Last Modified March 23, 2005

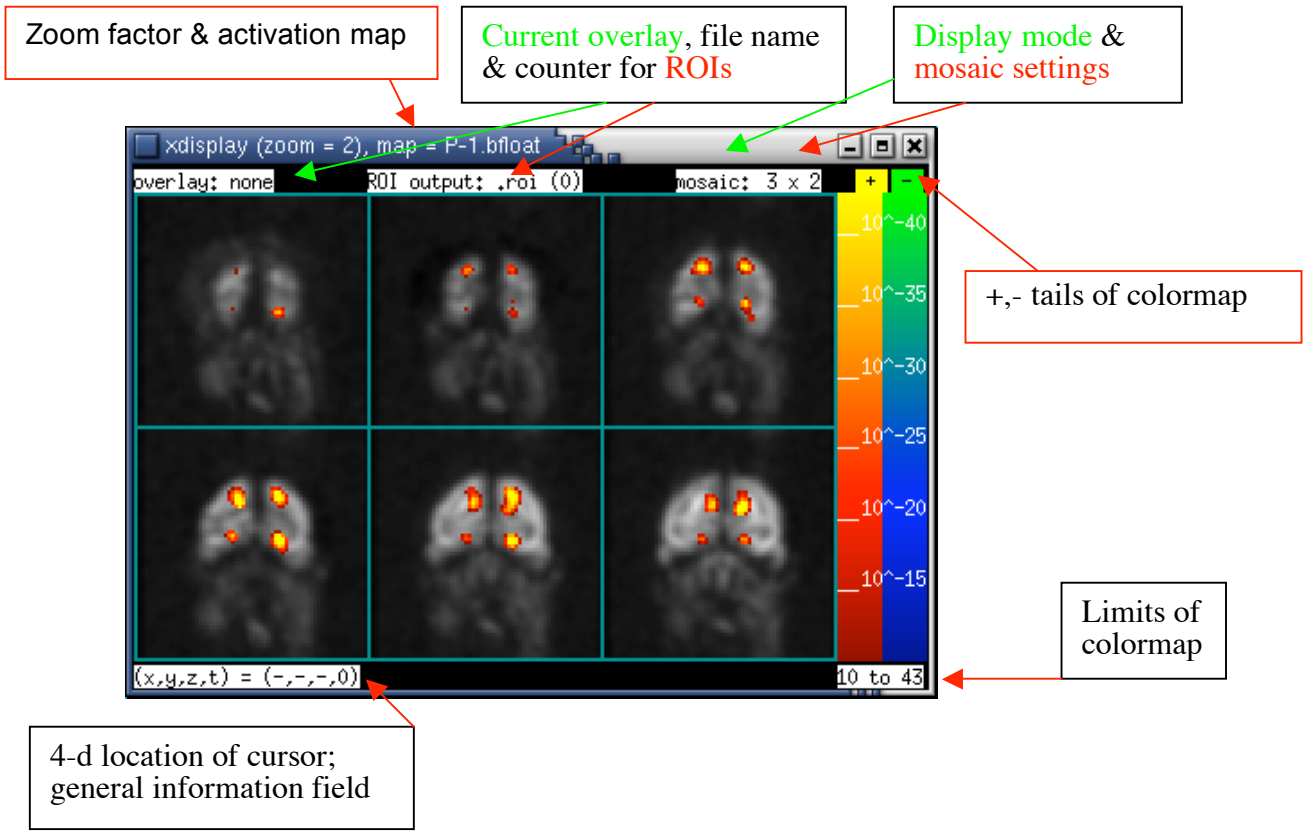
xdisplay Documentation

xdisplay is a simple, intuitive, low-overhead visualization tool for 4-dimensional (x, y, z, time) data.

- All code is written using standard c and standard X-windows (X11), so it should run on almost any platform. Since X-windows doesn't provide much support for a graphical user interface, xdisplay relies heavily on the mouse and arrow buttons. Some interactive text fields are enabled, but most commands use single key strokes.
- Visualization of the 3rd spatial dimension (e.g., slice or z phase-encode) uses either a mosaic format, to show multiple slices simultaneously, or a tri-plane display of orthogonal projections.
- The user can page through multiple activation maps, using p-value format or linear scales.
- Region of interest analysis can be performed interactively, or by using pre-defined overlays.
- Visualization of fMRI data is tailored to an implementation of the General Linear Model (GLM), although activation maps that have been generated by other software can also be displayed.
- For analysis of pharmacological ("phMRI") data, regressors for the general linear model can be defined interactively.

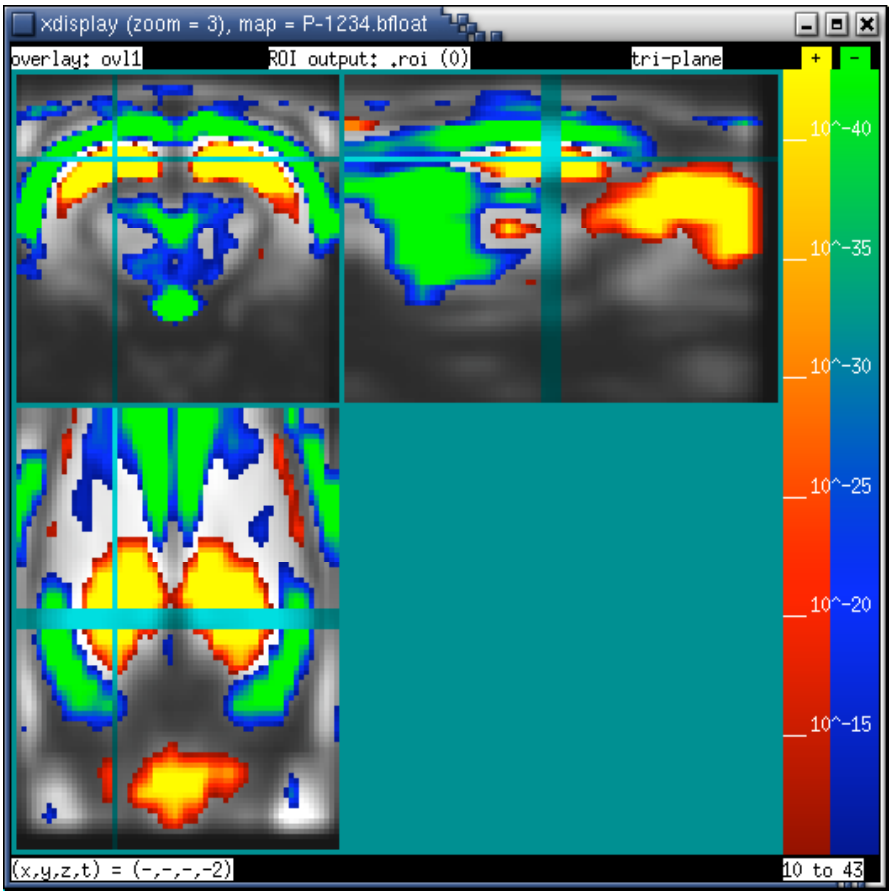
Screen-shots: image window

The screen-shot below shows 6 slices of a larger data set that have been selected (using "control-z" and then the mouse) and displayed in a 3x2 mosaic format. The activation map shows a retinotopic representation of a 4-point visual stimulus that was presented to an awake macaque. Within this screen-shot, interactive text fields include the overlay (top left), display mode and mosaic setting (top right), and tails of the colorbar. The zoom factor (size of window) can be changed ('z' or 'Z'), the map (P-1.bfloat in the screen-shot) can be changed by stepping through a list (+= or -_ with cursor on colorbar), and the colorbar scale can be altered (arrows or mouse buttons with cursor on colorbar).



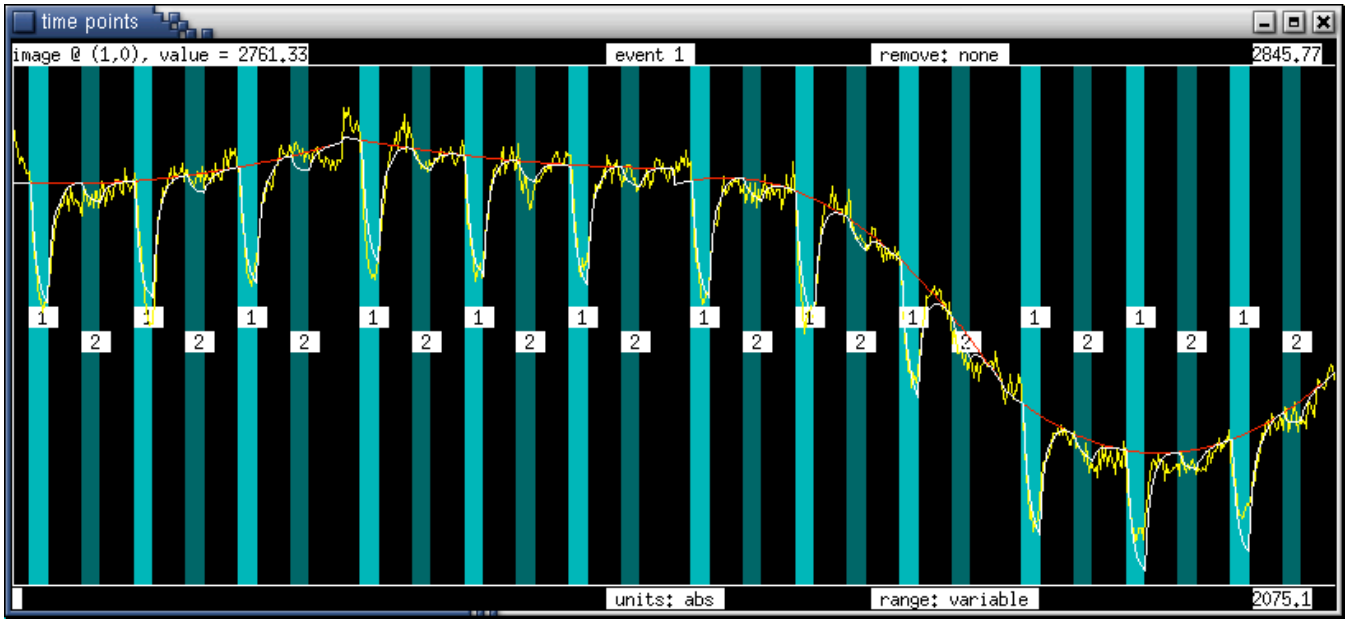
Alternatively, data can be displayed in a tri-planar mode using orthogonal projections (RIGHT). This is phMRI data in rats (n=5) showing increases (red-yellow) and decreases (blue-green) of cerebral blood volume. Data were acquired with an asymmetric spatial resolution. The top left panel shows the actual data without any interpolation. The other two panels show x-z and y-z planes, where data in the z dimension has been interpolated so that the proper aspect ratio is obtained. In order to obtain the proper aspect ratio, the spatial resolutions (x,y,z) must be read by ***xdisplay*** (see input file format below).

The cross-hairs above can be moved using the mouse or arrow functions. The display of cross-hairs can be changed (lines or pixilated, as shown) or removed.



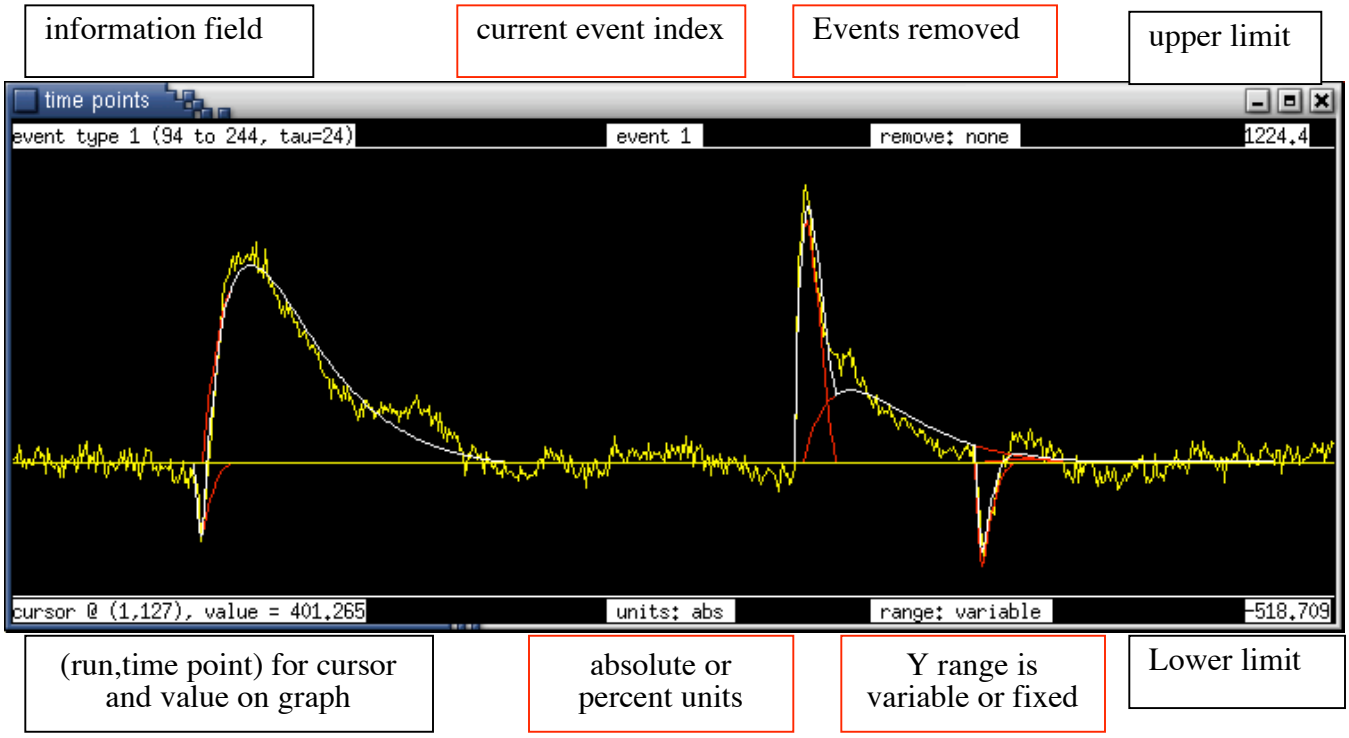
Screen-shots: graph window

The screen shot below shows a graph window that displays the time course for a region of interest with a GLM definition of stimuli. In this display, events are labeled as 1 or 2 (with a type of "square" corresponding to a square function for neural activation), an impulse response function has been defined and convolved with the timing diagram (timing windows are shaded), and a 2nd order baseline is employed with a separate baseline for each of the 4 runs.



The next screen shot shows events defined as gamma functions for pharmacological stimuli. The baseline has been removed ('b'), and labeling of events is not displayed. Red boxes indicate fields than can be changed by a mouse click.
The left fields on the top and bottom show information. Other fields in the graph are

- The current event number: This is the field for creating or modifying new events. Clicking on this field with step the list of currently defined events, plus the next undefined event, plus the option to "ignore" regions selected with the mouse (e.g., the first time point in a run, images acquired prior to injection of MION, region with image artifacts or motion, …).
- One can look at residual data in which one or more events have been removed from the fit. Clicking on this field removes individual events. To remove multiple events, use "control-r", and then input a list of events.
- The units field toggles between absolute or percent units in the graph display.
- The range field toggles between variable or fixed. Arrow commands increase or decrease the upper and lower limits.



Other graph windows include x or y projections through the data. To bring up a graph window, or to change from one graph window to another, enter keystrokes on the image window:

- 't' for time graph
- 'x' for a projection along x,

'y' for a projection along y.

Input file format

At the current time, the only supported input file formats are

- binary short integers (file.bshort) with associated header (file.hdr),
- binary long integers (file.blong) with associated header (file.hdr),
- binary floating point numbers (file.bfloat) with associated header (file.hdr)

Header files are simple text files that are delimited by key words. The general format is "keyword [values]". Minimally, the header file needs to provide the matrix dimensions. This can be accomplished in one line (matrix [x] [y] [z] [t]) or in 4 lines (x [x], so on). Additional useful information includes the spatial

resolution, which is necessary to provide the proper aspect ratio in the tri-planar display mode, and the byte order, for those who switch between PCs and Macs. To provide some compatibility with an older display program (***xds*** by Tim Davis), the first line can be a series of 4 numbers in this order: [# y] [#x] [#images] [byter order]. The number of images combines the z and time dimensions, since that package provided only 2-dimensional display in space.

An example header file is shown below. Note that the "#" character enables comments.

```
64 128 2000 1              # This optional 1st line can be used for xds compatibility
x 128                      # All other lines are for xdisplay
y 64                       # The order of x,y,z,t is irrelevant
z 20
t 100
resolution .35 .35 1.0     # mm per pixel in x, y, z; useful for tri-plane display mode
byte-order 1               # optional; default is 1 for data created on PC, but use 0 for Macs
```

Unix command line options

Input options are listed below in groups according to function. Input follows the common unix syntax, in which each option starts with a dash and then has optional arguments (e.g. xdisplay –flag [optionanal argument]). There are two exceptions to this syntax.

1) Files to be displayed in grayscale (i.e., not activation maps) can either use no flag or use the **-i** flag (where **i** stands for input). In other words, **xd –i input.bshort** is the same as **xd input.bshort**.
2) If no options are given (**xd** or **xd &**), the program will look for a file named **.xdisplay** on the current directory and read the arguments from that file. This is a very useful feature, because one can enter an appropriate string once into the file **.xdisplay** and then always bring up the full results of an analysis without having to type a complicated input syntax.

Note that the order of input options is irrelevant, so that **xd fmri.bshort –A P.bfloat** is the same as **xd –A P.bfloat fmri.bshort**.

Activation overlays

Multiple activation maps can be overlaid on the image(s). For instance,

xd run.bshort –A map1.bfloat –A map2.bfloat … –A mapN.bfloat

will overlay N activation maps on the image(s).

All files containing activation data use a linear scale. Options **–A** and **–p** assume that the file contains p-values, and so the text written on the colorbar uses the 10^n format, whereas option **–a** uses a linear scale on the colorbar. The difference between **–A** and **–p** is that the files are assumed in to contain $\log_e(p)$ and $\log_{10}(p)$, respectively. The $\log_e(p)$ format was employed by a previous display program (***xds***) and associated analysis tools.

xdisplay uses 2 colorbar threshold settings: one for P-value maps and one for linear maps. Multiple p-value maps all use the same upper and lower threshold, and multiple linear scale maps all use the same (different from p-value maps) upper and lower thresholds. If a p-value maps precedes a linear maps in memory (or in the order entered on the unix command line), the linear maps applies an implicit p-value threshold using the preceding p-value map.

- **-A** (Activation)

Assume the file contains $\log_e(p)$, the standard method used by xds.

- **-p** (p value)
- **-P**

Assume the file contains $\log_{10}(p)$. This is the format generated by ***glm***.

- **-a** (activation)

Use a linear scale format to write text on the colorbar (e.g., the file contains T values, or % CBV, …). A special feature for these types of maps is that an implicit threshold is applied by previous p-value map, so that voxels on the linear activation map only appear colored if the same voxel is colored on the previous p-value map. For instance, **xd file.bshort –p P-1.bfloat –a fCBV-1.bfloat –p P-2.bfloat –a fCBV-2.bfloat** will apply the P-1 filter to map fCBV-1 and the P-2 filter to map fCBV-2. On the other hand, the command **xd file.bshort –a fCBV-1.bfloat –p P-1.bfloat –p P-2.bfloat –a fCBV-2.bfloat** will apply the P-2 filter to map fCBV-2 as before, but no filter will be applied to map fCBV-1.

GLM files

- **-G** (GLM)

Read a GLM control file. Only one control file can be used in the list of arguments.

xd –G glm.dat

This command goes through the GLM control file, reads all the fMRI runs in the files, all the timing files, and all the P-*.bfloat files (p-value maps), if they exist. Note that a successful run of the stand-alone GLM analysis program (~jbm/glm/standalone/fmri-glm) will create the file **.xdisplay** containing a command as above, so that the unix command "xd &" without any arguments will look to read the arguments from **.xdisplay**, and suck in everything produced the GLM analysis.

- **-T** (T statistic)

Include the T maps (T statistic, or contrast-to-noise ratio), along with the P maps when using the –G option above.

xd –G glm.dat -T

- **-S** (Signal change)

Include the S maps (signal change), along with the P maps when using the –G option above.

- **-g** (GLM timing file)

Read a file containing timing information. Typically, these files have used an extension of ".glm". Multiple files can be read:

xd fmri1.bshort fmri2.bshort … –g stim1.glm –g stim2.glm …

The number of times points in all the fmri files should agree with the sum of the time points in all .glm files. Normally, one would choose to use –G above. When there are a whole lot of data files, the method above may be useful to look at a small subset.

Image (grayscale) files

- **-i** (image file, or input file)

This is a file containing fMRI data (or other data) to be displayed in grayscale format. This is the default option, so it need not be specified.

xd –I file.bshort
xd file.bshort

- **-e** (image file that is Excluded from GLM)

This is a file containing fMRI data (or other data) to be displayed in grayscale format. The diffence between this option and the option above is that files appearing with **–e** will be excluded from the GLM signal model. This is useful for adding anatomical images, for instance, to an fMRI time course so that activation maps can be displayed on top of the anatomical images. All data specified by this option is appended to the beginning of the time course, but not shown in the time course graph.

xd –G glm.dat –e anatomy.bshort

Other options

- **-h** (help)

Print out help about input options.

- **-o** (Overlay list)
- **-O**

Read a list of overlay files. This allows any number of pre-defined overlays to be read into memory so any of the overlays can be used within xdisplay.

xd –G glm.dat –o overlay-list.dat

The format for the file specified by this option is as follows:

mpfc /homes/1/me/overlay/mpfc.ovl
thal /homes/1/me/overlay/thalamus.ovl
…

Each line gives a label to the overlay and a file location for the overlay. The format of the **.ovl** files is a list of (x,y) pixel locations, as used by xds. Files of this type can be written as output from **xdisplay** by using the **O** command.

- **-z** (zoom)
- **-Z**

Use the specified "zoom" factor to set the beginning size of the image display window. If this option is not used, a default size will be chosen based upon the size of the display.

xd file.bshort –z 2

- **-3** (3-plane display mode)

This option will start the display using the tri-planar mode, rather than the mosaic mode.

xd file.bshort –3

Interactive options

All commands are one-key events. Commands can either be associated with the image window or the graph window. I've tried to avoid overlap between graph and image commands, so that most commands will function on both windows. Additionally, commands may be interpreted according to context (e.g., whether or not an overlay is on the image) or the location of the cursor. This may sound confusing, but you get used to it pretty quickly. The arrow keys and the mouse buttons, in particular, can do many different jobs.

Arrows and mouse buttons: contextual commands

- **↑ and ↓** Perform actions with the following priorities.
  1. If the cursor is on the activation colorbar, increase (↑) or decrease (↓) the threshold nearest to the cursor.
  2. If the cursor is highlighting text in one of the special fields above or below either the image or the graph, then toggle through the allowed options in that field in a forward direction (e.g., A→B→C→A→…) for ↑ and the reverse direction for ↓.
  3. If a cross-hair (tri-plane display mode) or a projection associated with an x or y graph are shown, move the cross-hair or projection up/down.
  4. If the cursor is on the image and an overlay is displayed on the image, move the overlay.
  5. If the cursor is on the image and an overlay is NOT displayed on the image, change the *brightness* up or down.
  6. If the cursor is on the graph, set the graph y scale to "fixed" (shown below graph), and increment or decrement the limit (upper or lower) that is closer to the cursor.

- **← and →** Perform actions with the following priorities.
  1. If the cursor is on the activation colorbar, page through the activation tails (for →, the order is +/-, +, - → +/-).
  2. If the cursor is highlighting text in one of the special fields above or below either the image or the graph, then toggle through the allowed options in that field in a forward direction (e.g., A→B→C→A→…) for ← and the reverse direction for **→.** If the cursor is on the field of pre-defined overlays, add overlays to the image using pre-defined colors; this enables colored overlays for display purposes (e.g., manuscripts).
  3. If a cross-hair (tri-plane display mode) or a projection associated with an x or y graph are shown, move the cross-hair or projection up/down.
  4. If the cursor is on the image and an overlay is displayed on the image, move the overlay.
  5. If the cursor is on the image and an overlay is NOT displayed on the image, change the *contrast* up or down.
  6. If the cursor is on the graph, move the graph index (time, x, or y) to the selected point.

- **Left mouse button (#1)**
  1. If the cursor is on the image and the graph window is not showing an x or y projection, draw an overlay by dragging the cursor.
  2. If the cursor is on the image and the graph window is showing an x or y projection, move the location of the projection.
  3. If the cursor is on the activation colorbar, change the nearest threshold to the selected point on the colorbar.
  4. If the cursor is on the field above the colorbar (+ and -), select the tail for the colorbar and activation display.
  5. If the cursor is on selected text above or below the image or graph, toggle through the list of possible choices using a forward direction.
  6. If the cursor is on the graph window, move the graph index (x, y, or time) to the selected point on the graph, and update the image window to reflect the change. A vertical line on the graph marks the selected graph point. For a time graph, the image moves to the current time point. For a projection graph, the highlighted pixel along the x or y projection moves on the image window.

- **middle or right mouse buttons (#2, #3)**
  1. If the cursor is on the image, adjust the brightness (up-down) or contrast (left-right).
  2. If the cursor is on the activation colorbar, expand the nearest threshold by about 10% of the colorbar range.
  3. If the cursor is on selected text above or below the image or graph, toggle through the list of possible choices using a backward direction.
  4. If the cursor is on the field above the colorbar (+ and -), change sign for the colorbar and activation display (+ to -, and vice versa).
  5. If the cursor is on the graph window, use the mouse motion to select a range for fitting. For an x or y projection, this selects a region for Gaussian fitting. For a time graph, this affects the GLM. If the current event type is “ignore”, ignore the selected region (actually, toggle every time point in the region between exclusion and inclusion). If the event type points to an event that is not currently included in the GLM, or the event type points to an existing event but the selected range does not overlap with a stimulus of that event type, add the event/stimulus using the selected range. If the event type points to an existing event in the GLM and the selected range overlaps an existing stimulus region, change the limits of that stimulus to the selected range. If the start/end of the selected range (button press / button release) point to the same image, delete an event or stimulus at that location.

Other commands on Image Window

- **h** help

- **A** Animate. Use any key to quite. This is a case where the image window must be active to stop the animation.

- **P** Calculate p values, automatically generating a list of test conditions. This command is designed mostly for interactive GLM analysis of drug injections, rather than complicated sensory stimulation paradigms that are better handled by using external timing files.

- **q** quit the program.

- **r** write ROI for the current overlay.

- R write ROIs for all pre-defined overlays.

- **w** Window the grayscale images, using an overlay if one is active. Currently, there is only one window setting (brightness, contrast) for all images. In other words, images cannot be windowed individually.

- **x** Bring up a graph window that shows projections along x. When the graph first comes up, it will show projections of the underlying grayscale image. If an activation map is active, the subsequent "x" commands will toggle between showing the activation map values or the grayscale image values.

- **y** Same as x, but for y projections.

- **t** Bring up a graph showing the time course for the stack of images.

- **z** -zoom window
  Decrease the zoom factor (size) of the image window.

- **Z** +zoom window
  Increase the zoom factor (size) of the image window.

- **Control-z** select slices (z dimension) for display
  When the full complement of slices is shown, this will prompt the user to select a sub-set of slices for display. When a sub-set of slices is shown, this command will revert back to showing all slices.

- **+ or =** next image
  Show the next image (time point) in the image stack.

- **- or _** previous image
  Show the previous image (time point) in the image stack.

Other commands on the Graph Window

Once a graph is active (by using 'x', 'y', or 't' on the image window), the following commands apply when the graph window is active. The functions of arrows and mouse buttons on the graph window are described above.

- **Spacebar**

If shaded regions are shown to indicate GLM timing, toggle the timing windows on/off.

- **b** baseline on/off
  If a GLM fit is displayed, toggle removal of the baseline on/off.

- **e** select event type
  The event type is displayed on the graph window on the top. Selection of regions using the mouse (below) alters the GLM definition according to the current event type. If the event type is set to ignore, then the mouse selection will set regions to be excluded from analysis.

- **f** Fit.

1. If an x or y graph is displayed, fit the projection to a Gaussian function within the grayscale highlighted window. The location of the vertical bar (corresponding to the highlighted pixel in the window image) gives the starting guess for the peak location of the Gaussian fit.
2. If a time graph is displayed, toggle the shape of the event window between a square and a gamma function. The gamma function is designed for analysis of drug injections. Sensory stimuli should use a square function (until proven otherwise). The shape represents the hypothesized shape of neural activity. If a hemodynamic response function is active (currently enabled only by reading a GLM control file using the –G option on startup), convolve the stimulus shape with the HRF to get the reference vector for each event type.

1 **g** Save current GLM settings.
Write the file glm.dat and a file named runx.glm for the timing of each run. This is a way to start with a GLM (-G option), modify it interactively, then write out the modified settings.

2 **i** Ignore out-of-range values.
This is a test option for monkey analysis. By selecting an overlay near an edge that will be sensitive to motion, large deviations will be seen in the graph window. When y limits of the graph are manually set (e.g., +- 10%), every time point that exceeds the limits will be set to "ignored" in the GLM.

3 **Control-r** remove GLM events
Prompt for a series of events to remove from the GLM analysis. This is similar to using the mouse to step through events in the field labeled "remove" above the graph, except that this command enables multiple event types to be removed simultaneously. This is useful for both analysis and presentation. For example, one could remove brief blood pressure effects as "effects of no interest" from a GLM analysis, or see a drug affects a sensory stimulus by viewing the drug/stimulus responses separately.

4 **u** Units
toggle y units between absolute and percent. Percent units are generally preferable when the time series represent raw images. When the time series contains something like the percent change in CBV or delta-R2, then absolute units are preferable.

GLM – The general linear model for fMRI data analysis

xdisplay incorporates a general linear model for data analysis. Before describing details, a brief background is provided. This is simple GLM implementation without some of the methods that are sometimes used to describe noise (e.g., pre-whitening, or lag-n autoregression).

The general linear model (GLM) is a standard method of describing data in many fields and a standard way of analyzing fMRI data, in particular. The model is quite simple and general. Each voxel in a 4-dimensional fMRI data set is analyzed as a time vector where each member (y_t) is fit using a summation of basis functions (x), including drift terms (d), and noise (n) that accounts for variations not described by the model:

$$y_t = \sum_i x_{ti}\beta_i + \sum_i d_{ti}\beta_i + n_t \qquad [1]$$

This equation is often written using a simplified formalism as follows:

$$\vec{y} = \mathbf{X}\vec{\beta} + \vec{n} \quad , \qquad [2]$$

where **X=[x d]** is called the "design matrix" and the implicit summation over "i" (the index of $\vec{h}$) now includes both the "effects of no interest" (constant and drift terms) and other "event types" that will be analyzed for statistical significance. The "design matrix" contains pre-defined shapes as a function of time that will be scaled to provide the best fit for each voxel. The members of $\vec{\beta}$ are simply coefficients of a "fit" that can be calculated explicitly as

$$\vec{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\,\vec{y} = \mathbf{S}\,\vec{y} \quad . \qquad [3]$$

S is calculated once for a data set, and then the solution of h involves just one matrix multiplication for each voxel. This makes analysis remarkably fast.

Statistical significance can be assigned to individual events, to more complicated tests involving differences or sums of events, or to "omnibus testing" or "general effects" associated with more than one type of specific contrast [ref Friston 1995]. Within the standard GLM framework, a contrast matrix ($\mathbf{C}$) or vector ($\vec{C}$) is used to define statistical tests. Given two event types, a test for the first event is specified by the contrast vector {1 0}, and the difference of the two events is specified as {1 –1}. To reject the hypothesis that the magnitude of the sum of two events matches the baseline condition, the contrast vector is {1 1}. Alternatively, to perform an omnibus test that one or both events reaches significance, a 2 by 2 contrast matrix is employed: {1 0; 0 1}.

Given the solution of parameters in Eq. 2 and the contrast matrix as described above, the statistical significance of one or more contrasts is evaluated by an F test, where the number of degrees of freedom (N_F) is the number of time points minus the number of parameters that have been fit, and N_C is the number of rows in the contrast matrix.

$$F = \frac{\left(\mathbf{C}\,\vec{\beta}\right)^T\left[\mathbf{C}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{C}\right]^{-1}\left(\mathbf{C}\,\vec{\beta}\right)}{\left(\vec{y} - \mathbf{X}\vec{\beta}\right)^2}\frac{N_F}{N_C} \quad , \qquad [3]$$

Although the F test is useful to identify statistical significance associated with multiple contrasts, hypothesis testing generally employs a single contrast (N_C=1). In this case, the contrast matrix becomes a vector ($\vec{C}$), and the F test produces results equivalent to a T statistic, which is the square root of F:

$$T = \frac{\vec{C}\cdot\vec{\beta}}{\sigma} \qquad [4]$$

$$\sigma^2 = \vec{C}\cdot\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\vec{C}\;\left(\vec{y} - \mathbf{X}\vec{\beta}\right)^2/N_F$$

For GLM implementation in conjunction with xdisplay, the code was written with the following in mind:

1) It should interface with a program designed for data analysis (alias ***glm*** ~jbm/glm/standalone/fmri-glm). To analyze data, including many multiple fMRI runs or analyses across subjects, the amount of data in memory should be a tradeoff between minimizing data memory and minimizing limitations of file i/o. To visualize data, sometimes the entire data can be stored in memory; alternatively, just the results can be displayed. In many cases, individual subject data or runs can be analyzed and visualized with ***xdisplay***, and then a cumulative analysis can be done with ***glm*** by reading the same files used in the piece-wise analysis.
2) Data visualization/analysis should work both for 'fMRI' (sensory, motor, cognitive) and 'phMRI' (drug stimuli). For fMRI, stimulus paradigms are generally sufficiently complicated that it is better to use ***xdisplay*** as a visualization tool, rather than as an analysis tool. In this case, timing is defined in an external file, rather than interactively, and ***xdisplay*** can be used to validate the analysis. For phMRI, there are generally a very few events (drug injections) per session, and the neural function may not

be well known. In this case, event durations and shapes can be defined interactively to surf through data sets.

3) In additional to handling multiple images, the viewer should be able to multiple activation overlays, and multiple pre-defined regions-of-interest.

Defining baseline basis functions

Simply giving the order of the polynomial to use specifies the basis functions that describe baseline drift. Use a 2nd order polynomial (`baseline-terms 3') in order to provide a constant level, a linear drift, and a quadratic drift. This seems to be a good choice for analyzing fMRI (sensory, motor, cognitive) paradigms. When an impulse response function is omitted from the GLM control file, the programs assume that pharmacological data is being analyzed, and a 1st order polynomial is used.

Defining event basis functions for fMRI

Event types for sensory, motor, or cognitive functions should be assigned based upon the stimulus timing. Because experimental paradigms can be quite complicated, the best way to assign such events is to create a timing file, as shown below, with an editor. The shape should be "square", and then an impulse response function file will produce the final shape by convolving the IRF with experimental paradigm.

Defining event basis functions for pharmacological MRI (phMRI)

Event types for pharmacological MRI can be created in the same was as described above for fMRI, or events can be defined interactively in xdisplay. Generally, there will be a very few injections per session, and the time course is not known *a priori*. By using the middle or right mouse buttons to define a gamma function or functions for each injection, xdisplay allows one to efficiently surf through the data. Once the time course of a drug is known, results can be analyzed across animals using a consistent event shape.

GLM control files

Both ***xdisplay*** and ***glm*** use a "GLM control file". This file contains the list of fMRI runs, together with an event timing file for each run, plus some additional information, such as an optional hemodynamic response file and the list of conditions for statistical tests. The following is an example GLM control file.

```
notify                  # optional flag to turn on output information
baseline-terms 3        # optional; order of polynomial to fit baseline (1-3, default=2)
normalize-runs          # optional flag to normalize signal intensities by the global average per run
IRF-file bold.irf       # optional file for FIR analyses; `HRF-file' also works
convolve-table 8 9      # optional; table events 8 and 9 will be convolved with the IRF above
conditions 1 2 12 1m2       # mandatory list of statistical tests ("12" = 1+2, "1m2" = 1m2)
runs:                       # mandatory; all further lines indicate data files
../10/clip.bshort ../10/run1.glm      # run file, timing file, optional table file for `table' event type
../11/clip.bshort ../11/run2.glm      # run file, timing file, optional table file
../12/clip.bshort  run3.glm           # run file, timing file, optional table file
```

The character "#" can be used to add comments. "notify is just a variable to control program output. The number of baseline terms should generally be set to 3 (baseline offset, slope, quadratic term) for multiple stimuli, and a value of 2 is appropriate for a single drug injection. The run normalization flag will lead to each run being normalized by the global average signal intensity of the entire run. The IRF files refers to an "impulse response function" file. If this is present, then the events contained in the timing files will not be convolved with an impulse response function as specified in the file (see format below).

Event types in the conditions must appear later in one or more of the timing files. If the event type appears before or without "m", then the corresponding bit in the condition vector ($\vec{c}$ in Eq. [4]) is set to 1. If the event appears after "m", the bit is set to –1. Thus, the condition "12m34" mean "1+2-3-4".

Hemodynamic impulse response function (IRF/HRF) files

If an impulse response function is used, because the `IRF-file' line as shown above was included in the GLM control file, then a file will be read to define an impulse response function as a weighted series of exponentials (see Leite et al., NeuroImage 2002). Events specified in the GLM timing files then will be convolved with the impulse response function, and the result will form the basis functions for GLM analysis of all voxels. Generally, when using an IRF file, it is best to keep all units throughout the GLM in seconds (rather than image numbers).

The IRF file has a simple fixed format, as shown below. The length of the IRF is 60 seconds, in 1 second steps, and a series of 3 weighted exponentials gives the IRF its shape. The time per image volume (given in the timing files) must be divisible by the step size. Using a very small step size (0.1 ms) with a very large data set will slow down interactive display with xdisplay.

```
60      1       3       # Length of IRF (seconds), time step (seconds), number of exponentials
1.5     4.5     13.5    # time constants for each exponential
-0.21   0.41    0.80    # relative weight of each exponential
```

GLM event/timing files

Within the GLM control file, each fMRI data run (or subject) has a corresponding file that gives the event types and the timing of events. Event types generally refer to presumed shapes of the neural response. If an IRF file is used, then all events are convolved with the IRF to define the basis functions for analysis. Current event types, and the file formats for the events, are listed below.

The timing file gives the time per 3-d volume and the total time of the run on the first line, and then lists events, with each event type separated by a blank line (see the example file at the end of this section). Every event has an identifier (integer from 1 to 9). Events can use 4 types or "shapes", which are listed below.

1. Format for square event type

This is the default event type used for standard fMRI protocols (sensory, motor, cognitive). A typical definition within a timing file would like the following. Note that one timing file is given per run, so on and off times are referenced with respect to the beginning of the run. Events cannot overlap in time. In the list below, everything is self-explanatory except for the stimulus magnitude. Generally, this entry should be left out; in fact, I can't think of any reason for it in the context of a `square' event, but it's included for consistency.

```
1       square          # [event identifier] square
30      60      1.      # [on time] [off time] [optional stimulus magnitude; default=1]
120     150             # [on time] [off time] [stimulus magnitude set to 1]
210     240             # etc
```

2. Format for gamma event type

This is the default event type. For this event type, only "on" times are used, and then the gamma function is defined to the end of the run. Obviously, gamma functions will have some overlap when multiple onset times are used in a single run. In principle, stimulus magnitudes can be assigned to account for variations in dose.

```
1       gamma           16      # [event identifier] gamma [time constant tau for t*exp(-t/tau)]
30              1.              # [on time] [optional stimulus magnitude; default=1]
120             2.              # [on time] [optional stimulus magnitude; e.g., double dose]
```

3. Format for drug-IRF event type

This event type provides a 2nd way to analyze drug responses. An FIR approach is used, which is similar to the way normal fMRI is done: use square events, and then convolve with an IRF. However, I didn't want to use the hemodynamic IRF, because sensory and drug stimuli may occur within the same data. Instead, convolution time constants are given in the event definition.

There are 2 convolution time constants. The 1st value is mandatory. If only this value is given, then each square window will be convolved with a gamma function of this time constant. If a 2nd time constant, then a 2-step approach is used. First, the square window is convolved with the 2nd time constant (as an exponential function) to represent the concentration of drug in the blood plasma (i.e., the 2nd time constant is the blood half-life). Then this resulting function is convolved with the gamma function.

```
1       drug-IRF    15      3       # [event identifier] drug-IRF [mandatory tau1] [optional tau2]
20      60          1.              # [on time] [off time] [optional stimulus magnitude; default=1]
120     130         4.              # etc., different magnitudes represent different infusion rates
220     380         .25             # etc.
```

4. Table event type

Tabular data (e.g., blood pressure, eye tracking, motion-correction parameters….) can be entered using this single-line event definition. The event type (`table') is followed by a mandatory integer that gives the column number in the table file for the run. By default, table events will NOT be convolved with the hemodynamic IRF. To convolve the table data, use the line "convolve-table [event list]" in the GLM control file.

```
1       table   2       # [event identifier] table [mandatory column #]
```

Example GLM timing file.

```
1       300             # TR (actually, time per volume) and run duration
                        # An empty line appears before each new event type
1       square          # event "1" uses a square shape
30      60              # on at time 20, off at time 40; time = image number * TR
120     150             # on, off
210     240             # on, off

2                       # event '2"; the default shape is square
75      105             # on, off
135     165             # on, off
225     255             # on, off

3       gamma           20      # event 3 is a gamma function t*exp(-t/tau), with tau = 20
100                     # on at 100, and then the event is played out for the rest of the run
200                     # on again at 200

4       table   1       # event 4 uses the 1st column entry in the table for this run

-1                      # exclude time points from analysis
0 10                    # these are image numbers, not time = image number * TR
```