

BROCCOLI

-

**Software for Fast fMRI Analysis
on Many-Core CPUs and GPUs**

Anders Eklund

2014-11-28

Table of Contents

1	Introduction	3
1.1	Introduction	3
1.2	Downloading BROCCOLI	3
1.3	Installing OpenCL	3
1.4	Running BROCCOLI from a terminal	4
1.5	Compiling the bash wrappers	4
1.6	Compiling the BROCCOLI library	4
1.7	Checking the OpenCL drivers	5
1.8	Selecting the OpenCL platform and device	8
1.9	Compiling the OpenCL kernel code	8
2	First level analysis	9
2.1	Introduction	9
2.2	OpenCL options	10
2.3	Registration options	11
2.4	Preprocessing options	11
2.5	Statistical options	11
2.6	Outputs	13
2.7	Output options	13
2.8	Additional options	15
2.9	Checking the registration	16
3	Registration	17
3.1	Introduction	17
3.2	OpenCL options	17
3.3	Registration options	17
3.4	Outputs	18
3.5	Output options	18
3.6	Additional options	19
4	Transformation	21
4.1	Introduction	21
4.2	OpenCL options	21
4.3	Transformation options	22
4.4	Outputs	22
4.5	Output options	22
4.6	Additional options	22

5	Motion correction	23
5.1	Introduction	23
5.2	OpenCL options	23
5.3	Motion correction options	23
5.4	Outputs	24
5.5	Output options	24
5.6	Additional options	24
6	Permutation testing	25
6.1	Randomise	25
6.2	OpenCL options	26
6.3	Permutation options	26
6.4	Outputs	27
6.5	Output options	27
6.6	Additional options	28

Introduction

1.1 Introduction

BROCCOLI is a software mainly created for analysis of functional magnetic resonance imaging (fMRI) data. The major advantage of BROCCOLI, compared to other software packages, is that it is much faster. The main reason for this is that BROCCOLI is written in OpenCL (Open Computing Language), making it possible to run the analysis in parallel on a large variety of hardware platforms (such as CPUs, Nvidia GPUs and AMD GPUs).

1.2 Downloading BROCCOLI

BROCCOLI is available as open source at <https://github.com/wanderine/BROCCOLI/>. For Linux and Mac computers, the following git command can be used to download BROCCOLI into the folder BROCCOLI.

```
git clone https://github.com/wanderine/BROCCOLI.git BROCCOLI
```

For Windows computers, see <https://windows.github.com/> or download BROCCOLI as a zip file from github.

1.3 Installing OpenCL

BROCCOLI requires that at least one OpenCL driver is installed. All hardware platforms (e.g. Intel, AMD and Nvidia) require a specific OpenCL driver. It is possible to install several OpenCL drivers on a single machine.

Intel drivers are currently located at <https://software.intel.com/en-us/articles/opencl-drivers>.

AMD drivers can be found at <http://developer.amd.com/tools-and-sdks/opencl-zone/opencl-tools-sdks/amd-accelerated-parallel-processing-app-sdk/>.

Nvidia drivers can be found at <http://www.nvidia.com/Download/index.aspx>. Note that a driver for an Nvidia graphics cards includes an OpenCL driver.

1.4 Running BROCCOLI from a terminal

It is possible to run BROCCOLI in a number of ways, for example from a Matlab terminal or from a Python terminal. The fMRI software packages AFNI and FSL can be launched from a Linux terminal, and BROCCOLI includes a bash wrapper for this purpose.

1.5 Compiling the bash wrappers

The bash wrapper functions first need to be compiled, and the following command can be used from the folder BROCCOLI/code/Bash_Wrapper

```
./compile_wrappers.sh
```

Similarly, the following command can be used for Mac

```
./compile_wrappers_mac.sh
```

Since BROCCOLI uses the Nifti library to read Nifti files, it may be necessary to first compile the Nifti library, by running Make in the folder BROCCOLI/code/Bash_Wrapper/nifticlib-2.0.0.

1.6 Compiling the BROCCOLI library

BROCCOLI is written as a C++/OpenCL library, such that it can be linked to a number of softwares. To compile the BROCCOLI library on a Linux computer, the following command can be used from the folder BROCCOLI/code/BROCCOLI_LIB (where BROCCOLI denotes where you saved BROCCOLI from github).

```
./compile_broccoli_library.sh
```

Note that you may need to edit compile_broccoli_library.sh to point to a directory which contains opencl.h. For Mac, a similar bash script exists

```
./compile_broccoli_library_mac.sh
```

On Windows computers, the BROCCOLI library can for example be compiled using Microsoft Visual Studio. Visual Studio project files are available in BROCCOLI/code/BROCCOLI_LIB.

1.7 Checking the OpenCL drivers

To check the installation of the OpenCL driver(s), the function `GetOpenCLInfo` can be used. It lists all the available devices for each OpenCL platform. The following output is from a computer with three OpenCL drivers (Intel, AMD, Nvidia) and one Intel CPU (with 32 GB of memory), one AMD graphics card (3 GB of memory) and one Nvidia graphics card (6 GB of memory). For each bash function, it is easy to select which platform and which device to use. Note that the Intel CPU can be used with the Intel platform or the AMD platform.

```
[andek@d11-a10003 Bash_Wrapper]$ ./GetOpenCLInfo
Device info
```

```
-----
Platform number: 0
-----
```

```
Platform vendor: Intel(R) Corporation
Platform name: Intel(R) OpenCL
Platform extensions: cl_khr_fp64 cl_khr_icd cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics cl_khr_byte_addressable_store
cl_intel_printf cl_ext_device_fission cl_intel_exec_by_local_thread
Platform profile: FULL_PROFILE
-----
```

```
-----
Device number: 0
-----
```

```
Device vendor: Intel(R) Corporation
Device name:      Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz
Hardware version: OpenCL 1.2 (Build 67279)
Software version: 1.2
OpenCL C version: OpenCL C 1.2
Device extensions: cl_khr_fp64 cl_khr_icd cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics cl_khr_byte_addressable_store
cl_intel_printf cl_ext_device_fission cl_intel_exec_by_local_thread
Global memory size in MB: 32071
Global memory cache size in KB: 256
Local memory size in KB: 32
Constant memory size in KB: 128
Parallel compute units: 8
```


Clock frequency in MHz: 3500
Max number of threads per block: 1024
Max number of threads in each dimension: 1024 1024 1024

Platform number: 1

Platform vendor: Advanced Micro Devices, Inc.
Platform name: AMD Accelerated Parallel Processing
Platform extensions: cl_khr_icd cl_amd_event_callback cl_amd_offline_devices
Platform profile: FULL_PROFILE

Device number: 0

Device vendor: Advanced Micro Devices, Inc.
Device name: Tahiti
Hardware version: OpenCL 1.2 AMD-APP (1214.3)
Software version: 1214.3 (VM)
OpenCL C version: OpenCL C 1.2
Device extensions: cl_khr_fp64 cl_amd_fp64 cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_3d_image_writes
cl_khr_byte_addressable_store cl_khr_gl_sharing cl_ext_atomic_counters_32
cl_amd_device_attribute_query cl_amd_vec3 cl_amd_printf
cl_amd_media_ops cl_amd_media_ops2 cl_amd_popcnt cl_khr_image2d_from_buffer
Global memory size in MB: 3035
Global memory cache size in KB: 16
Local memory size in KB: 32
Constant memory size in KB: 64
Parallel compute units: 32
Clock frequency in MHz: 1000
Max number of threads per block: 256
Max number of threads in each dimension: 256 256 256

Device number: 1

Device vendor: GenuineIntel
Device name: Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz
Hardware version: OpenCL 1.2 AMD-APP (1214.3)
Software version: 1214.3 (sse2,avx)

```
OpenCL C version: OpenCL C 1.2
Device extensions: cl_khr_fp64 cl_amd_fp64 cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_3d_image_writes
cl_khr_byte_addressable_store cl_khr_gl_sharing cl_ext_device_fission
cl_amd_device_attribute_query cl_amd_vec3 cl_amd_printf cl_amd_media_ops
cl_amd_media_ops2 cl_amd_popcnt
Global memory size in MB: 32071
Global memory cache size in KB: 32
Local memory size in KB: 32
Constant memory size in KB: 64
Parallel compute units: 8
Clock frequency in MHz: 1600
Max number of threads per block: 1024
Max number of threads in each dimension: 1024 1024 1024
```

```
-----
Platform number: 2
-----
```

```
Platform vendor: NVIDIA Corporation
Platform name: NVIDIA CUDA
Platform extensions: cl_khr_byte_addressable_store cl_khr_icd
cl_khr_gl_sharing cl_nv_compiler_options cl_nv_device_attribute_query
cl_nv_pragma_unroll
Platform profile: FULL_PROFILE
-----
```

```
-----
Device number: 0
-----
```

```
Device vendor: NVIDIA Corporation
Device name: GeForce GTX TITAN
Hardware version: OpenCL 1.1 CUDA
Software version: 331.67
OpenCL C version: OpenCL C 1.1
Device extensions: cl_khr_byte_addressable_store cl_khr_icd cl_khr_gl_sharing
cl_nv_compiler_options cl_nv_device_attribute_query cl_nv_pragma_unroll
cl_khr_global_int32_base_atomics cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics cl_khr_local_int32_extended_atomics cl_khr_fp64
Global memory size in MB: 6143
Global memory cache size in KB: 224
Local memory size in KB: 48
Constant memory size in KB: 64
```

Parallel compute units: 14
Clock frequency in MHz: 875
Max number of threads per block: 1024
Max number of threads in each dimension: 1024 1024 64

1.8 Selecting the OpenCL platform and device

As mentioned previously, it is easy to select which OpenCL platform and device to use for each BROCCOLI function (the default platform and device is 0). For each bash function, the option `-platform` sets the OpenCL platform to use, and the option `-device` sets the device to use. The first level analysis can, for example, be performed using platform 1 and device 1 by using the command

```
./FirstLevelAnalysis fMRI.nii T1.nii BrainTemplate.nii ...  
regressors.txt contrasts.txt -platform 1 -device 1
```

For the test computer used in the introduction, this means that the analysis runs on the Intel CPU using the AMD platform. By changing the device to 0, the AMD graphics card will instead be used.

1.9 Compiling the OpenCL kernel code

The file `broccoli_lib_kernel.cpp` contains all the OpenCL code that runs on the selected device. The first time BROCCOLI uses a specific device, it has to compile all the OpenCL code to a form that can be used by the device. For this reason, the first function call will take more time. The compiled code will however be saved as a binary file, such that BROCCOLI can simply read the compiled code for all subsequent function calls. For our specific test computer, BROCCOLI will produce the following binary files

```
broccoli_lib_kernel_Intel_Intel(R)Core(TM)i7-3770KCPU@3.50GHz.bin  
broccoli_lib_kernel_AMD_Intel(R)Core(TM)i7-3770KCPU@3.50GHz.bin  
broccoli_lib_kernel_AMD_Tahiti.bin  
broccoli_lib_kernel_Nvidia_GeForceGTXITITAN.bin
```

First level analysis

2.1 Introduction

BROCCOLI performs first level fMRI analysis using a single command. The analysis involves registering the anatomical volume to a brain template (using linear as well as non-linear registration), registering one fMRI volume to the anatomical volume, slice timing correction, motion correction, smoothing and statistical analysis. Contrary to other software packages, it is up to the user to decide which intermediate results to save. In its simplest form, a first level analysis can with the bash wrapper be performed as

```
./FirstLevelAnalysis fMRI.nii T1_brain.nii BrainTemplate.nii ...  
regressors.txt contrasts.txt
```

fMRI.nii is here the 4D fMRI data, T1_brain.nii is the skullstripped anatomical volume of the subject and BrainTemplate.nii is for example the MNI template in the FSL software (without skull). Currently it is necessary that all the NIfTI files are stored in the same orientation (e.g. RPI). The orientation can for example be checked with the AFNI function 3dinfo, and the orientation can be changed using the function 3dresample (e.g. 3dresample -orient RPI -input volume.nii -prefix volume_RPI.nii).

regressors.txt is a text file that contains the number of regressors to use, and the filename of each regressor. It can for example look like

```
NumRegressors 3
```

```
task1.txt  
task2.txt  
task3.txt
```

Each task file needs to include the number of events, the start time of each event, the length of each event and the value for the regressor of each event

(this format is very similar to the one used by the FSL software). A task file can for example look like

```
NumEvents 8

47.532543      2.517515      1
68.789386      2.950905      1
93.547212      2.934332      1
165.670213     3.584591      1
190.078151     1.817203      1
257.349564     2.584177      1
280.457018     1.367038      1
308.66593      2.084071      1
```

contrasts.txt contains a list of all the contrasts for which BROCCOLI will calculate a volume with t-scores. An example is given by

```
NumRegressors 3
NumContrasts 9

1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
1.0 -1.0 0.0
-1.0 1.0 0.0
1.0 0.0 -1.0
0.0 1.0 -1.0
-1.0 0.0 1.0
0.0 -1.0 1.0
```

2.2 OpenCL options

The following OpenCL options are available

- -platform

The OpenCL platform to use (default 0).

- -device

The OpenCL device to use (default 0).

2.3 Registration options

Several options can be used to control the image registration, see the chapter about image registration for further information.

2.4 Preprocessing options

The following preprocessing options are available

- -slicepattern

Set the sampling pattern used during scanning

(overrides pattern provided in NIFTI file)

0 = sequential 1-N (bottom-up), 1 = sequential N-1 (top-down),

2 = interleaved 1-N, 3 = interleaved N-1

No slice timing correction is performed if pattern in

NIFTI file is unknown and no pattern is provided.

- -iterationsmc

Set the number of iterations used for the

motion correction (the default is 5).

- -smoothing

Set the amount of smoothing applied to

each fMRI volume (the default is 6 mm FWHM).

2.5 Statistical options

The following statistical options are available

- -rawregressors

Use raw regressors (FSL format, one line per volume/TR). These regressors will not be convolved with any hemodynamic response function.

- -regressmotion

Add the 6 estimated motion parameters (3 for translation and 3 for rotation) to the design matrix, to further suppress the effect of head motion. The contrast vectors are automatically extended with zeros for these additional regressors.

- `-regressglobalmean`

Include a regressor for the global mean in the design matrix. The contrast vectors are automatically extended with one zero for this additional regressor.

- `-temporalderivatives`

Add one additional regressor for each original regressor; the temporal derivative of each regressor. This makes it possible to adjust for a small time difference between the original regressor and the time series in each voxel. The contrast vectors are automatically extended with zeros for these additional regressors.

- `-permute`

Run a permutation test after the conventional first level analysis. In each permutation, a random reshuffling of the fMRI volumes is performed to create a new set of null data. The statistical analysis is performed in each permutation, to empirically estimate the null distribution.

- `-inferencemode`

Set if voxel-wise or cluster-wise p-values should be calculated for the permutation test, 0 = voxel, 1 = cluster extent, 2 = cluster mass, 3 = threshold free cluster enhancement (TFCE) (default 1)

- `-cdt`

Cluster defining threshold for permutation based cluster inference (default 2.5).

- `-permutations`

Set the number of permutations (default 1,000)

- `-bayesian`

Run Bayesian first level analysis. Note that this option currently only works for 2 regressors. In each voxel, a Gibbs sampler is used to estimate the posterior probability of each contrast being larger than 0. This option cannot be combined with `-permute`.

- `-iterationsmcmc`

Set the number of MCMC iterations to be used (default 1,000).

2.6 Outputs

The first level analysis results in beta weights, contrast estimates, t-scores and p-values (if the option -permute is used). For an analysis with 2 regressors, a total of 6 regressors will be used (the 2 original regressors and 4 detrending regressors). The beta weights for these regressors will be saved in the brain template space as

```
fMRI_beta_regressor1_MNI.nii  
fMRI_beta_regressor2_MNI.nii  
fMRI_beta_regressor3_MNI.nii  
fMRI_beta_regressor4_MNI.nii  
fMRI_beta_regressor5_MNI.nii  
fMRI_beta_regressor6_MNI.nii
```

For each contrast, BROCCOLI will also store the contrast of parameter estimate (COPE) and the corresponding t-scores, e.g.

```
fMRI_cope_contrast1_MNI.nii  
fMRI_cope_contrast2_MNI.nii
```

```
fMRI_tscores_contrast1_MNI.nii  
fMRI_tscores_contrast2_MNI.nii
```

2.7 Output options

The following output options are available

- -savet1interpolated

Save T1 volume after resampling to MNI voxel size and resizing to MNI size (default no).

- -savet1alignedlinear

Save T1 volume linearly aligned to the MNI volume (default no).

- -savet1alignednonlinear

Save T1 volume non-linearly aligned to the MNI volume (default no).

- -saveepialignedt1

Save EPI volume aligned to the T1 volume (default no).

- -saveepialignedmni

Save EPI volume aligned to the MNI volume (default no).

- -saveallaligned

Save all aligned volumes (T1 interpolated, T1-MNI linear, T1-MNI non-linear, EPI-T1, EPI-MNI) (default no).

- -saveepimask

Save mask for fMRI data (default no).

- -saveslicetimingcorrected

Save slice timing corrected fMRI volumes (default no).

- -savemotioncorrected

Save motion corrected fMRI volumes (default no).

- -savesmoothed

Save smoothed fMRI volumes (default no).

- -saveactivityepi

Save activity maps in EPI space
(in addition to MNI space, default no).

- -saveactivityt1

Save activity maps in T1 space
(in addition to MNI space, default no).

- -saveresiduals

Save residuals after GLM analysis (default no).

- -saveresidualsmni

Save residuals after GLM analysis, in MNI space (default no).

- -saveoriginaldesignmatrix

Save the original design matrix used (default no). This is the design matrix before convolving the regressors with the hemodynamic response function.

- -savedesignmatrix

Save the total design matrix used (default no). This is the total design matrix being used by BROCCOLI. It contains detrending regressors and motion regressors (if -regressmotion is used).

- -savearparameters

Save the estimated AR coefficients (default no).

- -savearparameterst1

Save the estimated AR coefficients, in T1 space (default no).

- -savearparametersmni

Save the estimated AR coefficients, in MNI space (default no).

- -saveallpreprocessed

Save all preprocessed fMRI data
(slice timing corrected, motion corrected, smoothed) (default no).

- -saveunwhitenedresults

Save all statistical results without voxel-wise whitening (default no).

- -saveall

Save everything (default no).

2.8 Additional options

The following additional options are available

- -quiet

Don't print anything to the terminal (default false).

- -verbose

Print extra stuff (default false).

- -debug

Get additional debug information saved as nifti files (default no).
Warning: This will use a lot of extra memory!

2.9 Checking the registration

To check the registration between the anatomical volume and the brain template, set one volume as the underlay (e.g. `T1_brain_aligned_linear.nii` or `T1_brain_aligned_nonlinear.nii`) and one as the overlay (e.g. `MNI152_T1_1mm_brain.nii.gz`). To check the registration between the fMRI volume and the anatomical volume, set the interpolated anatomical volume (e.g. `T1_brain_interpolated.nii`) as the underlay and the aligned fMRI volume (e.g. `fMRI_aligned_t1.nii`) as the overlay.

Registration

3.1 Introduction

BROCCOLI provides a separate function for image registration. Linear as well as non-linear registration is applied to the input volume. The function automatically resizes and rescales the input volume to match the reference volume. In its simplest form, a registration between two volumes can with the bash wrapper be performed as

```
./RegisterTwoVolumes input_volume.nii reference_volume.nii
```

3.2 OpenCL options

The following OpenCL options are available

- -platform

The OpenCL platform to use (default 0).

- -device

The OpenCL device to use (default 0).

3.3 Registration options

The following registration options are available

- -iterationslinear

Number of iterations for the linear registration (default 10).

- -iterationsnonlinear

Number of iterations for the non-linear registration (default 10), 0 means that no non-linear registration is performed.

- -sigma

Amount of Gaussian smoothing applied for regularization of the displacement field, defined as sigma of the Gaussian kernel (default 5.0).

- -zcut

Number of mm to cut from the bottom of the input volume, can be negative, useful if the head in the volume is placed very high or low (default 0).

3.4 Outputs

By default, the function saves the results as `input_volume_aligned_linear.nii` (the result after linear registration) and `input_volume_aligned_nonlinear.nii` (the result after non-linear registration).

3.5 Output options

The following output options are available

- -savefield

Save the estimated displacement field to file (default false). This will generate `input_volume_displacement_x.nii`, `input_volume_displacement_y.nii.gz` and `input_volume_displacement_z.nii.gz`. They can be used with the function `TransformVolume`.

- -saveinterpolated

Save the input volume rescaled and resized to the size and resolution of the reference volume, before alignment (default false).

- -output

Set output filename (default `input_volume_aligned_linear.nii` and `input_volume_aligned_nonlinear.nii`).

3.6 Additional options

The following additional options are available

- -mask

Mask to apply after linear and non-linear registration, to the aligned volume. The option can for example be used to do a skullstrip (default none).

- -maskoriginal

Mask to apply after linear and non-linear registration. The option can for example be used to do a skullstrip. Returns the volume skullstripped and unregistered (but interpolated to the reference volume size) (default none).

- -quiet

Don't print anything to the terminal (default false).

- -verbose

Print extra stuff (default false).

- -debug

Get additional debug information saved as nifti files (default no).
Warning: This will use a lot of extra memory!

Transformation

4.1 Introduction

BROCCOLI provides a separate function for transformation of volumes. In its simplest form, a transformation can with the bash wrapper be performed as

```
./TransformVolume volume_to_transform.nii reference_volume.nii ...  
displacement_field_x.nii displacement_field_y.nii displacement_field_z.nii
```

where `reference_volume.nii` is the reference volume that was used for the registration, and `displacement_field_x.nii` `displacement_field_y.nii` `displacement_field_z.nii` are generated by `RegisterTwoVolumes`.

4.2 OpenCL options

The following OpenCL options are available

- `-platform`

The OpenCL platform to use (default 0).

- `-device`

The OpenCL device to use (default 0).

4.3 Transformation options

The following transformation options are available

- `-interpolation`

The interpolation to use, 0 = nearest, 1 = trilinear (default 1). Nearest interpolation can for example be useful if you want to transform a binary mask.

- `-zcut`

Number of mm to cut from the bottom of the input volume, can be negative (default 0). Should be the same as for the call to `RegisterTwoVolumes`.

4.4 Outputs

By default, the function saves the result as `volume__to__transform__warped.nii`.

4.5 Output options

The following output options are available

- `-output`

Set output filename (default `volume__to__transform__warped.nii`).

4.6 Additional options

The following additional options are available

- `-quiet`

Don't print anything to the terminal (default false).

Motion correction

5.1 Introduction

BROCCOLI provides a separate function for motion correction. In its simplest form, motion correction can with the bash wrapper be performed as

```
./MotionCorrection fMRI.nii
```

where fMRI.nii is a 4D fMRI dataset.

5.2 OpenCL options

The following OpenCL options are available

- -platform

The OpenCL platform to use (default 0).

- -device

The OpenCL device to use (default 0).

5.3 Motion correction options

The following motion correction options are available

- -iterations

Number of iterations for the motion correction (default 5).

5.4 Outputs

By default, the function saves the results as `input_mc.nii`. The estimated motion parameters are saved as `motion.1D`. They can for example be viewed using the AFNI function `1dplot` (`1dplot motion.1D`).

5.5 Output options

The following output options are available

- `-output`

Set output filename (default `input_mc.nii`).

5.6 Additional options

The following additional options are available

- `-quiet`

Don't print anything to the terminal (default `false`).

- `-verbose`

Print extra stuff (default `false`).

- `-debug`

Get additional debug information saved as nifti files (default `no`).

Warning: This will use a lot of extra memory!

6

Permutation testing

6.1 Randomise

BROCCOLI provides a function for permutation testing at the group level, similar to the FSL function `randomise`. In its simplest form, a permutation test can with the bash wrapper be performed as

```
./RandomiseGroupLevel volumes.nii -design design.mat -contrasts design.con
```

where `volumes.nii` is a 4D file with data from all subjects in the group(s). The `design.mat` file can for example be defined as

```
NumRegressors 2
NumSubjects 10
```

```
1.0 0.0
1.0 0.0
1.0 0.0
1.0 0.0
1.0 0.0
0.0 1.0
0.0 1.0
0.0 1.0
0.0 1.0
0.0 1.0
```

while the `design.con` file can be defined as

```
NumRegressors 2
NumContrasts 2
```

```
1.0 -1.0
-1.0 1.0
```

6.2 OpenCL options

The following OpenCL options are available

- -platform

The OpenCL platform to use (default 0).

- -device

The OpenCL device to use (default 0).

6.3 Permutation options

The following permutation options are available

- -design

The design matrix to apply in each permutation.

- -contrasts

The contrast vector(s) to apply to the estimated beta values.

- -groupmean

Test for group mean, using sign flipping
(design and contrast not needed).

- -mask

A mask that defines which voxels to permute (default none).

- -permutations

Number of permutations to use (default 10,000).

- -teststatistics

Test statistics to use, 0 = GLM t-test, 1 = GLM F-test, 2 = CCA, 3 = Searchlight (default 0). Only t-test is currently implemented.

- -inferencemode

Inference mode to use, 0 = voxel, 1 = cluster extent, 2 = cluster mass, 3 = threshold free cluster enhancement (TFCE) (default 1).

- -cdt

Cluster defining threshold for cluster inference (default 2.5).

- -significance

The significance level to calculate the threshold for (default 0.05).

- -permutationfile

Use a specific permutation file or sign flipping file (e.g. from FSL), provide the filename after the option.

6.4 Outputs

By default, the function saves the results as `volumes__perm__tvalues.nii` (t-scores for the unpermuted case) and `volumes__perm__pvalues.nii` (p-values from the permutation test).

6.5 Output options

The following output options are available

- -output

Set output filename (default `volumes__perm__tvalues.nii` and `volumes__perm__pvalues.nii`).

- -writepermutationvalues

Write all the permutation values to a text file, provide the filename after the option.

- -writepermutations

Write all the random permutations (or sign flips) to a text file, provide the filename after the option.

6.6 Additional options

The following additional options are available

- -quiet

Don't print anything to the terminal (default false).

- -verbose

Print extra stuff (default false).