GPS Implementation Notes

Tom Sgouros

September 26, 2019

1 A look at the data flow

Roughly speaking, GPS is a processor for MEG data, with MRI data used to help place it in context within the brain. There is a manual for it and a cookbook, and both of those are good references for using it. This document contains some implementation notes relevant to porting the software to a new computer system.

1.1 MRI

GPS uses MRI data to localize signals found in the MEG data, and for the visualization of processing results. There is a certain amount of processing of the raw MRI data to squeeze it into a form that is usable by the MEG processing stream, and we describe that here.

Most of the processing of the MRI data is accomplished by a software suite known as Freesurfer. This is used for segmenting the raw data into surfaces, both the cortical surface and surfaces related to the skull (inner, outer, skin). The software also generates an inflated view of the cortical surface, as well as a spherical view.

You can use `freeview` to check out some of the results, which are found in the `MRI` subdirectory. Within that directory, you'll find subdirectories for each of the subjects, plus one for an average subject, constructed from the other subjects in the study. Within these, you'll find a number of subdirectories corresponding to the various processing steps between the raw MRI and the spherical inflation of the cortical surface.

1.2 MEG

The MEG data is at the heart of GPS. The data consists of a time series for each of the MEG detectors attached to the subject, along with the same for each of the EEG detectors. There are over 300 of the former and 70 of the latter. Before feeding it to the Granger processing,

the MEG data is aligned with the experiment's events, and combined with the MRI data to derive a spatial location within the brain for data using the MNE software suite. The data records at these spatial locations are the input to the Granger analysis.

The processed MEG data and the data byproducts from processing it with MNE can be found in the `MEG` and `MNE` directories.

1.3 Data file and directory structure

The inputs and outputs for each step of the GPS process are contained in a directory tree that contains five subdirectories: MRIraw, MRI, MEG, MNE, and Granger. The names are more or less indicative of what is in them, but fair warning: assume nothing.

The GPS workflow is organized into four broad streams that appear as different screens in the GUI (click on the "Analysis" button that appears on the first GUI menu).

MRI Processing the raw MRI data, projecting it onto a standard surface, building averages among the study participants.

MEG Processing the measured MEG data, and lining it up with the experimental "events" around which the study is arranged.

MNE Developing the MRI and MEG data into vertices with individual time series appropriate for putting together into regions of interest.

Granger Steps associated with the Granger causality analysis of the regions of interest.

Utilities There are several utility functions grouped under this heading, including selecting time series, and creating the regions of interest (ROIs).

The next sections will outline a rough account of the data flow among the various subdirectories of the study directory, organized more or less like the menu structure of the GPS/Analysis dialog.

Note that some values are cached in MAT files whose location is specified when you initialize the system with `gps_init.m`. The default location for these files is in the home directory of the GPS software itself: `GPS/parameters/<study name>`

The basic data flow starts with a few ingredients:

1. MRI data, imported with MRI >Import. (That is the Import button on the MRI dialog.)
2. MEG data, imported with MEG >Import.
3. A way to group events. This is the MEG >Process Events stage. This option should not be used through the GUI.

1.3.1 MRI Processing

- Import

  Data In: Data from outside GPS, maybe Martinos DBMS (Bourget), maybe somewhere else on a disk.
  Data Out: Data is put into the MRIraw folder for that subject.

  Imports the raw MRI data (DICOM files). These are high resolution structural slices of a head, instantaneous data. Import is Martinos specific stuff where it's getting the DICOM data from the Martinos DBMS (Bourget).

- Find T1-MPRAGE

  Data In: MRI data from the `MRIraw`
  Data Out: Modifications to .mat files in `GPS/parameters`, and a summary file and unpack.log in the `MRIraw` directory

  Finds the high-resolution structural MRIs that we are interested in – the T1 data — from the DICOM file. We are not interested in the other stuff, such as functional imagery, initial localizers, DTI, etc.

  Note for Mac/OSX: This part of the procedure calls the Freesurfer function `unpacksdcmdir`, which has a timeout in it of 20 seconds. A sample dataset run on my laptop easily takes more than that to process, even though it's all functioning just fine. I hacked the tcl script to change the timeout. (Line 983 of unpcksdcmdir.tcl)

  From within Matlab on my Mac (OS X 10.12), I get this error from `mri_parse_sdcmdir`:

```
mri_parse_sdcmdir --sortbyrun --d GBU_2/MRIraw/GBU_12
                              --o GBU_2/MRIraw/GBU_12/dicomdir.sumfile
                              --status GBU_2/MRIraw/GBU_12/parse.status


dyld: lazy symbol binding failed: Symbol not found: ___emutls_get_address
  Referenced from: /Applications/freesurfer/bin/../lib/gcc/lib/libgomp.1.dylib
  Expected in: /usr/lib/libSystem.B.dylib
```

```
dyld: Symbol not found: ___emutls_get_address
  Referenced from: /Applications/freesurfer/bin/../lib/gcc/lib/libgomp.1.dylib
  Expected in: /usr/lib/libSystem.B.dylib
```

From outside Matlab, runs fine. This was addressed by disabling something called System Integrity Protection (SIP), which is some kind of extra-OS supervision of certain system directories. Apparently it also governs how environments are inherited. Specifically, it prevents environment variables associated with dynamic libraries (e.g. DYLD_LIBRARY_PATH) from being inherited when launching protected processes. To disable SIP, boot into the single user (hold down cmd-R during boot), bring up a terminal window, and do `csrutil disable; reboot`.

- Organize

  Data In: MRIraw

  Data Out: MRI

  This moves the raw data into place for further processing. For each subject, it creates an `MRI` directory, and populates it with lightly-processed MRI data as well as scripts, logs. Runs `mri_convert.bin`, `mri_add_xform_to_header, mri_normalize`, `mri_em_register`, `mri_watershed`, at least. Creates and populates subdirectories mri, scripts, and touch, and empty placeholders label, stats, surf, tmp, and trash.

  You will need a valid Freesurfer license to proceed past this point. Get it from `http://surfer.nmr.mgh.harvard.edu/registration.html`.

- Build surfaces

  Data In: Data in MRI/(subject) directories

  Data Out: files in label, mri/*.mgz, files in `stats`, `surf`, `touch`

  Invokes the Freesurfer `recon-all` command. Also runs `mri_em_register`, `mri_ca_register`. This is a long one, 4 hours on my laptop.

- FS average

  Data In: Data in `MRI/surf`, others

  Data Out: `surf/*h.sphere` and `surf/*h.sphere.reg`

  – copies an "average" subject from FS software. Another input. Runs `mris_sphere`, `mris_register`, `mris_make_surfaces`, `mris_convert`, `mri_volmask`, `mris_anatomical_stats`, `mri_aparc2aseg`, `mri_binarize`, `mri_segstats`, `mri_label2label.bin`. Also quite long.

- Source space

  Data In: `surf/*h.sphere` and `surf/*h.sphere.reg`

  Data Out: `MRI/(subject)/bem`

  Takes cortical surface that Freesurfer extracted, makes a grid of 3-4mm dipoles, models for the brain activity. Hypothesizing electrical sources at each dipole. Still purely

MRI processed data. Create a tesselation of the surface and the nodes are the hypothesized dipoles. Invokes `mne_setup_source_space`.

- Setup coreg

  Data In: `MRI/(subject)/bem/*src.fif`

  Data Out: `MRI/(subject)/mri/brain-neuromag`

  Creates directories and prepares the ground. The coregistration step is where the MRI and MEG data get combined. This is not that step, but is somehow getting ready for it. Invokes `mne_setup_mri`.

- BE model

  Data In: `MRI/(subject)/mri/T1-neuromag/sets`

  Data Out: .surf files in `MRI/(subject)/bem`, and in `MRI/(subject)/bem/watershed`

  Boundary element model. Find boundary to brain, differentiate with skull. Invokes `mri_watershed`

- BE model to fif

  Data In: .surf files in `MRI/(subject)/bem`, and in `MRI/(subject)/bem/watershed`

  Data Out: *-bem.fif files in `MRI/(subject)/bem`

  Convert model to fif file, maybe more. Invokes `mne_prepare_bem_model`.

- Brain2mat

  Data In: `MRI/(subject)/surf`, `MRI/(subject)/label/*.annot`, `MNE/(subject)/*.fif`

  Data Out: `MRI/(subject)/brain.mat`, `MRI/avg-subject/brain-fsaverage.mat`

  Mostly rearranging. Fills out several variables and stores them in .mat files.

- Average surface

  Data In: files in `MRI/(subject)/surf/`

  Data Out: output in `MRI/(study)_ave`

  Process all the subjects in the study up to the Brain2Mat step before doing this one. This step averages all of them so you get one average subject. Uses `mris_make_template`, `mris_make_average_surface`, `mris_inflate`, `mri_surf2surf`, `mri_annotate`, `mri_vol2vol`, `mris_volmask`, others.

1.3.2 MEG Preprocessing

- Import

  Data In: raw fif files somewhere

Data Out: `MEG/(subject)/scans`

The raw fif files contain a time series for each sensor on a subject's scalp. This button imports a collection of raw fif files for a subject from somewhere specified via a GUI to a "MEG" directory for each subject. Also sets up a bunch of other directories: events, evoked, images, and scans_filtered.

- Extract events

  Data In: `MEG/(subject)/scans`

  Data Out: `MEG/(subject)/events/*.eve`

  Locate the stimulus in time. The events are encoded inside the raw fif file. Extracted and stored in a .eve file. Each trigger has a number.

- Process events

  Data In: `MEG/(subject)/events/*.eve`

  Data Out: `MEG/(subject)/events/*grouped.eve`

  DO NOT USE THIS. This triggers an experiment-specific script to scan the .eve files and pick up and group the events it finds in there. Unfortunately, the groupings are specific to the experiment, so attempts to create a general solution have not succeeded. Create your own groupings, and put the files in this directory when you're done so the next processes can be run. The GPS Cookbook and manual have advice on grouping.

- Bad channels

  Data In: text dialog

  Data Out: `MEG/(subject)/(subject)-bad-channels.txt`

  This just accepts a list of bad channels from a user dialog box and records their numbers in a text file. Before you do this, though, go to the 'mne browse raw' step in the utilities menu to note which channels are bad, write them down, and note them here. The channel numbers look like this: `EEG1234` and `MEG1234`.

- EOG projections – go back to utilities, use 'mne browse raw' to find eye blink events, make a fif file describing them, and specify the location of that file here. Look in scans, the word is "projection" see the file 'proj.fif'.

- Coregistration – Go to utilities, use 'mne analyze', see Cookbook, mapping the meg data to mri data. Creates a cor- fif file, maybe in the MRI directory. This step just identifies the file location and name.

1.3.3 MNE Analysis

- Average waves

  Data In: `MEG/(subject)/events/*grouped*`

MEG/(subject)/scans_filtered, also look in

Data Out: MNE/(subject)/blocks, and MNE/(subject)_ave.fif, subject_cov.fif

Creates event-related responses. Finds data within the time window defined by each event. Averaging across all trials for each individual subject. look in MNE/*.ave.fif, all kinds of other ave files, descriptions, the script, and so on. commands, blocks, higher level ave files, too. This step requires you to have already specified the event codes belonging to each condition, a step that must be done by hand.

- Forward solution

  Data In: MRI/(subject)/bem/*-bem.fif

  Data Out: MNE/(subject)/(subject)-fwd.fif

  Requires co-registration step from the MEG menu.

- Inverse solution

  Data In: MNE/(subject)/(subject)-fwd.fif

  Data Out: MNE/(subject)/(subject)_meg-inv.fif and MNE/(subject)/(subject)_meg_eeg-inv.fif

  Creates *.inv.fif files.

- Evoked trials

  Data In: MEG/(subject)/scans_filtered/*filtered_raw.fif

  Data Out: MEG/(subject)/evoked/*evoked_filtered.mat

  Saves data around each event in a .mat file.

- Make .stc

  Data In: MEG/(subject)/evoked/*evoked_filtered.mat

  Data Out: MNE/(subject)/stcs/*act-lh.stc and *act-rh.stc

  Source time code, putting MRI and MEG together. Tesselation of Freesurfer is very small triangles. So we decimate the grid here and impute the time series for each source point. Also generates some figures.

- Morphed stc

  Data In: MNE/(subject)/stcs

  Data Out: MNE/(subject)/stcs/*avebrain-lh.stc, etc

  Transforming the individual data onto the spherical surface, so they can be averaged in the next step.

- Average subject

  Data In: MNE/(subject)/stcs/*avebrain-lh.stc

  Data Out: MNE/(study)_ave

  Create the average subject from the collection of individuals. Who all must be complete before you get here. That is, complete all individuals before clicking on this button once.

1.3.4 Granger Analysis

- Process ROIs

  Data In: `Granger/(study)/rois/(condition)/(AVERAGEsubject)/*.label`, `MNE/(subject)/(subject)-fwd.fif`, `MRI/(subject)/*.annot`

  Data Out: `Granger/rois/(condition)`, subdirectories, `Granger/rois/(condition)/(subject)/(subject)_rois.mat`, `Granger/(study)/rois/(condition)/(subject)/*.label`,

  The ROIs for the average brain must already have been created for this step. The first step here is to generate the ROIs for the individual brains *from* the average brain, using `mne_morph_labels`. The label files for the average brain are created by the GPS:ROIs utility, in the 'Utilities' dialog. Look in `Granger/*/rois` directory. This step uses those ROIs to select comparable ROIs in the individual brains.

- MNI Coordinates

  Data In: `Granger/(subject)/rois/(condition)/*.label`

  Data Out: `Granger/rois/(study)_ave/mni_coordinates.txt`

  For each ROI, finds coordinates in a standardized brain.

- ROI Timecourses

  Data In: `MNE/(subject)/(subject)_meg_eeg-inv.fif`, `Granger/rois/(condition)/(subject)/(subject)_rois.mat`

  Data Out: `Granger/roiwaves/(condition)/(subject)*.mat`

  From the ROI, pick a representative vertex. Outputs a mat file containing a description and the time series for each identified ROI.

- Consolidate

  Data In: `Granger/roiwaves/(condition)/(subject)*.mat`

  Data Out: `Granger/input/(study)*.mat`

  The output here is a .mat file that contains a time series for each ROI and each subject, along with descriptions of the ROIs and the subjects.

- Compute Granger

  Data In: `Granger/input/(study)*.mat`

  Data Out: `Granger/results/raw`

  Computes all Granger possibilities. These are the conditions that *might* affect the output.

- Null hypothesis

  Data In: `Granger/results/raw`

  Data Out: `Granger/results/*.mat`, `Granger/results/nullhypothesis/*.mat`

  Cycles through the possibilities, recalculating the error terms for the N-1 set.

1.3.5 Utilities

- FS: TkMedit
- Visualize BEM
- mne_browse_raw

  Data In:

  Data Out: fif file, by default goes in `MEG/(subject)/scans`
- mne_analyze

  Data In: raw fif, also inflated brain

  Data Out: `MRI/(subject)/mri/T1-neuromag/sets`
- GPS: ROIs

  Data In: `MNE/(subject)/stcs/*act-lh.stc`

  Data Out: `Granger/rois/(AVERAGEsubject)/*.label`
- GPS: Plot – Used for making visualizations of the results.

2 Looking at the code

The analysis functions all seem to operate with two parameters, the state and a flag, a 't', a 'c', or a 'p'. These appear to stand for 'p' for 'progress', 'c' for 'compute', and 't' for, well I don't know, 'test'? The `gpsa_do()` function calls the 't' version, and that tells it something about whether the result is subject specific or condition specific. Then it calls the same function with a 'p', to see whether it needs to be computed at all, and if so, it calls the same function yet one more time with a 'c'.

This allows each function to set its own conditions on whether it runs or not, which is important with such a heterogeneous set of functions.