**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: June 11, 2024**

The goal of this project is to create an off-the-shelf mac preprocessing pipeline (with registration and motion correction combined via one-step resampling) with outputs immediately analyzable with FSL FEAT. If you have questions or suggestions to improve this document, please contact [bphilip@wustl.edu](mailto:bphilip@wustl.edu).

OGRE workflow and table of contents:

1. [Create required files](#Create_Required_Files)
   1. [One “scanlist.csv” for each subject](#Create_Scanlist), to describe & name the scans
   2. [*OPTIONAL*: One “protocol.csv” for each study](#Create_Protocol), to aid automatic generation of scanlist.csv
   3. [.fsf files](#Create_FSF), for FSL FEAT analyses
   4. [*OPTIONAL*: one locator.txt file for each subject](#Create_Locator), to specify location of their .fsf files
2. [Convert dicom to nifti files](#DCM2NIIX)
3. [Review pipeline operation common principles](#PipelinePrinciples)
4. [Run structural pipeline](#PipeStruct) via **OGREstructpipeSETUP.sh**
5. [Run functional pipeline](#PipeFunc) via **OGREfMRIpipeSETUP.py**

*Note: Steps 6-8 will happen automatically as part of step 5 if you specified your FEAT analyses in step 1D*

1. [Perform your FEAT first-level analyses](#FEAT_L1)
2. [Use the FEAT adapter](#FEAT_Adapter) (**<sub>\_FEATADAPTER.sh**) to generate the “reg” subdirectory in your .feat directories
3. [Perform your FEAT higher-level analyses](#FEAT_L2)
4. [Cleanup (optional)](#Cleanup)
5. [List of OGRE outputs](#Outputs)

# 1) Creating required files

OGRE uses csv files to describe which scans were performed in which order (see sections 1A, 1B).

You will also need to create fsf files for your FSL analysis (section 1C), and optionally specify these fsf files' locations (section 1D).

To maintain flexibility, file names are never required to be any particular pattern; instead, OGRE learns their names when you pass the names as inputs to OGRE scripts. We recommend filenames that are recognizable to BIDS format (e.g. “sub-XXXX\_scanlist.csv”).

Future versions of OGRE will be able to draw this information from BIDS .json files.

## 1A. Create scanlist.csv

The “scanlist.csv” links the dicom names to nifti file outputs, and also is used by the rest of OGRE to identify file names and locations. You can create “scanlist.csv” manually.

An example is in **OGRE-pipeline/examples/sub-2025\_scanlist.csv**, and reproduced here:

7,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/anat/sub-2025\_T1w

8,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_dir-AP\_epi

9,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_dir-PA\_epi

10,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-1\_sbref

11,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-1\_bold

13,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-1\_sbref

14,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-1\_bold

19,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-2\_sbref

20,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-2\_bold

22,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-2\_sbref

23,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-2\_bold

31,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_run-3\_dir-AP\_epi

32,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_run-3\_dir-PA\_epi

33,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-3\_sbref

34,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-3\_bold

36,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-3\_sbref

37,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-3\_bold

39,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-dbsi\_dir-PA\_epi

40,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/dwi/sub-2025\_acq-dbsi\_dwi

41,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-1\_sbref

42,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-1\_bold

44,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-2\_sbref

45,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-2\_bold

47,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-3\_sbref

48,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-3\_bold

50,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/anat/sub-2025\_T2w

In this example, the dicom names were just single numbers (e.g. “7”). If multiple field maps exist (highlighted), OGRE will apply the most recent one to a given func scan.

* Future versions of OGRE will use BIDS metadata, if available, to determine which fieldmap applies to which scan. This will supersede the “most recent” default behavior.

For automatic generation of scanlists, we have included **pdf2scanlist.py** which is designed to take a CNDA screenshot pdf and compare it to the “protocol.csv” (see below) to create a “scanlist.csv”.

**% pdf2scanlist.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_CNDA.pdf -p /Users/mcavoy/repo/NRL-misc/10\_Connectivity\_protocol.csv**

When using pdf2scanlist.py, review your pdf’s and scanlists to ensure that no scans have been omitted (e.g. due to page breaks).

## 1B. Create protocol.csv (OPTIONAL)

The “protocol.csv“ defines the data that was acquired in the scanner, and is only required if you intend to use **pdf2scanlist.py**. If you are creating your scanlist.csv files manually, “protocol.csv” is not needed.

Consider the example for study “10\_Connectivity“:

t1\_mpr\_1mm\_p2\_pos50, overwrite, anat, T1w

SpinEchoFieldMap2\_AP, append, fmap, dir-AP\_epi

SpinEchoFieldMap2\_PA, append, fmap, dir-PA\_epi

CMRR\_fMRI\_TASK\_R1\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawRH\_run-1\_sbref

CMRR\_fMRI\_TASK\_R1\_AP\_3mm\_488meas, overwrite, func, task-drawRH\_run-1\_bold

CMRR\_fMRI\_TASK\_R2\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawLH\_run-1\_sbref

CMRR\_fMRI\_TASK\_R2\_AP\_3mm\_488meas, overwrite, func, task-drawLH\_run-1\_bold

CMRR\_fMRI\_TASK\_R3\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawRH\_run-2\_sbref

CMRR\_fMRI\_TASK\_R3\_AP\_3mm\_488meas, overwrite, func, task-drawRH\_run-2\_bold

CMRR\_fMRI\_TASK\_R4\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawLH\_run-2\_sbref

CMRR\_fMRI\_TASK\_R4\_AP\_3mm\_488meas, overwrite, func, task-drawLH\_run-2\_bold

CMRR\_fMRI\_TASK\_R5\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawRH\_run-3\_sbref

CMRR\_fMRI\_TASK\_R5\_AP\_3mm\_488meas, overwrite, func, task-drawRH\_run-3\_bold

CMRR\_fMRI\_TASK\_R6\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawLH\_run-3\_sbref

CMRR\_fMRI\_TASK\_R6\_AP\_3mm\_488meas, overwrite, func, task-drawLH\_run-3\_bold

ep2ddbsi\_b0\_PE=PA, overwrite, fmap, acq-dbsi\_dir-PA\_epi

ep2ddbsi\_19\_2mm\_iso\_LowBW\_2av, overwrite, dwi, acq-dbsi\_dwi

CMRR\_fMRI\_REST\_R1\_AP\_3mm\_550meas\_SBRef, overwrite, func, task-rest\_run-1\_sbref

CMRR\_fMRI\_REST\_R1\_AP\_3mm\_550meas, overwrite, func, task-rest\_run-1\_bold

CMRR\_fMRI\_REST\_R2\_AP\_3mm\_550meas\_SBRef, overwrite, func, task-rest\_run-2\_sbref

CMRR\_fMRI\_REST\_R2\_AP\_3mm\_550meas, overwrite, func, task-rest\_run-2\_bold

CMRR\_fMRI\_REST\_R3\_AP\_3mm\_550meas\_SBRef, overwrite, func, task-rest\_run-3\_sbref

CMRR\_fMRI\_REST\_R3\_AP\_3mm\_550meas, overwrite, func, task-rest\_run-3\_bold

t2\_spc\_sag\_p2\_iso1.0, overwrite, anat, T2w

Each item contains 4 fields:

1. Scan name. As received from scanner.
2. Write mode: can be *append* or *overwrite*
   1. *Append*: retains all scans with this name. See highlighted lines in examples.
   2. *Overwrite*: if there are multiple scans with this name, keeps last one only.
3. Bids directory: specify a subdirectory for BIDS compliance. E.g. *anat*, *fmap*, *func*, *dwi*
4. Output root: add this to end of filename to produce BIDS-compliant nifti

## 1C. Create .fsf files for FSL FEAT analysis

This document assumes you are already familiar with FSL. FSL documentation is available [here](https://fsl.fmrib.ox.ac.uk/fsl/docs/#/).

To use FSL FEAT on data preprocessed by OGRE, you will use FEAT's “statistics” option (rather than “full analysis”), to skip preprocessing and registration.

Confirm in your first-level .fsf files: “set fmri(analysis) 2”

In addition, even if you tell FEAT to omit preprocessing, FEAT will run SUSAN spatial smoothing (which is ostensibly part of preprocessing). Therefore, you must also set the spatial smoothing kernel to 0 in your first-level analyses.

Confirm in your first-level .fsf files: “set fmri(smooth) 0”

*Alternatively, you could perform your spatial smoothing (if any) in FEAT instead of OGRE. We find that logistically cumbersome, but it's functionally equivalent. The critical part is to avoid accidentally smoothing your data more times than you intended!*

You will also need to select the appropriate OGRE output as your input to first-level FEAT. The correct file is:

If no smoothing: **sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_bold.nii.gz**  
 If using SUSAN smoothing (from -f option): **sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_susan-#mm\_hptf-%s\_bold.nii.gz**

*(Where # is your SUSAN smoothing FWHM, and % is your temporal cutoff)*

Otherwise, create your .fsf files as normal for your study design. Example first- and second-level .fsf's are included in OGRE-pipeline/examples.

This step does not need to be performed prior to running the pipeline; it gets called during or after the functional pipeline.

## 1D. Create locator.txt files to let OGRE identify your .fsf files (OPTIONAL)

These files will allow OGRE to automatically run your first- and second-level FSL analyses (and the necessary adapter). They will make your life easier, but they are not necessary.

For each subject, create a .txt file with the full paths of all first-level analyses. The paths can be separated by commas, spaces, tabs and/or newlines. See **OGRE-pipeline/examples/sub-2025\_locatorOne.txt**

If you want OGRE to also autorun your second-level analyses, create a second separate .txt file with their paths. Same format; see **OGRE-pipeline/examples/sub-2025\_locatorTwo.txt**

# 2. Convert dicoms to niftis

OGREdcm2niix.sh uses Chris Rorden’s dcm2niix to perform the dicom to nifti conversion automatically on an entire scanlist.csv.

**% OGREdcm2niix.sh /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv \**

**-i /Volumes/NRLbackup/10\_Connectivity/dicom/sub-2025**

In the above example, the “scanlist.csv“ created in section 1A is used to convert the dicoms located at **/Volumes/NRLbackup/10\_Connectivity/dicom/sub-2025** as specified by the -i option.

If nifti files are not created, check **/Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_dcm2niix.sh.txt**. If it says “input folder invalid“ this may be a permissions error, run chmod -R 775 or 777 on the subject directory.

**% chmod -R 775 /Users/Shared/10\_Connectivity/raw\_data/sub-2025**

Note that this step is “required” in the sense that nifti files are required for OGRE. You may be able to use preexisting nifti files, though OGREdcm2niix.sh ensures the creation of .json files with appropriate metadata.

# 3. Pipeline operation common principles

The main OGRE scripts are “OGRExxxSETUP” scripts that will take in a scanlist.csv and other arguments to create subject-specific .sh files.

Subject-specific scripts will be produced alongside with a “\_fileout.sh” variant. The difference is that the “\_fileout.sh” version will send its output to a text file instead of the command line.

To automatically execute the \_fileout.sh when you create it via a SETUP script, use the -A argument when calling the SETUP script.

A convenient way to set up the structural and functional pipelines together is via the -b “batchscript” option:

**% OGREstructpipeSETUP.sh /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv -b**

**% OGREfMRIpipeSETUP.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv [arguments] -b**

This will create:

**% /Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREbatch\_fileout.sh**

which is identical to running the subject-specific struct script, followed by the subject-specific func script.

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

# 4. Structural pipeline

The structural pipeline is modified from Glasser's HCP v3.27 pipelines (Glasser 2013 Neuroimage), and 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.

Basic usage is with “scanlist.csv“ as the sole argument,

**% export OGREDIR=/path/to/OGRE-pipeline**

**% OGREstructpipeSETUP.sh /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv**

This will create the scripts

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREstruct.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREstruct\_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 -A “-autorun“ argument is set; however, the structural pipeline can be run by simply executing from the “struct\_fileout.sh“ from the command line.

# 5. 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 spatial smoothing **-f**, high pass filtering **-p**, and first-level **-o** and second-level **-t** feat analyses. For example,

**% pipedir=/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1**

**% OGREfMRIpipeSETUP.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv -f 6 -p 60 -o ${pipedir}/sub-2025\_locatorOne.txt -t {pipedir}/sub-2025\_locatorTwo.txt**

This will create the scripts

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREfMRIvol.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREfMRIvol\_fileout.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_bidscp.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_FEATADAPTER.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_cleanup.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 “locator.txt” file described in previous section. If you specify **-o**, the script will run the functional pipeline, copy the outputs to bids directories, smooth and filter, run all your first level FSL analyses, then run the FEATADAPTER (see below). If you also specify **-t**, it will then run your second-level analyses.

Normally you will not need to invoke “bidscp.sh” or “FEATADAPTER.sh”, these are both automatically executed by “fMRIvol.sh.”

“bidscp.sh“ copies the OGRE pipeline outputs in “pipedir“ to the appropriate BIDS directories, e.g.

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/anat**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/func**

Spatial smoothing (**-f**) and temporal filtering (**-p**), if any, are applied to these final BIDS outputs.

The sub-XXXX\_cleanup.sh script is described under 10. Cleanup.

# 6. FEAT first-level analysis

With the output of the functional scripts, you are ready to run first-level FEAT analysis (as described in section 2D). The OGRE output to use as an input to FEAT is:

/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-XXXX/func/xyz where xyz is, for example:  
 If no smoothing: **sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_bold.nii.gz**  
 If using SUSAN smoothing (from -f option): **sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_susan-#mm\_hptf-%s\_bold.nii.gz**

*(Where # is your SUSAN smoothing FWHM, and % is your temporal cutoff)*

To set yourself up for the next step, all your FEAT work should go in, for example:

**/Users/Shared/10\_Connectivity/derivatives/analysis/sub-2025/sub-2025\_model-OGRE**

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

# 7. FEAT adapter

One additional step is needed between first-level FEAT analysis and higher-level FEAT analyses. The first-level results will lack registration information, because FEAT does not (yet) know about all the registration work completed by OGRE. The “FEATADAPTER.sh” script will copy the appropriate transformations and matrices to create a “reg“ directory inside each first-level FEAT output directories. All of this is automatically performed by “fMRIvol.sh” if you set -o, but you have the power to run it later or manually.

% **sub-2025\_FEATADAPTER.sh**

As noted above, FEATADAPTER.sh script is automatically created as part of the functional pipeline. To recreate FEATADAPTER.sh without running any other analysis, use the **-F** option:

**% OGREfMRIpipeSETUP.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv -o ${pipedir}/sub-2025\_locatorOne.txt -t {pipedir}/sub-2025\_locatorTwo.txt -F**

This will create a sub-XXXX\_FEATADAPTER\_fileout.sh and sub-XXXX\_FEATADAPTER.sh; run either of those (as always, fileout doesn't lock up your terminal window).

You may find it useful to run the adapter manually on individual .feat directories. The script that does the work here is **OGRE-pipeline/lib/OGREmakeregdir.sh**, which is documented and ready for independent use on an individual .feat directory:

% **OGREmakeregdir.sh -f sub-2025\_task-drawLH\_run-1.feat**

This is the same as running FEATADAPTER on a list of directories. All FEATADAPTER does is run **OGREmakeregdir.sh** on each .feat directory defined within the .fsf's in your locator.txt files.

# 8. 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!

# 9. Cleanup

If you are satisfied that you know which files you need, you can save disk space by deleting everything except the final BIDS-format outputs via "sub-XXXX\_cleanup.sh", e.g.:

% **/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_cleanup.sh**

will delete all OGRE subdirectories in the parent directory **derivatives/preprocessed/sub-XXXX/pipeline7.4.1**. Individual files in the parent directory and non-OGRE subdirectories are retained.

The only reason to retain pipeline7.4.1 files is if you want to use them in your analyses. (If so, you should of course copy out any useful files before you run “cleanup.sh”). The files in pipeline7.4.1 will not help with future processing. If you need to rerun a struct script, you are starting that analysis from scratch and should delete all existing outputs first. If you need to rerun a func script, it will overwrite any previous func outputs.

# 10. OGRE outputs

OGRE's primary outputs are BIDS-compatible, created when *sub-XXXX\_bidscp.sh* copies the OGRE pipeline's outputs from the *pipeline7.4.1* working directory to the bids output directories (specified originally in scanlist.csv and maintained in raw\_data).

Let *parent* = derivatives/preprocessed/sub-XXXX

Let *runname* = everything between the subject ID and the suffix in scanlist.csv (e.g. “task-X\_run-Y”)

BOLDs:

From *parent*/pipeline7.4.1/MNINonLinear/Results*/runname*/sub-XXXX\_<run name>\_bold.nii.gz

To *parent*/func/sub-XXXX *runname*\_OGRE-preproc\_bold.nii.gz

Brain masks:

From *parent*/pipeline7.4.1/MNINonLinear/Results/<run name>/brainmask\_fs.2.nii.gz

To *parent*/func/sub-XXXX\_*runname*\_OGRE-preproc\_res-2\_label-brain\_mask.nii.gz

Structurals:

T1 and T2 weighted whole head images

From *parent*/pipeline7.4.1/MNINonLinear/TXw\_restore.nii.gz

To *parent*/anat/sub-XXXX\_OGRE-preproc\_desc-restore\_TXw.nii.gz

T1 and T2 weighted brain-only images *(not BIDS-compliant, instead in filename format required by FSL)*

From *parent*/pipeline7.4.1/MNINonLinear/TXw\_restore\_brain.nii.gz

To *parent*/anat/sub-XXXX\_OGRE-preproc\_desc-restore\_TXw\_brain.nii.gz

Future versions of the OGRE manual will detail the files in the pipeline7.4.1 working directory.