BROCCOLI

-

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

Anders Eklund

2014-07-04

Table of Contents

1

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 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 or a Mac 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.

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.

1.5 Compiling the bash wrappers

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. 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
```

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 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@d1l-a10003 Bash_Wrapper]$ ./GetOpenCLInfo
Device info

-----------------------------------------
Platform number: 0
-----------------------------------------
Platform vendor: Intel(R) Corporation
Platform name: Intel(R) OpenCL
Platform extentions: 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 extentions: 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 extentions: 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
```

1.7 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.

1.8 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_GeForceGTXTITAN.bin
```

2

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 Registration options

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

2.3 Preprocessing options

There are currently two options for the preprocessing; the option -iterationsmc controls the number of iterations used for the motion correction (the default is 5) and the option -smoothing controls the amount of smoothing applied to each fMRI volume (the default is 6 mm).

2.4 Statistical options

A number of statistical options are available to the user. The option -regressmotion adds 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 option -temporalderivatives adds 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. For both these options, the contrast vectors are automatically extended with zeros for these additional regressors.

The option -permute results in that a permutation test is applied 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. The option -inferencemode determines if voxel-wise or cluster-wise p-values should be calculated, and the option -cdt sets the initial (uncorrected) voxel-wise threshold applied to the statistical map to define the clusters. The default number of permutations is 1,000, and can be changed with the option -permutations.

Bayesian first level analysis can be performed by using the option -bayesian. 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. The option -iterationsmcmc sets the number of MCMC iterations to be used (the default is 1,000).

2.5 Checking the results

To check the quality of the registrations, the option -saveallaligned can be used. This will save the anatomical (T1) volume interpolated and resized to match the brain template, the anatomical volume linearly aligned to the template, the anatomical volume non-linearly aligned to the template, the fMRI volume aligned to the anatomical volume and the fMRI volume aligned to the brain template.

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.

The total design matrix being used by BROCCOLI can be saved to a text file by using the option -savedesignmatrix. This design matrix contains the original regressors (and their temporal derivatives, if -temporalderivatives is used) convolved with a hemodynamic response function, detrending regressors (mean, linear trend, quadratic trend, cubic trend) and motion regressors (if -regressmotion is used). The unconvolved regressors can be saved to a text file by using the option -saveoriginaldesignmatrix.