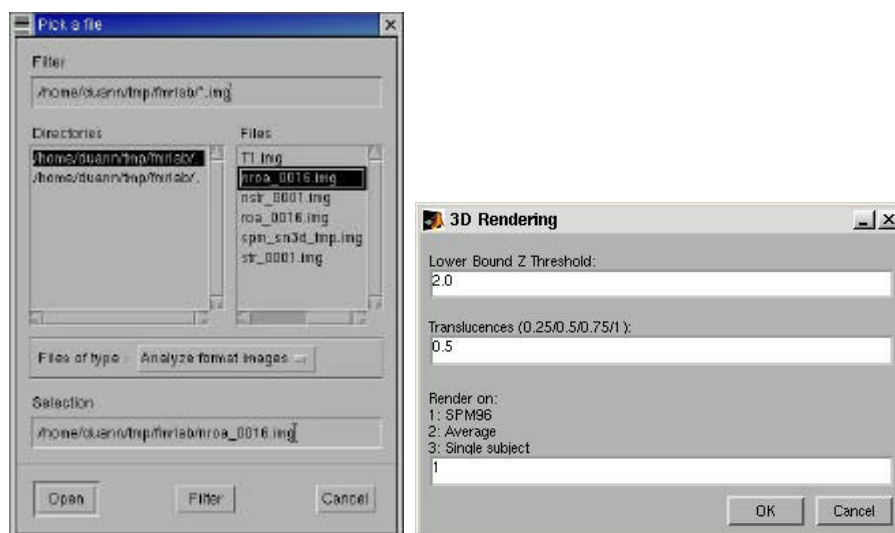
Positive z values will be assigned to warm (red | orange | yellow) colors and the negative z values cold colors (green | blue | indigo). The third parameter (above) identifies the range of slice coordinate (in mm, in Talairach space) to display. It can be given by a 1-D vector (such as **[2 3.5 5 ...]**), or in Matlab **[start:gap:end]** format. Example: Entering **[-26:2:20]** with axial orientation, tells FMRLAB to display slices from z-axis position -26 mm to +20 mm with a 2-mm gap. A typical 2-D slice-overlay display is shown below.



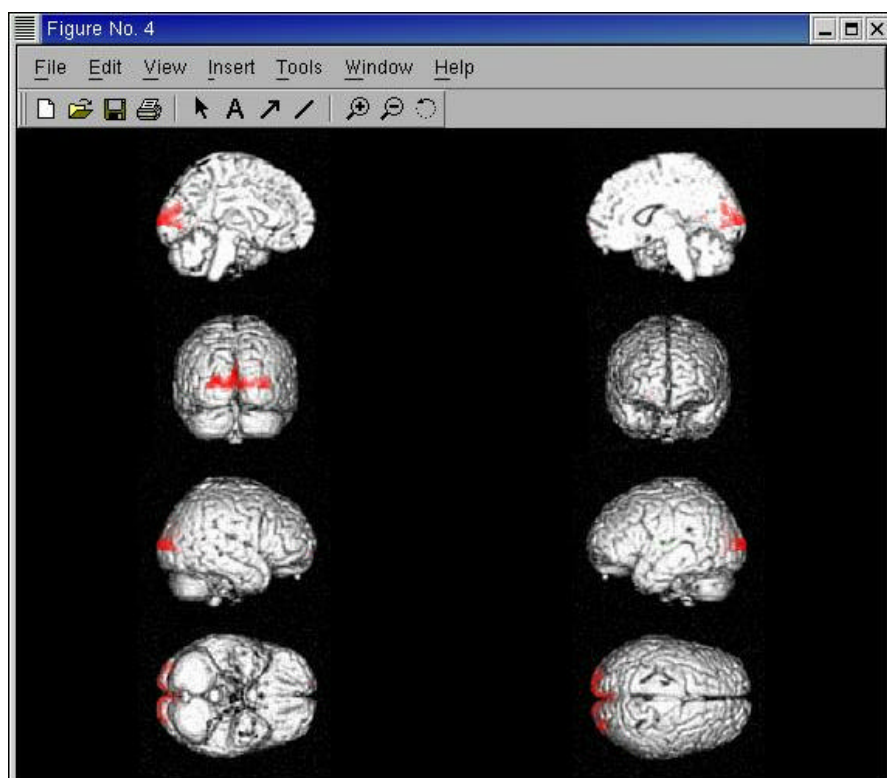
3.9 Produce a 3-D head-model rendered display

3-D rendering is probably the most popular and well-accepted format for displaying fMRI results. FMRLAB uses the 3-D template provided with SPM99 to overlay the region of activity (ROA) map onto the SPM'99 3-D template brain. The ROA maps must first be spatially normalized to standard Talairach coordinates using the spatial normalization function (see Section 3.6 above).

When the normalized ROA maps have been created, select **Visualize > 3-D Display** to start the 3-D rendering. First, the **Pick a File Window** will pop up (below, left), allowing you to specify the ROA map to display (normally a file named, again, something like “**nroa_005.img**”). When ready, click the **Open** button to close the file selection window and bring up the **3-D Rendering** window (right below). The top parameter input is the lower-bound z-value threshold, which is used to ignore insignificant voxels in the ROA map. The second entry specifies the translucency of the color display. Translucency allows the viewer to “see through” the brain to activations within the outer brain surface. Typical values for translucency are 0.25, 0.5, 0.75, 1 or NaN. The lower the value, the more opaque the brain template. To display without translucency, use [NaN] (Matlab for “not a number”). The third option allows the user to specify which 3-D brain template to use. Possible values are 1 (SPM96 template), 2 (subject-average template) or 3 (single-subject template).



After these inputs are complete, press **OK** to begin the 3-D rendering, which will produce a figure like that below.



For other functions in the FMRLAB toolbox, see the FMRLAB manual. We hope that you will enjoy exploring the complexity of BOLD data sets using FMRLAB, and that in so doing, you may make exciting discoveries about what hemodynamics may tell us about how human brain dynamics support experience and behavior.

Jeng-Ren Duann

Scott Makeig

La Jolla 9/2002

Appendix – Function List of FMRLAB

A. 1 Main Files:

fmrlab.m	main function of FMRLAB toolbox
fmrlab.mat	MAT file to keep necessary parameters for FMRLAB toolbox
license.txt	GNU license
boldimage.m	image the intertrial dynamics of BOLD signal
clear_fmri_global.m	clear FMRI data structure from the working environment
clear_workspace.m	clean up the workspace by closing all the opened windows by FMRLAB
dilation.m	perform dilation on input image (used in extract_brain_ui())
erosion.m	perform erosion on input image (used in extract_brain_ui())
execute_ica.m	execute ICA with GUI for users to specify parameters
export_result.m	export region of activity (ROA) maps to ANALYZE format for further visualization
extract_brain_by_edit.m	set threshold value for removing off-brain voxels by key in value in edit box
extract_brain_ui.m	GUI for user to remove the off-brain voxels
extract_brain_ui.mat	MAT file to keep the necessary fields for extract_brain_ui()
fmri_bpfiler.m	perform ideal high/band/low-pass filter on fMRI time courses
get_status.m	get current status of FMRI data structure
ica_linux	main program of binary ICA
jr_color.m	specify the colormap used to display the functional ROAs
jr_normalization.m	3D normalize ROA map to standard brain template
jr_render.m	3D rendering of ROA map on 3D standard brain template provided by SPM99
load_dataset.m	load FMRI data structure up to the working space
make_blobs.m	read spatially normalized ICA ROA map and convert it to the data structure used to in 3D rendering processes
map_on_fmri.m	component browser by overlaying ROA onto 2D slices of functional images with interactive graphic user interface
map_on_struc.m	component browser by overlaying ROA onto 2D slices of structural images with interactive graphic user interface
modify_param.m	modify necessary parameters for data analysis and visualization
modify_struc_info.m	modify parameters of structural images
progressbar.m	progress bar showing the progress of the running program
pvafigmap_ui.m	display percentage variance accounted for (pvaf) map with graphic user

	interface
read_analyze_hdr.m	read header file of images saved in ANALYZE format
read_structure.m	read structural images according to the specified parameter
remove_dummy.m	remove dummy scans from the fMRI time series data
reselect_fmri.m	select new fMRI data set with the same parameters
rm_slice.m	remove noisy slices from fMRI data
roamap_ui.m	display ROA maps with graphic user interface
roaproj_ui.m	ROA back-projection to find the back_projected ICA time courses and mean time course of the ROA voxels and calculate the PVAF for a specified component
roatc_ui.m	find the mean time course of the ROA voxels
save_dataset.m	save FMRI data structure as .fmr file in disk
set_fmri_global.m	construct FMRI data structure as global variable in current workspace for further analysis
set_fmri_global_ui.m	set_fmri_global() with interactive graphic user interface
set_struc_info.m	select structural images into FMRI data structure and set the necessary parameters
show_2d.m	call show_actslice() and display normalized ICA ROA maps onto normalized 2D structural image of individual subjects or 2D brain template in a slice-by-slice manner
show_3d.m	display normalized ICA ROA maps onto the rendered 3D brain templates provided by SPM99
show_actslice.m	overlay the activation map onto the structural image. Both structural images and activation map should be normalized to the standard brain space (Talairach space) with SPM
show_mip.m	display normalized ICA ROA maps on mip template provided by SPM99
show_parameters.m	show parameters of image acquisition and analysis in main window
slice_timing.m	adjust image inhomogeneity due to different acquisition timing for each slice
slice_timing_ui.m	graphic user interface of slice_timing()
slice_timing_ui.mat	necessary information needed for slice_timing_ui()
spatial_smooth.m	spatially smooth image slices to remove the spiky noise due to signal loss in image acquisition
temporal_smooth.m	temporally smooth fMRI time courses with 3 time-point averaging
tightsubplot.m	compact version of subplot()

A.2 Functions from ICA Toolbox

binica.m	Matlab function to interface stand alone binary version ICA (executable by C)
binica.sc	script file to keep initial values for ICA training
cbar.m	showing color bar
eegfilt.m	(high band low)-pass filter fMRI time courses using two-way least-square FIR filtering (Signal Processing Toolbox needed)
floatread.m	read floating-point binary data from a file
floatwrite.m	write floating-point binary data into a file
icadefts.m	define ICA defaults
sbplot.m	create axes in arbitrary subplot grid positions and sizes
scale.m	scales an image such that its lowest value attains newMin and its highest value attains newMax
textsc.m	print text at the specified location in Matlab figure

A.3 Functions from SPM'99 Toolbox

mip.mat	template file for maximal intensity projection (MIP) display
render_single_subj.mat	3D rendered brain template from single subject to render the resulting ROA maps
render_smooth_average.mat	3D rendered brain template from smoothed averaged brain to render the resulting ROA maps
render_spm96.map	3D rendered brain template from SPM96 to render the resulting ROA maps
spm_affsub3.m	highest level subroutine involved in affine transformations
spm_atranspa.m	Multiplies the transpose of a matrix by itself - a compiled routine
spm_atnasp.mexlx	mex file for spm_atranspa() in Linux.
spm_chi2_plot.m	display a plot showing convergence of an optimization routine
spm_conv_vol.m	convolves a mapped volume with a three dimensional separable function
spm_conv_vol.mexlx	mex file for spm_conv_vol() in Linux
spm_create_image.m	create an image file
spm_dctmtx.m	creates basis functions for Discrete Cosine Transform
spm_figure.m	setup and callback functions for Graphics window
spm_get.m	user interface for filename selection
spm_get_space.m	get or set the best guess for the space of the image
spm_global.m	returns the global mean for a memory mapped volume image
spm_global.mexlx	mex file for spm_global() in Linux

spm_hread.m	reads a header of ANALYZE formatted image
spm_hwrite.m	write a header of ANALYZE formatted image
spm_list_files.m	lists files and directories
spm_list_files.mexlx	mex file for spm_list_files() in Linux
spm.m	Statistical Parametric Mapping (startup function)
spm_matrix.m	returns an affine transformation matrix
spm_platform.m	platform specific configuration parameters for SPM
spm_project.m	forms maximum intensity projections
spm_project.mexlx	mex file for spm_project() in Linux
spm_sample_vol.m	returns voxel values from a memory mapped image
spm_sample_vol.mexlx	mex file for spm_sample_vol() in Linux
spm_slice_vol.m	returns a slice through a memory mapped image
spm_slice_vol.mexlx	mex file for spm_slice_vol() in Linux
spm_smooth.m	3 dimensional convolution of an image
spm_str_manip.m	miscellaneous string manipulation options
spm_type.m	translates data type specifiers between SPM & Matlab representations
spm_unlink.m	routine for silently deleting files on disk
spm_unlink.mexlx	mex file for spm_unlink() in Linux
spm_vol_ecat7.m	get header information etc. for ECAT 7 images
spm_vol.m	get header information etc for images
spm_vol_minc.m	get header information etc. for MINC images
spm_write_plane.m	write a transverse plane of image data
spm_write_sn.m	write out normalized images
T1.hdr	header file of SPM99 T1 template images
T1.img	image file of SPM99 T1 template images

A.4 Function from Supplement of SPM'99 Toolbox

slice_overlay.m	overlay functional map onto structural images in standard Talairach coordinates
-----------------	---