**Methods of fMRI segmentation**

1. Brief state of the art in methods of fMRI segmentation – o metodach segmentacji fMRI – tutaj też można wycisnąć ze 2-3 strony??
2. Description of the implemented PCA method
3. Foundations and analytical derivation of the methods ???? – tutaj podstawy teoretyczne algorytmu PCA, 1 strona
4. Block diagram of the code (general)
5. Get\_mu – center the data
6. Svd
7. Enforce a sign convention

3. Mój plan pracy po ‘Foundations and analytical derivation’

1. Wstęp do matlabowskiej wersji algorytmu – o tym, że jest to oparte na matlabie
2. I tutaj dopiero można wkleić jakiś diagram / schemat algorytmu
3. Opis implementacji w CUDA

Plan opisu implementacji w CUDA:

1. Punkt pierwszy

Wstęp do matlabowskiej wersji algorytmu

Matlab pca function (standard library) is a reference implementation of this study. The implementation on CUDA bases on a code of this method. It is a default mode so singular value decomposition algorithm is used. An ‘economic’ version is implemented (**which means that the first N columns are returned**) as it is more efficient and the obtained result is enough to continue algorithm. It is worth mentioning that non-economic version would not run successfully on many GPUs as it requires a lot of memory for big datasets.

Diagram / schemat algorytmu

Selecting the only needed steps from Matlab’s pca function, the implemented algorithm on CUDA can be illustrated in the diagram below:

Which form the implemented algorithm on CUDA

General description:

In general PCA methods include some optional steps and exist different versions based on singular value decomposition (SVD), eigenvalue decomposition or alternating least squares

The implementation on CUDA bases on Matlab version algorithm with singular value

The first step (but might be optional) in the PCA method is to center the data. Then the main part is coming which is

Enforce a sign convention on the coefficients – the largest element in each column will have a positive sign.

Multicore threads analysis – CUDA IMPLEMENTATION

“to conform CULA alignment requirements”

The implemented algorithm was optimized for specific dimensions of matrices (m >> n) of tested fMRI data. It might not be efficient on data with another ratio of dimensions.

The method starts with centering the data. It means calculate the average of each column, and subtract this average from each element of the column.

The simplest scheme to do it seems to do a sum reduction

Each column is processed by one block, (sum reduction) so that shared memory

A version using shuffle instruction (to compute sum reduction in the columns) was tested but did not result in better performance (there was no speed-up).

W sumie to można napisać o tych transpozycjach, że je robimy.

The most important (computational complexity and cost) part of the algorithm is singular value decomposition. A quick research of already implemented SVD method has shown that there are not many libraries for CUDA offering it. In fact CUDA API includes cuSOLVER library with SVD methods but they do not support “economic” version of the algorithm, so they are impractical for the large datasets (large matrices). In this work an implementation of SVD from CULA library was used. This is the library of linear algebra methods basing on LAPACK library implementation.

However CULA library has not been developed since 2013, so it does not take advantage of the features new CUDA release offers.

Przy podawaniu wyników czasowych, należy podać oczywiście dane techniczne procesorów na których były przeprowadzane testy.