**Introduction to the One-step General Registration and Extraction (OGRE) Pipeline**

[**https://github.com/PhilipLab/OGRE-pipeline**](https://github.com/PhilipLab/OGRE-pipeline)

**Last update: October 9, 2023**

The goal of this project is to create a mac turnkey preprocessing pipeline with outputs immediately analyzable with FSL FEAT. If you have questions or suggestions to improve this document, please contact bphilip@wustl.edu.

This tutorial covers the following steps:

1. Creating a scanlist.csv
2. Converting dicoms to the nifiti file format with a scanlist.csv.
3. Converting the scanlist.csv into a pipeline compatible dat file.
4. Running the pipeline
5. Integrating pipeline outputs with FEAT

The pipeline itself has three parts. The first two of these are modified from Glasser’s HCP v3.27 pipelines (Glasser 2013 Neuroimage) and include the structural pipeline that uses the T1 and T2 (if available) and the functional pipeline that uses the BOLD data and associated SBRef images (if available) along with the field maps (if available). The third is an adapter that facilitates the use of the structural and functional pipeline outputs in the FSL FEAT statistical analysis. The pipelines must be run serially as they are dependent on one another’s outputs. The structural pipeline is run first, followed by the functional pipeline and finally the FEAT adapter pipeline.

**Creating a scanlist.csv**

The "scanlist.csv" links the dicom names to the nifti file outputs. Start from a sample and modify to match your data. The pipeline currently uses a T1 weighted image, a T2 weighted image (if available), spin echo field maps (if available, e.g. AP and PA) and any number of functional runs, ideally with a reference scan for each (i.e., SBRef). Examples should be available in github, and here is another example:

Scan,Series Desc,nii

7,t1\_mpr\_1mm\_p2\_pos50,t1\_mpr\_1mm\_p2\_pos50

8,SpinEchoFieldMap2\_AP,SpinEchoFieldMap2\_AP

9,SpinEchoFieldMap2\_PA,SpinEchoFieldMap2\_PA

10,CMRR\_fMRI\_TASK\_R1\_AP\_3mm\_488meas\_SBRef,run1\_RH\_SBRef

11,CMRR\_fMRI\_TASK\_R1\_AP\_3mm\_488meas,run1\_RH

13,CMRR\_fMRI\_TASK\_R2\_AP\_3mm\_488meas\_SBRef,run1\_LH\_SBRef

14,CMRR\_fMRI\_TASK\_R2\_AP\_3mm\_488meas,run1\_LH

24,CMRR\_fMRI\_TASK\_R3\_AP\_3mm\_488meas\_SBRef,run2\_RH\_SBRef

25,CMRR\_fMRI\_TASK\_R3\_AP\_3mm\_488meas,run2\_RH

27,CMRR\_fMRI\_TASK\_R4\_AP\_3mm\_488meas\_SBRef,run2\_LH\_SBRef

28,CMRR\_fMRI\_TASK\_R4\_AP\_3mm\_488meas,run2\_LH

38,CMRR\_fMRI\_TASK\_R5\_AP\_3mm\_488meas\_SBRef,run3\_RH\_SBRef

39,CMRR\_fMRI\_TASK\_R5\_AP\_3mm\_488meas,run3\_RH

44,CMRR\_fMRI\_TASK\_R6\_AP\_3mm\_488meas\_SBRef,run3\_LH\_SBRef

45,CMRR\_fMRI\_TASK\_R6\_AP\_3mm\_488meas,run3\_LH

57,CMRR\_fMRI\_REST\_R1\_AP\_3mm\_550meas\_SBRef,rest01\_SBRef

58,CMRR\_fMRI\_REST\_R1\_AP\_3mm\_550meas,rest01

60,CMRR\_fMRI\_REST\_R2\_AP\_3mm\_550meas\_SBRef,rest02\_SBRef

61,CMRR\_fMRI\_REST\_R2\_AP\_3mm\_550meas,rest02

63,CMRR\_fMRI\_REST\_R3\_AP\_3mm\_550meas\_SBRef,rest03\_SBRef

64,CMRR\_fMRI\_REST\_R3\_AP\_3mm\_550meas,rest03

66,t2\_spc\_sag\_p2\_iso\_1.0,t2\_spc\_sag\_p2\_iso\_1.0

Note that only the first and last fields (columns) are used, so if you prefer a little less documentation the following scanlist.csv would work just as well

Scan,nii

7,t1\_mpr\_1mm\_p2\_pos50

8,SpinEchoFieldMap2\_AP

9,SpinEchoFieldMap2\_PA

10,run1\_RH\_SBRef

11,run1\_RH

⁞

Fields must be comma separated, although the addition of spaces and tabs is ok.

**Convert dicoms to niftis**

This uses Chris Rorden’s dcm2niix to perform the dicom to nifti conversion. Basic usage is **/STUDYPATH/SUBJDIR/scanlist.csv** as the sole argument.

% **dcm2niix.sh** **/STUDYPATH/SUBJDIR/scanlist.csv**

or, if you run from **/STUDYPATH/SUBJDIR**

% **cd** **/STUDYPATH/SUBJDIR**

% **dcm2niix.sh** **scanlist.csv**

Dicoms are read from **/STUDYPATH/SUBJDIR/dicom**

Niftis are written to **/STUDYPATH/SUBJDIR/nifti**

A script is created and executed **/STUDYPATH/SUBJDIR/SUBJDIR\_dcm2niix.sh** via **/STUDYPATH/SUBJDIR/SUBJDIR\_dcm2niix\_fileout.sh**

Example

% **dcm2niix.sh** /Users/Shared/10\_Connectivity/10\_1001/10\_1001\_scanlist.csv

The dicom directory can be specified with -i

% **dcm2niix.sh** /Users/Shared/10\_Connectivity/10\_1001/10\_1001\_scanlist.csv -i/Users/Shared/10\_Connectivity/10\_1001/DICOM\_10\_1001\_220602

To see all options

% **dcm2niix.sh**

If it isn't creating nifti files, check **/STUDYPATH/SUBJDIR/SUBJDIR\_dcm2niix.sh.txt**. If it says "input folder invalid" this may be a permissions error, run chmod -R 775 or 777 on your subject directory.

**Conversion of scanlist.csv to dat file**

This script creates a "dat file" that contains parameters for the structural and functional pipeline setup scripts. Basic usage is **/STUDYPATH/SUBJDIR/scanlist.csv** as the sole argument.

% **scanlist2dat.py** **/STUDYPATH/SUBJDIR/scanlist.csv**

or, if you run from **/STUDYPATH/SUBJDIR**

% **cd** **/STUDYPATH/SUBJDIR**

% **scanlist2dat.py** **scanlist.csv**

Example

**% scanlist2dat.py** /Users/Shared/10\_Connectivity/10\_1001/10\_1001\_scanlist.csv

You may need to precede the script name with "python", e.g:

% python **scanlist2dat.py** **/STUDYPATH/SUBJDIR/scanlist.csv**

With options, much more complicated things can be done. Let’s say we need to convert scanlists for three subjects: 2002, 1019 and 2016. Note that 2002 has two scanlists.

% **scanlist2dat.py** -s /Users/Shared/10\_Connectivity /10\_2002/10\_2002a\_scanlist.csv /Users/Shared/10\_Connectivity /10\_2002/10\_2002b\_scanlist.csv \

-s /Users/Shared/10\_Connectivity /10\_1019/10\_1019\_scanlist.csv -s /Users/Shared/10\_Connectivity /10\_2016/10\_2016\_scanlist.csv \

-a /Users/Shared/10\_Connectivity /IHC4-19-20.dat

Four dat files are created: 1) /Users/Shared/10\_Connectivity/10\_2002/10\_2002.dat

2) /Users/Shared/10\_Connectivity/10\_1019/10\_1019.dat

3) /Users/Shared/10\_Connectivity/10\_2016/10\_2016.dat

4) /Users/Shared/10\_Connectivity/IHC4-19-20.dat

The fourth file includes 2002, 1019 and 2016. Note that the .dat file is created in the same location as the scan list.

To see all options

% **scanlist2dat.py**

**Structural pipeline**

The structural pipeline uses the T1 (required) and T2 (if available) to extract, segment and parcellate the brain along with the registration to the MNI atlas via FSL and Freesurfer (Glasser 2013 Neuroimage). Basic usage is **STUDYPATH/SUBDIR/dat** as the sole argument,

% OGREstrutpipeSETUPT.sh **STUDYPATH/SUBDIR/dat**

or if run from within **STUDYPATH/SUBDIR**,

% **cd** **STUDYPATH/SUBDIR**

% OGREstrutpipeSETUPT.sh **dat**

A script is created **STUDYPATH/SUBDIR/pipeline7.3.2/SUBDIR\_hcp3.27struct.sh** with its executable **/STUDYPATH/SUBDIR/pipeline7.3.2/SUBDIR\_hcp3.27struct\_fileout.sh**. The difference between the "struct.sh" and the "struct\_fileout.sh" is that the former will put its output to the command line, while the latter will put its output into a text file. The "struct\_fileout.sh" is not automatically executed unless the **-autorun** option is set; however, the structural pipeline can be run by simply executing from the command line

% **STUDYPATH/SUBDIR/pipeline7.3.2/SUBDIR\_hcp3.27struct\_fileout.sh**

By default scripts are created in **/STUDYPATH/SUBJDIR/pipeline7.3.2**, however this can be changed by editing the **OUTDIR** field in the **dat**.

One can also utilize a **batchscript** to set up executables for several subjects. For example,

% d0=/Users/Shared/10\_Connectivity

% s0=(${d0}/10\_2016/10\_2016.dat ${d0}/10\_2017/10\_2017.dat ${d0}/10\_2018/10\_2018.dat)

% **OGREstrutpipeSETUPT.sh**  ${s0[@]} -F 7.4.0 **-batchscript** /Users/Shared/10\_Connectivity/scripts/$(date +%y%m%d).sh

This creates subject scripts in **STUDYPATH/SUBDIR/pipeline7.4.0**, and an across-subject bash script in the location specified, which you then run to execute the structural pipeline. These "output scripts" will perform the structural analysis. Any previous results will be overwritten. You can run the subject-directory scripts individually via the "struct\_fileout.sh" scripts, or run the batch script which will run the participants serially (i.e., one at a time).

To see all options

% **OGREstrutpipeSETUPT.sh**

**Functional pipeline**

The functional pipeline uses the outputs of the structural pipeline along with the SBRef images (if available) and field maps (if available) to preprocess the BOLD time series. The implementation follows that of the structural pipeline with additional options for smoothing **-f**, high pass filtering **-p**, and first **-o** and second level **-t** feat analyses. For example,

% cd /Users/Shared/10\_Connectivity /10\_2000

% **OGREfMRIpipeSETUP.sh** 10\_2000.dat **-f** 4 6 **-p** 60 **-o** 10\_2000\_fsf1.txt **-t** 10\_2000/10\_2000\_fsf2.txt

Three output scripts are created: 1) 10\_2000\_hcp3.27 fMRIvol.sh

2) 10\_2000\_hcp3.27fMRIvol\_fileout.sh

3) 10\_2000\_FEATADAPTER.sh

The **-o** and **-t** options are used to run your FEAT analyses as part of a single step. Each one takes as input a text file with a list of .fsf files. (It should be flexible as to format; separate with commas, tabs, spaces, etc.) If you specify **-o**, the script will run the functional pipeline, run all your first level analyses, then run the FEATADAPTER (see below). If you also specify **-t**, it will then run a second-level analysis.

If you execute FEATADAPTER.sh, it will run just the "-o, -FEATADAPTER, -t" process. To recreate FEATADAPTER.sh without running any other analysis, use the **-F** option.

% **OGREfMRIpipeSETUP.sh** 10\_2000.dat **-o** 10\_2000\_fsf1.txt **-t** 10\_2000/10\_2000\_fsf2.txt **-F**

Do not execute the functional pipeline scripts until the structural pipeline has completed.

To see all options

% **OGREfMRIpipeSETUP.sh**

**FEAT first-level analysis**

With the output of the functional scripts, you can run first-level FEAT analysis. In the GUI, use the "statistics" option (top right pulldown), and use the following input file:

SUBJDIR/pipeline7.3.2/MNINonLinear/Results/RUNNAME/xyz where xyz is...  
 If no smoothing: RUNNAME.nii.gz  
 If using SUSAN smoothing (from -f option): RUNNAME\_SUSAN#mmHPTF#s.nii.gz

(#s are: FWHM smoothing from -f, high pass filter cutoff from -p)

To set yourself up for the next step, all your FEAT work should go in **/STUDYPATH/SUBJDIR/pipeline7.3.2/model**/

If you used the **-o** option in the functional analysis, it will automatically run FEAT using .fsf files you specified.

**FEAT adapter**

One additional step is needed between first-level analysis and higher-level analyses. The first-level results from the previous step will lack registration information, because FEAT does not (yet) know about all the registration work completed by the main pipeline. This script will perform the necessary registrations and fill in the missing files (i.e. create the "reg" directory inside the first-level FEAT output directory).

**% OGREmakeregdir.sh** <SUBDIR> <RUNNAME> <ANALYSISNAME>

This script is automatically created as part of the functional pipeline. If you wish to recreate the adapter script (without rerunning the functional pipeline), use the FEAT adapter **(-F)** option. This will create a SUB\_FEATADAPTER\_fileout.sh and SUB\_FEATADAPTER.sh; run either of those (as always, fileout doesn't lock up your terminal window).

% **OGREfMRIpipe.sh** <DAT> **-o** <FIRST LEVEL FSL.txt> **-F**

To see all options

% **OGREmakeregdir.sh**

As normal with the functional script, if you use the -A option then it will automatically run, otherwise it creates scripts that you can run.

**FEAT higher-level analysis**

Now you can run a higher-level analysis on your first-level FEAT outputs. (Or maybe you already did with **-t**.) Go forth and get your results!