

## Research Article

# Fast Random Permutation Tests Enable Objective Evaluation of Methods for Single-Subject fMRI Analysis

Anders Eklund,<sup>1,2</sup> Mats Andersson,<sup>1,2</sup> and Hans Knutsson<sup>1,2</sup>

<sup>1</sup>Division of Medical Informatics, Department of Biomedical Engineering, Linköping University, Linköping, Sweden

<sup>2</sup>Center for Medical Image Science and Visualization (CMIV), Linköping University, Linköping, Sweden

Correspondence should be addressed to Anders Eklund, anders eklund@liu.se

Received 19 April 2011; Accepted 14 July 2011

Academic Editor: Yasser M. Kadah

Copyright © 2011 Anders Eklund et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Parametric statistical methods, such as  $Z$ -,  $t$ -, and  $F$ -values, are traditionally employed in functional magnetic resonance imaging (fMRI) for identifying areas in the brain that are active with a certain degree of statistical significance. These parametric methods, however, have two major drawbacks. First, it is assumed that the observed data are Gaussian distributed and independent; assumptions that generally are not valid for fMRI data. Second, the statistical test distribution can be derived theoretically only for very simple linear detection statistics. With nonparametric statistical methods, the two limitations described above can be overcome. The major drawback of non-parametric methods is the computational burden with processing times ranging from hours to days, which so far have made them impractical for routine use in single-subject fMRI analysis. In this work, it is shown how the computational power of cost-efficient graphics processing units (GPUs) can be used to speed up random permutation tests. A test with 10000 permutations takes less than a minute, making statistical analysis of advanced detection methods in fMRI practically feasible. To exemplify the permutation-based approach, brain activity maps generated by the general linear model (GLM) and canonical correlation analysis (CCA) are compared at the same significance level.

## 1. Introduction

Functional magnetic resonance imaging (fMRI) is used in neuroscience and clinic for investigating brain activity patterns and for planning brain surgery. Activity is detected by fitting an activity model to each observed fMRI voxel time series and then testing whether the null hypothesis of no activity can be rejected or not based on the model parameters. Specifically, this test is performed by subjecting a test statistic calculated from the model parameters to a threshold. To control the randomness due to noise in this test procedure, it is desirable to find the statistical significance associated with the detection threshold, that is, how likely it is that a voxel is declared active by chance. When the statistical distribution of the data is known *and* when the probability (null) distribution of the test statistic can be derived, parametric statistics can be used to this end. This is for example the case for the commonly used general linear model (GLM), for which the well-known  $t$ -test and  $F$ -test can be derived when the input data are

independently Gaussian distributed. However, when the data distribution is not known or the distribution of the test statistics cannot be derived, parametric statistical tests can only yield approximate thresholds or cannot be applied at all. This is generally the case in fMRI analysis as the noise in fMRI data is not Gaussian and independent [1–5]. Even though the noise can be made uncorrelated through a whitening procedure [6, 7], the noise structure must first be estimated using methods that themselves are susceptible to random errors. Accurately accounting for this variance in the test statistic distribution is difficult. Furthermore, more advanced detection approaches often adaptively utilize the spatial context of fMRI activation patterns to improve the detection, or they perform other operations that make the derivation of the test statistic distribution mathematically intractable [8–14]. Said otherwise, only for the very simplest test statistics, such as the GLM, can a parametric test distribution be derived theoretically. On top of the problems described above, the multiple testing problem [15] must be solved because one is generally interested to test whether

there is any activity in the entire brain at all and not just if there is activity in a single voxel. This complicates the derivation of the test statistic distribution even further. To conclude, the parametric statistical approach is applicable only to a very limited set of tests and is subject to many sources of error.

In contrast to the parametric approach, a nonparametric approach does not assume the statistical properties of the input data to be known [16]. Furthermore, there is no need to derive the theoretical distribution of the test statistic, and even thresholds corrected for multiple testing are straightforwardly found. Nonparametric approaches have been studied extensively in functional neuroimaging [10, 17–28]. Semiparametric approaches have also been proposed [29]. In particular, so-called resampling or permutation methods have been studied, which randomly permute or reshuffle the original fMRI data to remove any activation signal but otherwise keep its statistical structure. Thus, thousands of simulated null data sets without activation can be generated and analysed to empirically simulate the null distribution of the test statistic. The major drawback of nonparametric statistical approaches for single-subject fMRI analysis is the computational complexity, requiring hours or days of processing time on regular computer hardware.

Graphics processing units (GPUs) have seen a tremendous development during the last decade and have been applied to a variety of fields to achieve significant speedups, compared to optimized CPU implementations. The main difference between the GPU and the CPU is that the GPU does all the calculations in parallel, while the CPU normally does them in serial. In the field of neuroimaging and neuroscience the use of the GPU is quite new. As single-subject fMRI analysis normally is done separately for each time series in the fMRI data, it suits perfectly for parallel implementations. In our recent work [30] we therefore describe how to preprocess (i.e., apply slice timing correction, motion correction, smoothing, and detrending) and how to statistically analyze the fMRI data on the GPU. The result is a significantly faster analysis than if the CPU is used. For a small fMRI dataset (80 volumes of the resolution  $64 \times 64 \times 22$  voxels) all the preprocessing is done in about 0.5 s and the statistical analysis is done under 0.5 ms. Recently, GPUs have also been used to speed up functional connectivity analysis of fMRI data [31, 32]. A final example is the work by Ferreira da Silva [33] that used the GPU to speed up the simulation of a Bayesian multilevel model for fMRI analysis.

In this work, it is shown how nonparametric statistics can be made practical for single-subject fMRI analysis by using the parallel processing power of the GPU. The idea of using the GPU for random permutation tests is not new; it has recently been done in biostatistics [34, 35]. The GPU makes it possible to estimate the null distribution of a test statistic, corrected for multiple testing, in the order of minutes. This has significant implications on the way fMRI analysis can be carried out as it opens the possibility to routinely apply more powerful detection methods than the GLM. As an example, the results of the standard GLM detection is in this work compared with a restricted canonical correlation analysis

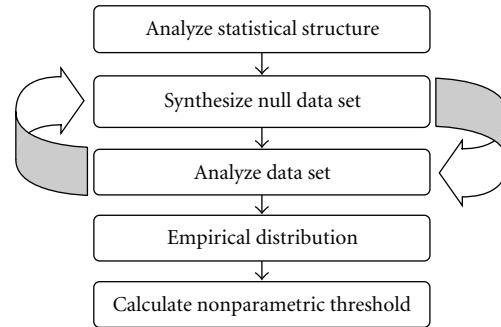


FIGURE 1: Flowchart containing the main building blocks for nonparametric analysis of single-subject fMRI data.

(CCA) method [9] that adaptively incorporates spatial context in the detection. The short processing time also facilitates deeper investigations into the influence of various noise and detrending models on the statistical significance, as well as validation of approximate parametric approaches such as Bonferroni and random field theory (RFT) [36–39].

## 2. Methods

**2.1. Basics of Random Permutation Tests.** One subset of nonparametric tests is permutation tests where the statistical analysis is done for all the possible permutations of the data. Complete permutation tests are, however, not feasible if the number of possible permutations is very large. For a time series with 80 samples, there exists  $7.16 \cdot 10^{118}$  possible permutations. It is therefore common to instead do *random* permutation tests [40], also called Monte Carlo permutation tests, where the statistical analysis is made for a sufficiently large number of random permutations, for example 10 000, of the data. The main idea is to estimate the null distribution of the test statistics, by generating and analysing surrogate data that is similar to the original data. The surrogate data is generated by permuting, or reshuffling, the data between the different groups to be compared. The main idea of the nonparametric approach is given in Figure 1.

**2.2. The Problem of Multiple Testing.** By applying a threshold to the activity map, each voxel can be classified as active or inactive. The threshold is normally selected as a level of significance, one may for example want that only voxels that with at least 95% significance are to be considered as active. If a statistical test is repeated and a family-wise error rate  $\alpha$  is desired, the error rate for each test must be smaller than  $\alpha$ . This is known as the problem of multiple testing. If Bonferroni correction is used, the error rate for each comparison becomes  $\alpha/N_v$ , where  $N_v$  is the number of tests (voxels). This is a correct solution if the tests are independent. In fMRI it is common to perform the statistical analysis for more than 20 000 brain voxels; if a threshold of  $P = 0.05$  is used to consider the voxel as active, the  $P$  value becomes  $0.05/20000$  with Bonferroni correction. The assumptions that are made about the behaviour of the tail of the distribution are thereby critical.

There are three problems with Bonferroni correction in fMRI. First, the test statistics is assumed to, under the null hypothesis, follow a certain distribution, such as Student's  $t$ -distribution. Second, the smoothness of the data is not taken into account as the Bonferroni threshold only considers the number of tests. The smoothing increases the spatial correlation of the data and thereby reduces the effective number of independent tests. Third, it is assumed that all the voxels have the same null distribution. To avoid Bonferroni correction, another approach based on Gaussian random field theory [36, 38, 39] has been developed and is used in the statistical parametric mapping (SPM) software (<http://www.fil.ion.ucl.ac.uk/spm/>). While this approach takes the smoothness of the data into account, several assumptions are necessary for the theory to be valid and it is still assumed that all the voxels have the same null distribution.

The nonparametric approach can be used to solve the problem of multiple testing as well. This is done by estimating the null distribution of the *maximum* test statistic [19, 21, 24, 39] by only saving the maximum test value from each permutation, to get a *corrected* threshold. This means that about 10 000 permutations have to be used [19, 21], while as little as 10 permutations can be enough if an *uncorrected* threshold is sufficient [18, 20, 22].

**2.3. Preprocessing of fMRI Time Series.** As fMRI time series are temporally correlated [6, 36, 41], the time series have to be preprocessed before they are permuted. Otherwise the exchangeability criterion is not satisfied and the temporal structure is destroyed. Most of the temporal correlations originate from different kinds of trends. In this work these trends are removed by a cubic detrending, such that the mean and any polynomial trend up to the third order is removed, but more advanced detrending is possible [42].

Several approaches have been proposed for the random resampling, the most common being whitening transforms [6, 18, 19], wavelet transforms [22, 26], and Fourier transforms [43]. A comparison of these approaches [27] indicates that whitening performs best, at least for fMRI data that is collected during block-based stimuli paradigms. The whitening transform is done by estimating an autoregressive (AR) model for each time series. This can, for example, be done by solving the equation system that is given by the Yule-Walker equations.

To accurately estimate AR parameters from a small number of time points (80 in our case) is quite difficult. To improve the estimates a spatial Gaussian lowpass filter (8 mm FWHM) is therefore applied to the estimated AR parameters [7]. In statistics this technique is normally known as variance pooling. The optimal amount of smoothing was found by testing the AR estimation procedure on temporally correlated Gaussian noise where the spatial patterns of the AR parameters were known. Our amount of smoothing (8 mm) is less than the first application of smoothing of the parameters (15 mm) [7] but close to the optimal amount of smoothing (6.5–7.5 mm) found by further investigation [44]. It has also been reported that the AR estimates are better without the spatial smoothing [45].

Normalized convolution [46] is used to prevent that the smoothing includes AR parameters from outside the brain. With normalized convolution it is possible to use a certainty value for each sample in the convolution. The certainty weighted filter response  $cwr$  is calculated as

$$cwr = \frac{(c \cdot s) * f}{c * f}, \quad (1)$$

where  $c$  is the certainty,  $s$  is the signal,  $f$  is the filter,  $\cdot$  denotes point-wise multiplication, and  $*$  denotes convolution. In our case the certainty is set to 1 for the brain voxels and 0 otherwise. Without the normalized convolution the estimated AR parameters at the edge of the brain are too low, as the AR parameters outside the brain are very close to 0. To further improve the estimates, the whitening procedure is iterated 3 times and the AR estimates are accumulated [7, 22] (a higher number of iterations seem to impair the estimates).

To investigate if the time series really are white noise after the whitening, several tests can be applied. One example is the Durbin-Watson test that previously has been used to test if the residuals from the GLM contain autocorrelation [2]. The problem with this test is, however, that it only tests if there is an AR(1) correlation or not, it cannot handle higher-order correlations. A more general test is the Box-Pierce test that tests if at least one of the autocorrelations up to a defined time lag  $h$  is significantly different from zero. The Box-Pierce test has also been used for testing whiteness of fMRI data [22]. The Ljung-Box test [47] has been proven to be better than the Box-Pierce test for small sample sizes and is therefore used in our case. The test statistic  $Q$  is calculated as

$$Q = N_t(N_t + 2) \sum_{k=1}^h \frac{(r_Y(k))^2}{N_t - k}, \quad (2)$$

where  $N_t$  is the number of time samples,  $r_Y(k)$  is the sample autocorrelation at time lag  $k$ , and  $h$  is the number of time lags being tested. The test statistic is asymptotically chi-square distributed with  $h - p$  degrees of freedom, where  $p$  is the order of the AR model used for the whitening, when  $N_t$  grows towards infinity. Since our whitening is done with the smoothed AR parameters, the Ljung-Box test is applied to smoothed auto correlations.

Since the spatial correlation should be maintained, but not the temporal, the same permutation is applied to all the time series [19, 43]. When the time series have been permuted, an inverse whitening transform is applied by simulating an AR model, using the permuted whitened time series as innovations.

**2.4. Statistical Analysis, GLM and  $t$ -Test.** The general linear model (GLM) is the most used approach for statistical analysis of fMRI data [37]. For each voxel time series, a linear model is fitted according to

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad (3)$$

where  $\mathbf{Y}$  is the time series,  $\mathbf{X}$  the regressors,  $\boldsymbol{\beta}$  the parameters to estimate, and  $\boldsymbol{\epsilon}$  the errors. The regressors were created by

convolving the stimulus paradigm with the hemodynamic response function (HRF) (difference of gammas) and its temporal derivative [6]. The two regressors were mean corrected, Euclidean normalized, and orthogonalized. No other regressors were used in the design matrix. The regression weights are estimated with ordinary least squares

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}, \quad (4)$$

and the  $t$ -test value is then calculated as

$$t = \frac{\mathbf{c}^T \hat{\boldsymbol{\beta}}}{\sqrt{\text{var}(\hat{\boldsymbol{\beta}}) \mathbf{c}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{c}}}, \quad (5)$$

where  $\mathbf{c}$  is the contrast vector.

Prior to the GLM the time series were whitened by using the same AR(1) model for all the voxels [6, 18]. No additional high-pass or low-pass filtering was used. The whitening step prior to the GLM is not necessary for the permutation-based analysis. The purpose of the whitening is to make sure that the errors are temporally uncorrelated, otherwise the assumptions that are necessary for the GLM to generate a true  $t$ -value are violated. Without the whitening a true  $t$ -value is not obtained, but a pseudo  $t$ -value. This is not a problem for the permutation-based analysis as the null distribution of the test statistics is estimated. If the thresholds from random field theory and a random permutation test are to be compared, the whitening has to be done in each permutation.

**2.5. Statistical Analysis, CCA.** One statistical approach for fMRI analysis that provides more adaptivity to the data is canonical correlation analysis (CCA) [48]. While the GLM works with *one* multidimensional variable (e.g., temporal basis functions, [37]), CCA works with *two* multidimensional variables (e.g., temporal and spatial basis functions, [9]). Ordinary correlation between two one-dimensional variables  $x$  and  $y$  with zero mean can be written as

$$\rho = \text{Corr}(x, y) = \frac{E[xy]}{\sqrt{E[x^2] E[y^2]}}. \quad (6)$$

This expression can easily be extended to multidimensional variables. The GLM calculates the correlation between one multidimensional variable  $\mathbf{x}$  and one one-dimensional variable  $y$  according to

$$\rho = \text{Corr}(\boldsymbol{\beta}^T \mathbf{x}, y), \quad (7)$$

where  $\boldsymbol{\beta}$  is the weight vector that determines the linear combination of  $\mathbf{x}$ . Canonical correlation analysis is a further generalization of the GLM, such that both the variables are multidimensional. The canonical correlation is defined as

$$\rho = \text{Corr}(\boldsymbol{\beta}^T \mathbf{x}, \boldsymbol{\gamma}^T \mathbf{y}) = \frac{\boldsymbol{\beta}^T \mathbf{C}_{xy} \boldsymbol{\gamma}}{\sqrt{\boldsymbol{\beta}^T \mathbf{C}_{xx} \boldsymbol{\beta} \boldsymbol{\gamma}^T \mathbf{C}_{yy} \boldsymbol{\gamma}}}, \quad (8)$$

where  $\mathbf{C}_{xy}$  is the covariance matrix between  $\mathbf{x}$  and  $\mathbf{y}$ ,  $\mathbf{C}_{xx}$  is the covariance matrix for  $\mathbf{x}$ , and  $\mathbf{C}_{yy}$  is the covariance matrix

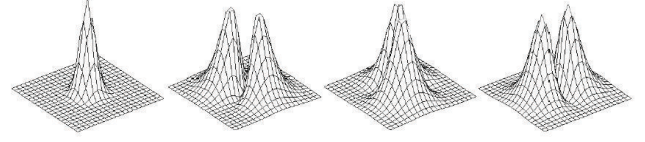


FIGURE 2: The four smoothing filters that are used for 2D CCA, one small isotropic separable filter and three anisotropic nonseparable filters. For visualization purposes, these filters are interpolated to a subpixel grid.

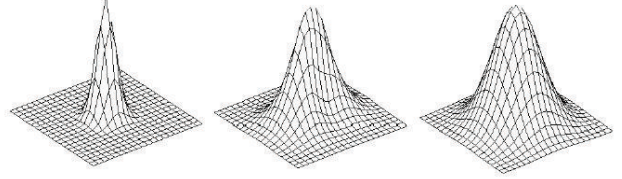


FIGURE 3: **Left:** A small isotropic lowpass filter can be used by CCA by setting the weights of all the other filters to zero. **Middle:** anisotropic lowpass filters with arbitrary orientation can be created by CCA by first combining the anisotropic filters and then adding the small lowpass filter. **Right:** by using the same weight for all the filters, a large isotropic lowpass filter can be obtained.

for  $\mathbf{y}$ . The temporal and spatial weight vectors,  $\boldsymbol{\beta}$  and  $\boldsymbol{\gamma}$ , that give the highest correlation are calculated as the eigenvectors of two eigenvalue problems. The canonical correlation is the square root of the corresponding eigenvalue.

The temporal basis functions for CCA are the same as for the GLM. The spatial basis functions can, for example, be neighbouring pixels [8, 10, 49] or a number of anisotropic filters [9] that linearly can be combined to a lowpass filter with arbitrary orientation, to prevent unnecessary smoothing. In contrast to the GLM, an adaptive anisotropic smoothing is obtained, instead of a fix isotropic smoothing. The four smoothing filters that are used for our implementation of 2D CCA are given in Figure 2. Three filters that can be constructed as a linear combination of the four filters are given in Figure 3.

Compared to other approaches that adaptively include spatial information [11–14], the advantage with CCA is that there exists an analytical solution that gives the best weight vectors, while the other approaches have to search for the best combination.

One disadvantage with CCA is that it is difficult to calculate the threshold for a certain significance level, as the distribution of the canonical correlation coefficients is rather complicated. If  $\mathbf{x}$  and  $\mathbf{y}$  are Gaussian distributed and independent, the joint probability distribution for *all* the sample canonical correlation coefficients is given by [50]

$$f = \prod_{i=1}^m \left( (r_i^2)^{(n-m-1)/2} (1-r_i^2)^{(N-n-m-1)/2} \prod_{j=i+1}^m (r_i^2 - r_j^2) \right), \quad (9)$$

where  $N$  is the number of (time) samples,  $n$  and  $m$  are the dimensions of the multidimensional variables  $\mathbf{x}$  and  $\mathbf{y}$ , and  $r_i$  are the canonical correlation coefficients.

Another problem is that restricted CCA (RCCA) [51] normally is used instead of ordinary CCA, in order to guarantee that the resulting combinations of the temporal and spatial basis functions are plausible. To our knowledge there is no theoretical distribution for restricted canonical correlation coefficients. The only solution to get a significance threshold for RCCA is thus to use nonparametric approaches.

As a 2D version of CCA already had been implemented [30], it was easy to extend the random permutation tests to include CCA as well. The problem with the original 3D CCA approach [9] is that it uses a total of seven 3D filters, and thereby a  $7 \times 7$  matrix must be inverted for each time series. Our GPU implementation, however, only supports inverting  $4 \times 4$  matrices, and thereby a maximum of 4 filters. Another approach [52] that uses two 3D filters, one isotropic Gaussian kernel and its derivative (with respect to the width parameter sigma), was therefore used. This makes it possible for CCA to create filters with different sizes, such that the amount of smoothing varies between the voxels. All the resulting filters are, however, isotropic, which makes this version of 3D CCA less adaptive.

**2.6. Spatial Smoothing.** The smoothing of the fMRI volumes has to be applied in each permutation. If the data is smoothed prior to the whitening transform, the estimated AR parameters will change with the amount of smoothing applied since the temporal correlations are altered by the smoothing. For our implementation of 2D CCA, 4 different smoothing filters are applied. If the smoothing is done prior to the permutations, 4 time series have to be permuted for each voxel and these time series will have different AR parameters. The smoothing will also change the null distribution of each voxel. This is incorrect as the surrogate null data that is created always should have the same properties, regardless of the amount of smoothing that is used for the analysis. If the data is smoothed after the whitening transform, but before the permutation and the inverse whitening transform, the time series that are given by simulating the AR model are incorrect since the properties of the noise are altered. The only solution to this is to apply the smoothing *after* the permutation and the inverse whitening transform, that is, in each permutation. This is also more natural in the sense that the surrogate data first is created and then analysed.

Similarly, if the activity map is calculated as how important each voxel is for a classifier [11–14], the classifier has to be trained in each permutation in order to estimate the null distribution.

**2.7. The Complete Algorithm.** The complete algorithm can be summarized as follows. The reason why the detrending is done separately, compared to having the detrending basis functions in the design matrix, is that the detrending has to be done separately for the CCA approach.

The whitening in each permutation is only performed to be able to compare the corrected  $t$ -thresholds from the random permutation test to the thresholds from Bonferroni correction and random field theory.

- (1) Preprocess the fMRI data, that is, apply slice timing correction, motion correction, smoothing, and cubic detrending. To save time, the statistical analysis is only performed for the brain voxels. A simple thresholding technique is used for the segmentation.
- (2) Whiten the detrended time series (GLM only).
- (3) Apply the statistical analysis to the preprocessed fMRI data and save the test values. These are the original test values  $t_{\text{voxel}}$ .
- (4) Apply cubic detrending to the motion compensated time series.
- (5) Remove the best linear fit between the detrended time series and the temporal basis functions in the design matrix, by ordinary least squares, to create residual data (as the *null* distribution is to be estimated). Estimate AR parameters from the residual time series. Apply a spatial smoothing to improve the estimates of the AR parameters. Apply whitening with the smoothed AR parameters. Repeat the whitening procedure 3 times.
- (6) For each permutation,
  - (i) apply a random permutation to the whitened time series,
  - (ii) generate new fMRI time series by an inverse whitening transform, that is, by simulating an AR model in each voxel with the permuted whitened time series as innovations,
  - (iii) smooth all the volumes that were generated by the inverse whitening transform,
  - (iv) apply cubic detrending to the smoothed time series,
  - (v) whiten the detrended time series (GLM only),
  - (vi) apply the statistical analysis,
  - (vii) find the maximum test value and save it.
- (7) Sort the maximum test values.
- (8) The threshold for a desired corrected  $P$  value is given by extracting the corresponding test value from the sorted maximum test values. If 10 000 permutations are used, the threshold for corrected  $P = 0.05$  is given by the sorted maximum test value at location 9500.
- (9) The corrected  $P$  value at each voxel is calculated as the number of maximum test values,  $t_{\text{max}_i}$ , that are greater than or equal to the original test value in the voxel,  $t_{\text{voxel}}$ , divided by the number of permutations  $N_p$

$$P_{\text{voxel}}^c = \frac{\sum_{i=1}^{N_p} (t_{\text{max}_i} \geq t_{\text{voxel}})}{N_p}. \quad (10)$$

A comparison of the flowcharts for a parametric analysis and a nonparametric analysis is given in Figure 4.

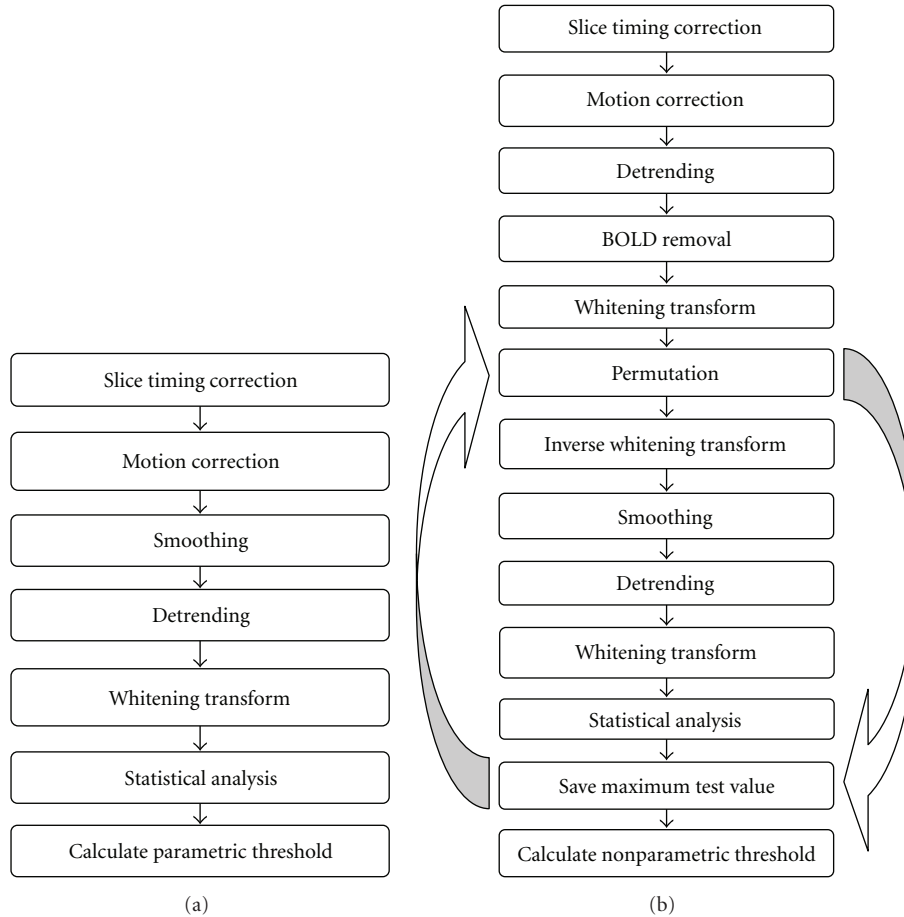


FIGURE 4: (a) Flowchart for conventional parametric analysis of fMRI data. (b) Flowchart for nonparametric analysis of fMRI data. In each permutation a new null dataset is generated and analysed.

**2.8. The Number of Permutations.** The number of permutations that are required depends on the desired  $P$  value and the accuracy that is required. The standard deviation of the desired (one sided)  $P$  value is approximately  $\sqrt{p(1-p)/N_p}$ , where  $N_p$  is the number of permutations [53]. Some examples of desired  $P$  value, number of permutations, and relative standard deviation are given in Table 1.

### 3. GPU Implementation

The random permutation test was implemented with the CUDA (Compute Unified Device Architecture) programming language by Nvidia [54], which is explained by Kirk and Hwu [55]. In this section we will shortly describe how to implement the whitening transform and the random permutation test on the GPU. The interested reader is referred to our recent work [30] for more details and how to implement the other processing steps. The main principle of our GPU implementation is that each GPU thread works on a separate voxel time series.

Our CUDA implementation was compared with a standard C implementation and an OpenMP-based implementation. The Open MP (Open Multi-Processing) library lets

TABLE 1: Relative standard deviation of the desired  $P$  value, as function of desired  $P$  value and number of permutations.

Number of Permutations/ $P$ value	0.1	0.05	0.01
1000	9.48%	13.78%	31.46%
5 000	4.24%	6.16%	14.07%
10 000	3%	4.35%	9.95%
50 000	1.34%	1.94%	4.45%
100 000	0.95%	1.37%	3.14%

the user take advantage of all the CPU cores in an easy way. All the implementations have been done in Matlab (Mathworks, Natick, Mass), using the mex interface where C and CUDA code can be used together with Matlab. For all implementations, 32 bit floats were used. The used graphics cards were three Nvidia GTX 480, each equipped with 480 processor cores and 1.5 GB of memory, giving a total of 1440 processor cores that run at 1.4 GHz. The used CPU was an Intel Xeon 2.4 GHz with 12 MB of L3 cache and 4 processor cores, and 12 GB of memory was used. The operating system used was Linux Fedora 12 64-bit. The total price of the

computer was about 4000 \$, a fraction of the price for a PC cluster with equivalent computational performance.

**3.1. Whitening and the Random Permutations.** Before the data is permuted an AR model is first estimated for each time series, as previously described. To solve the equation system that is given by the Yule-Walker equations requires a matrix inverse of the size  $p \times p$  where  $p$  is the order of the AR model. To actually do the matrix inverse in each thread on the GPU is not a problem, even for matrices larger than  $4 \times 4$ , but to do it for a  $7 \times 7$  matrix requires a lot of float registers and the Nvidia GTX 480 can only use 64 float registers per thread. The CUDA compiler will put the rest of the float variables, that do not fit into the registers, in the local memory which is extremely slow. Due to this it is hard to achieve good performance for matrices that are larger than  $4 \times 4$ . This is also the reason why the original 3D CCA approach, that uses seven 3D filters, cannot be used. Other than this the estimation of the AR parameters suits the GPU well, as the parameters are estimated in exactly the same way for each voxel time series. When the AR parameters have been estimated, they are spatially smoothed in order to improve the estimates. For this purpose a separable 3D convolver, created for 3D CCA and 3D GLM, is used.

For the estimation of AR parameters, the whitening transform, and the inverse whitening transform the shared memory is used to store the  $p$  last time points and each GPU thread loops along the time dimension for one voxel.

The permutation step is done by using randomized indices. A permutation matrix of size  $N_p \times N_t$  is first generated in Matlab, by using the function *randperm*, and is then copied to the GPU. For each permutation one row of the permutation matrix is copied to the constant memory and is used to read the data in the randomized order. It might seem difficult to achieve coalesced reads when the time samples are to be read in a randomized order, in our case this is however not a problem. The fMRI data is stored as  $(x, y, z, t)$  (i.e.,  $x$  first, then  $y$  and so on) and the permutation is only done along the time dimension, and not along the spatial dimensions. Due to this fact it is always possible to read 32 values at the time along  $x$ , regardless of the current time point. The code in Algorithm 1 generates a new time series for one voxel, by simulating an AR(4) model using the permuted whitened time series as innovations:

```
c_Permutation_Vector
```

is the index vector that contains the random time indices. The inverse whitening transform and the permutation step are thus performed at the same time. The help functions

```
Get_3D_Index, Get_4D_Index
```

calculate the linear index for the 3D and the 4D case.

For this kernel, and for most of the other kernels, each thread block contains a total of 512 threads (32 along  $x$ , 16 along  $y$ , and 1 along  $z$ ) and uses 16 KB of shared memory (one  $16 \times 8 \times 32$  float array) which makes it possible to run three thread blocks in parallel on each multiprocessor. This results in 1536 active threads per multiprocessor and

thereby a total of 23 040 active threads on the GPU, which is the maximum for the Nvidia GTX 480.

To find the maximum test value in each permutation, one fMRI slice ( $64 \times 64$  pixels) is first loaded into shared memory. The maximum value of this slice is then found by comparing two values at the time. The number of values is thus first reduced from 4096 to 2048, then to 1024 and after 12 reductions to the maximum test value. The maximum values of the 22 slices are then compared. After each permutation the maximum test value is copied to host memory.

In order to calculate the  $P$  value for each voxel, the maximum test values are first copied from host memory to constant memory. Each GPU thread then loops over all the maximum test values and calculates how many of the test values are bigger than or equal to the test value for one voxel.

**3.2. Multi-GPU.** As our computer contains three graphic cards, a multi-GPU implementation of the analysis was also made, such that each GPU does one-third of the permutations. Each GPU first preprocesses the fMRI data, GPU 1 uses the first part of the permutation matrix, GPU 2 uses the middle part, and GPU 3 uses the last part. The processing time thus scales linearly with the number of GPUs. A short demo of the multi-GPU implementation can be found at <http://www.youtube.com/watch?v=wxMqZw0jcOk>.

## 4. Results

In this section we will present the processing times for the different implementations, compare activity maps from GLM and CCA at the same significance level, and compare estimated thresholds from Bonferroni correction, Gaussian random field theory, and random permutation tests.

**4.1. Data.** Four single-subject datasets have been used to test our algorithms; the test subject was a 50-year-old healthy male. The data was collected with a 1.5 T Philips Achieva MR scanner. The following settings were used: repetition time 2 s, echo time 40 ms, flip angle 90 degrees, and isotropic voxel size 3.75 mm. A field of view of 240 mm thereby resulted in slices with  $64 \times 64$  pixels, and a total of 22 slices were collected every other second. The experiments were 160 s long, resulting in 80 volumes to be processed. The datasets contain about 20 000 within-brain voxels.

**4.1.1. Motor Activity.** For the *Motor 1* dataset the subject periodically activated the left hand (20 s activity, 20 s rest), and for the *Motor 2* dataset the subject periodically activated the right hand.

**4.1.2. Language Activity.** For the *Language* dataset the subject periodically performed a reading task (20 s activity, 20 s rest). The task was to read sentences and determine if they were reasonable or not.

**4.1.3. Null.** For the *null* dataset the subject simply rested during the whole experiment.

```

float alpha1 = alphas1[Get_3D_Index(x,y,z,DATA_W,DATA_H)];
float alpha2 = alphas2[Get_3D_Index(x,y,z,DATA_W,DATA_H)];
float alpha3 = alphas3[Get_3D_Index(x,y,z,DATA_W,DATA_H)];
float alpha4 = alphas4[Get_3D_Index(x,y,z,DATA_W,DATA_H)];

s_Y[threadIdx.y][0][threadIdx.x] =
whitened_volumes[Get_4D_Index(x,y,z,c.Permutation_Vector[0],DATA_W,DATA_H,DATA_D)];
s_Y[threadIdx.y][1][threadIdx.x] = alpha1 * s_Y[threadIdx.y][0][threadIdx.x]
+ whitened_volumes[Get_4D_Index(x,y,z,c.Permutation_Vector[1],DATA_W,DATA_H,DATA_D)];
s_Y[threadIdx.y][2][threadIdx.x] = alpha1 * s_Y[threadIdx.y][1][threadIdx.x]
+ alpha2 * s_Y[threadIdx.y][0][threadIdx.x]
+ whitened_volumes[Get_4D_Index(x,y,z,c.Permutation_Vector[2],DATA_W,DATA_H,DATA_D)];
s_Y[threadIdx.y][3][threadIdx.x] = alpha1 * s_Y[threadIdx.y][2][threadIdx.x]
+ alpha2 * s_Y[threadIdx.y][1][threadIdx.x] + alpha3 * s_Y[threadIdx.y][0][threadIdx.x]
+ whitened_volumes[Get_4D_Index(x,y,z,c.Permutation_Vector[3],DATA_W,DATA_H,DATA_D)];

permuted_volumes[Get_4D_Index(x,y,z,0,DATA_W,DATA_H,DATA_D)] = s_Y[threadIdx.y][0][threadIdx.x];
permuted_volumes[Get_4D_Index(x,y,z,1,DATA_W,DATA_H,DATA_D)] = s_Y[threadIdx.y][1][threadIdx.x];
permuted_volumes[Get_4D_Index(x,y,z,2,DATA_W,DATA_H,DATA_D)] = s_Y[threadIdx.y][2][threadIdx.x];
permuted_volumes[Get_4D_Index(x,y,z,3,DATA_W,DATA_H,DATA_D)] = s_Y[threadIdx.y][3][threadIdx.x];

// Loop over time points
for (t = 4; t < DATA_T; t++)
{
s_Y[threadIdx.y][4][threadIdx.x] =
alpha1 * s_Y[threadIdx.y][3][threadIdx.x]
+ alpha2 * s_Y[threadIdx.y][2][threadIdx.x]
+ alpha3 * s_Y[threadIdx.y][1][threadIdx.x]
+ alpha4 * s_Y[threadIdx.y][0][threadIdx.x]
+ whitened_volumes[Get_4D_Index(x,y,z,c.Permutation_Vector[t],DATA_W,DATA_H,DATA_D)];

permuted_volumes[Get_4D_Index(x,y,z,t,DATA_W,DATA_H,DATA_D)] = s_Y[threadIdx.y][4][threadIdx.x];

// Save old values
s_Y[threadIdx.y][0][threadIdx.x] = s_Y[threadIdx.y][1][threadIdx.x];
s_Y[threadIdx.y][1][threadIdx.x] = s_Y[threadIdx.y][2][threadIdx.x];
s_Y[threadIdx.y][2][threadIdx.x] = s_Y[threadIdx.y][3][threadIdx.x];
s_Y[threadIdx.y][3][threadIdx.x] = s_Y[threadIdx.y][4][threadIdx.x];
}

```

ALGORITHM 1

4.2. *Processing Times.* The processing times for the random permutation tests, for the different implementations, are given in Tables 2 and 3. The reason why the processing time does not scale linearly with the number of permutations is that it takes some time to copy the data to and from the GPU. Before the permutations are started, the fMRI data is preprocessed on the GPU and this takes about 0.5 s. The processing times for the different processing steps can be found in our recent work [30].

The reason why the processing time for CCA is much longer than for the GLM for the CPU implementations is that the 2D version of CCA uses one separable filter and three nonseparable filters for the smoothing while the GLM uses one separable filter. For the GPU implementation the 2D smoothing can be done extremely fast by using the shared memory.

4.3. *Verifying the Whitening Procedure.* To verify that the whitening procedure prior to the permutations works correctly, the Ljung-Box test was applied to each residual time series. The Ljung-Box test was applied to the four datasets after detrending, BOLD removal, and whitening with AR models of different order. The test was applied for 1–10 time lags (i.e., 10 tests), and the mean number of nonwhite voxels was saved. A voxel-wise threshold of  $P = 0.05$  was used, that is,  $\chi^2_{0.95, h-p}$  where  $h$  is the number of time lags tested and  $p$  is the order of the AR model used. This means that the test

only can be applied to certain time lags, since the degrees of freedom otherwise become zero or negative. The results with spatial smoothing of the auto correlations are given in Figure 5 and the results without spatial smoothing are given in Figure 6. The results for Gaussian white noise are included as reference when no smoothing is applied to the auto correlations. With the spatial smoothing, the number of voxels classified as nonwhite for the Gaussian noise is always zero.

If no smoothing is applied to the estimated auto correlations prior to the Ljung-Box test, the test statistic cannot be trusted as the standard deviation of the estimated auto correlations is too high. The reason why the number of nonwhite voxels increases, when no smoothing is applied to the auto correlations and when the degree of the AR model increases, is that the critical threshold of the Ljung-Box test decreases as the order of the AR model increases.

From the results in Figures 5 and 6 we draw the conclusion that cubic detrending and an individual AR(4) whitening is necessary to whiten the *Motor 1*, *Motor 2*, and *Language* datasets while an individual AR(5) or AR(6) whitening is necessary for the *Null* dataset. Long-term autocorrelations have previously been reported for resting state fMRI data.

For all the datasets an individual AR(4) whitening was therefore used prior to the permutations and in each permutation to generate new null data. For the *null* dataset a higher-order AR model is necessary, but to estimate an



TABLE 2: Processing times for random permutation tests with the GLM for the different implementations.

Number of permutations with GLM	C	OpenMP	CUDA, 1 × GTX 480	CUDA, 3 × GTX 480
1000	25 min	3.5 min	25.2 s	8.4 s
5 000	2 h 5 min	17.5 min	1 min 42 s	33.9 s
10 000	4 h 10 min	35 min	3 min 18 s	65.8 s
50 000	20 h 50 min	2 h 55 min	16 min 30 s	5 min 30 s
100 000	1 day 17 h 40 min	5 h 50 min	33 min	11 min

TABLE 3: Processing times for random permutation tests with 2D CCA for the different implementations.

Number of permutations with 2D CCA	C	OpenMP	CUDA, 1 × GTX 480	CUDA, 3 × GTX 480
1000	1 h 40 min	14 min 50 s	22.2 s	7.4 s
5 000	8 h 20 min	1 h 14 min	1 min 24 s	28 s
10 000	16 h 37 min	2 h 28 min	2 min 42 s	54 s
50 000	3 days 11 h	12 h 22 min	13 min 30 s	4 min 30 s
100 000	6 days 22 h	24 h 43 min	27 min	9 min

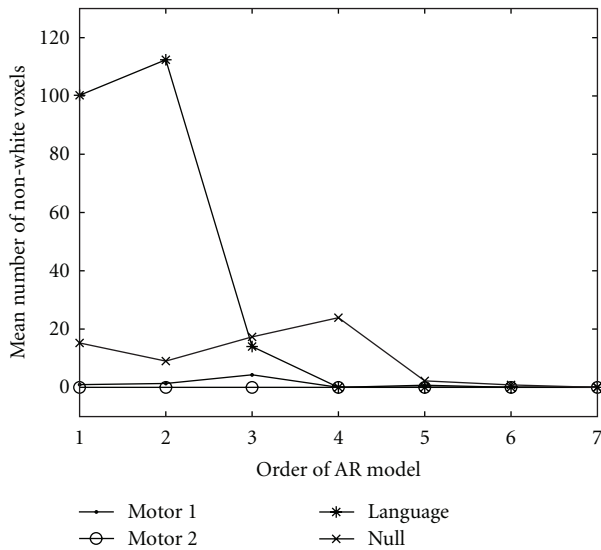


FIGURE 5: Mean number of voxels classified as nonwhite by the Ljung-Box test (1–10 time lags were tested and the mean number of nonwhite voxels for the 10 tests was saved). Prior to the Ljung-Box test the estimated auto correlations were spatially smoothed. The number of nonwhite voxels for Gaussian white noise is always zero.

AR(5) model requires matrix inverses of  $5 \times 5$  matrices for each voxel time series, which our GPU implementation does not support. Therefore, the voxels in the *null* dataset that were considered as nonwhite after the AR(4) whitening were instead removed from the random permutation test.

**4.4. Verifying the Random Permutation Test.** To verify that our random permutation test works correctly, all the pre-processing steps were removed and Gaussian white noise was used as data. The stimulus paradigm convolved with the HRF and its temporal derivative were used as regressors, and a  $t$ -test value was calculated for each voxel. A spatial mask from a real fMRI dataset was used to get the same number of brain

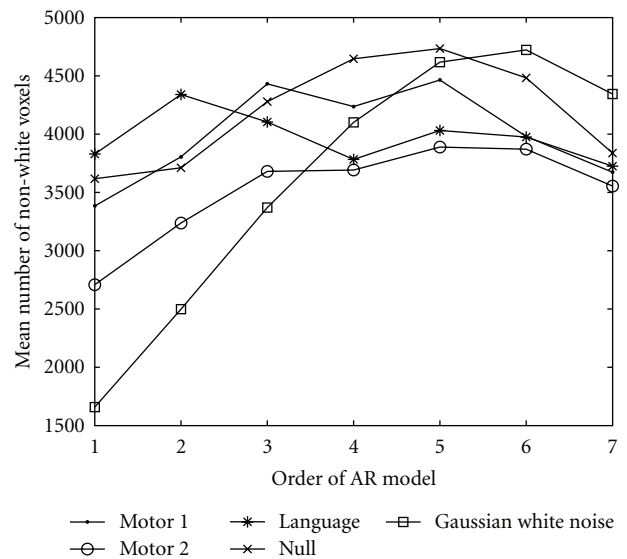


FIGURE 6: Mean number of voxels classified as nonwhite by the Ljung-Box test (1–10 time lags were tested and the mean number of nonwhite voxels for the 10 tests was saved). No spatial smoothing was applied to the estimated auto correlations prior to the Ljung-Box test. The number of nonwhite voxels for Gaussian white noise is included as reference (no whitening was applied to the noise).

voxels. A threshold for corrected  $P = 0.05$  was calculated, by using 100 000 permutations, and then 10 000 noise datasets were generated (for each amount of smoothing), analysed, and thresholded. If the calculated threshold is correct, 500 of the noise datasets should contain a test value that is higher than the threshold. The family-wise error rate (FWE) was estimated for the thresholds from Bonferroni correction, Gaussian random field theory, and the random permutation test and is given in Figure 7.

**4.5. GLM versus CCA.** With the random permutation test it is possible to calculate corrected  $P$  values for fMRI analysis by CCA, and thereby activity maps from GLM and CCA

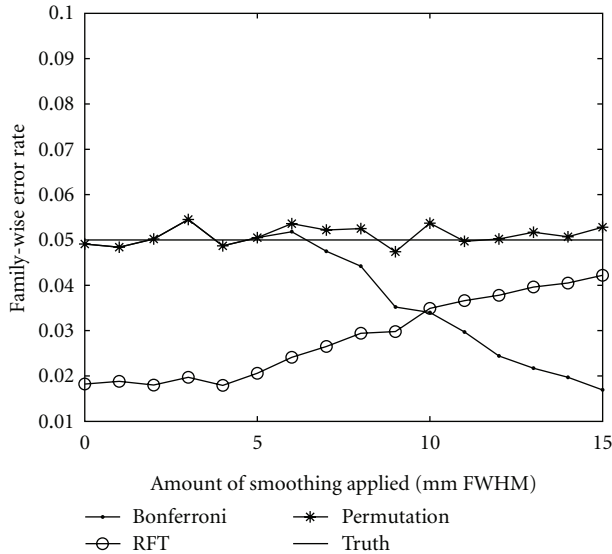


FIGURE 7: A comparison of family-wise error rates for Gaussian white noise for three different approaches to calculate an activity threshold, corrected for multiple testing.

can finally be compared at the same significance level. The activity maps are given in Figure 8. For these comparisons, the *Motor 1* dataset was used, 10 000 permutations were used both for GLM and CCA. The activity maps were thresholded at the same significance level, corrected  $P = 0.05$ . With 8 mm of 2D smoothing, GLM detects 302 significantly active voxels while CCA detects 344 significantly active voxels. With 8 mm of 3D smoothing, GLM detects 475 significantly active voxels while CCA detects 684 significantly active voxels. The aim of this comparison is not to prove that CCA has a superior detection performance, but to show that objective evaluation of different methods for single-subject fMRI analysis becomes practically possible by using fast random permutation tests.

For RCCA there is no theoretical distribution to calculate a threshold from, and therefore the corrected thresholds for the restricted canonical correlation coefficients are also presented, 10 000 permutations were used to calculate each threshold. Figure 9 shows the found thresholds for 2D CCA and 3D CCA for the *Motor 1* dataset. Similar results were obtained for the other datasets. Since fMRI analysis by CCA results in an adaptive smoothing, compared to a fix smoothing with the GLM, the amount of smoothing varies between the voxels. Due to this, the plots show the corrected thresholds for the different *maximum* amounts of smoothing that can be applied by CCA. These plots would have taken a total of about 14 days to generate with a standard C implementation, with our multi-GPU implementation they took about 30 minutes to generate.

**4.6. Comparison of Methods for Calculating Corrected Thresholds.** As the null distribution of the maximum  $t$ -test statistics can be estimated, it is possible to compare the thresholds that

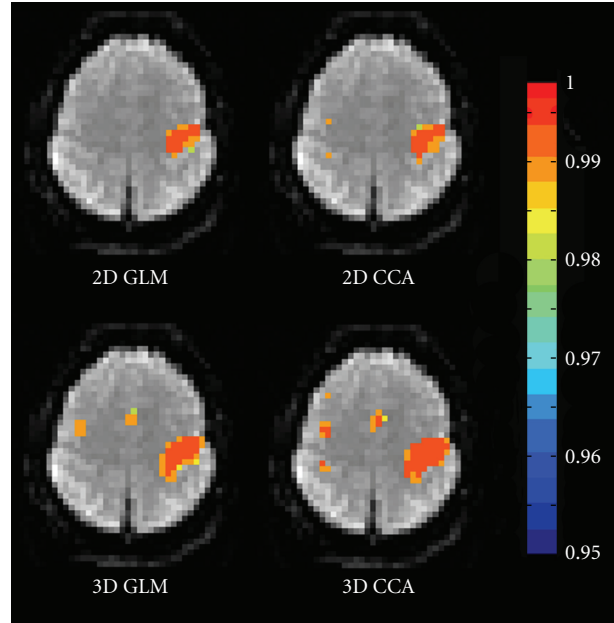


FIGURE 8: **Top:** A comparison between corrected  $P$  values from 2D GLM (left) and 2D CCA (right), calculated from a random permutation test with 10 000 permutations. The activity maps are thresholded at the same significance level (corrected  $P = 0.05$ ). The GLM used one isotropic 8 mm FWHM 2D Gaussian smoothing kernel while CCA used one isotropic 2D Gaussian kernel and 3 anisotropic 2D Gaussian kernels, designed such that the largest possible filter that CCA can create has an FWHM of 8 mm. The neurological display convention is used (left is left),  $1-p$  is shown instead of  $p$ . Note that CCA detects active voxels in the left motor cortex and in the left somatosensory cortex that are not detected by the GLM. **Bottom:** A comparison between corrected  $P$  values from 3D GLM (left) and 3D CCA (right), calculated from a random permutation test with 10 000 permutations. The activity maps are thresholded at the same significance level (corrected  $P = 0.05$ ). The GLM used one isotropic 8 mm FWHM 3D Gaussian smoothing kernel while CCA used one isotropic 3D Gaussian kernel and its derivative, designed such that the largest possible filter that CCA can create has a FWHM of 8 mm. The neurological display convention is used (left is left),  $1-p$  is shown instead of  $p$ . Note that CCA detects active voxels in the left somatosensory cortex that are not detected by the GLM.

are given by Bonferroni correction, Gaussian random field theory, and a random permutation test (which should give the most correct threshold); 10 000 permutations were used for the random permutation test.

Figure 10 shows the found thresholds for the *Motor 1* dataset, for different amounts of smoothing. Similar results were obtained for the other datasets. To our knowledge, a comparison of thresholds from Bonferroni correction, Gaussian random field theory, and a random permutation test has previously only been done for *multi-subject* fMRI [24]. These plots would have taken a total of about 5.5 days to generate with a standard C implementation; with our multi-GPU implementation they took about 38 minutes to generate.

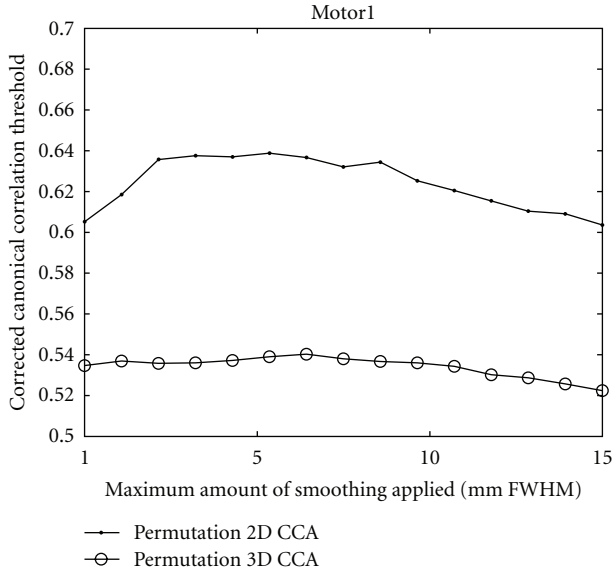


FIGURE 9: Canonical correlation thresholds, for corrected  $P = 0.05$ , as function of the maximum amount of smoothing that can be applied by CCA. The Motor 1 dataset was used.

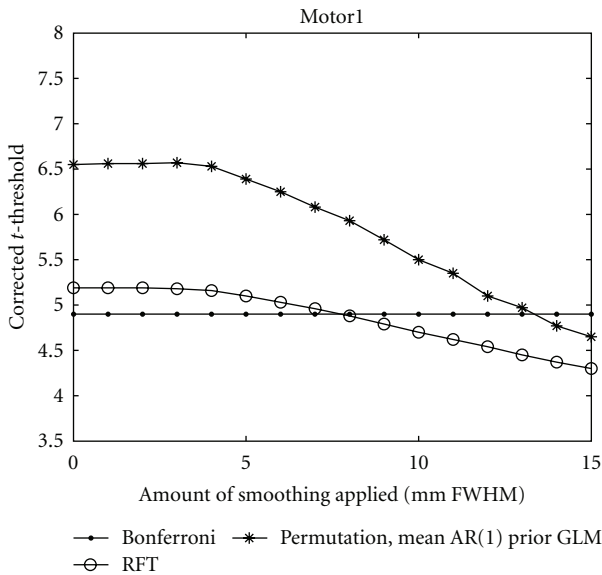


FIGURE 10: A comparison of  $t$ -thresholds, for corrected  $P = 0.05$ , from three approaches to calculate a corrected threshold, as function of the amount of smoothing applied. The Motor 1 dataset was used.

Figure 11 shows the estimated maximum  $t$  distribution for the *Motor 1* dataset; 8 mm of smoothing was applied to the volumes in each permutation.

**4.7. Distributions of Corrected  $t$ -Thresholds.** As a final result, distributions of the corrected  $t$ -thresholds are presented. The random permutation test for the GLM was repeated 1000 times and the resulting thresholds were saved. The *Motor 1* dataset was used with 8 mm of smoothing. The threshold

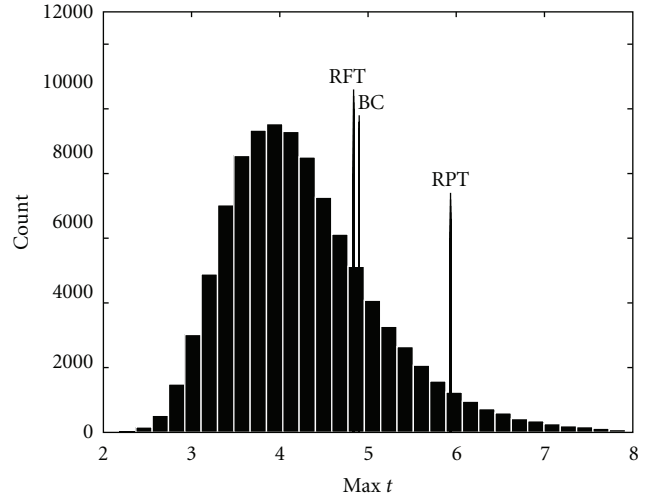


FIGURE 11: The estimated null distribution of the maximum  $t$ -test value from the GLM, when 8 mm smoothing was applied. The calculated thresholds for corrected  $P = 0.05$  for the three approaches are marked with BC (Bonferroni correction), RFT (random field theory), and RPT (random permutation test).

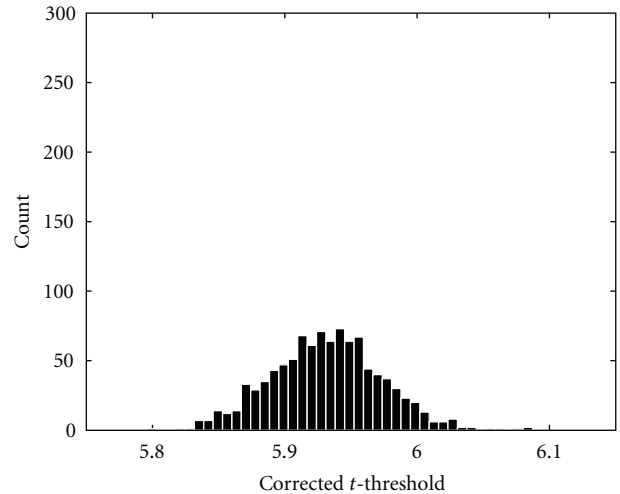


FIGURE 12: The distribution of the corrected  $t$ -threshold when 1000 permutations were used.

distribution for 1000 permutations is given in Figure 12 and the threshold distribution for 10 000 permutations is given in Figure 13. The standard deviation of the threshold calculated with 1000 permutations is 0.0364, and the standard deviation of the threshold calculated with 10 000 permutations is 0.0114. According to [53] the standard deviation should decrease with  $\sqrt{10}$  if 10 times more permutations are used. The difference in standard deviation between the threshold calculated with 1000 and 10 000 permutations is very close to this approximation.

If 1000 permutations are used (and it is assumed that the estimated distribution is correct), the estimated  $P$  value varies between 0.044 and 0.059 if the standard deviation of the corrected threshold is subtracted or added. For 10 000

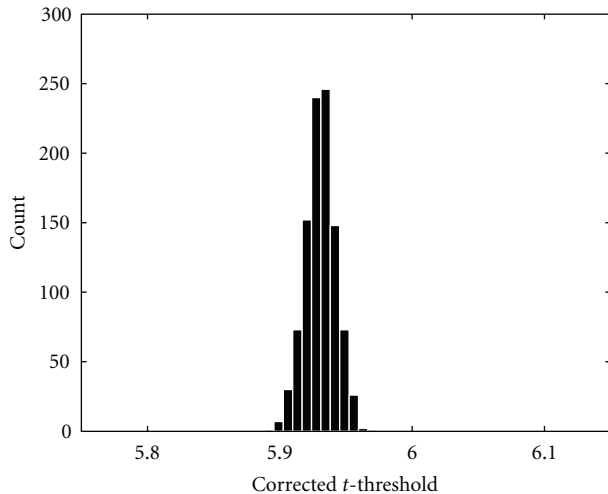


FIGURE 13: The distribution of the corrected  $t$ -threshold when 10 000 permutations were used.

permutations the estimated  $P$  value varies between 0.048 and 0.052. This is close to the expected relative standard deviation given in Table 1. It is very important to know the variance of the  $P$  values as it tells us how reliable the estimates are.

These plots would have taken a total of about 17.4 and 174 days to generate with a standard  $C$  implementation. With the multi-GPU implementation they took about 2.3 and 18 hours to generate.

## 5. Discussion

With the help of fast random permutation tests it is possible to objectively evaluate activity maps from any test statistics easily, by using the same significance level. As an example of this we compare activity maps from GLM and CCA. It is also possible to investigate how a change in the preprocessing (e.g., the amount of smoothing or the whitening applied) affects the distribution of the maximum test statistics. The search for the best test statistics, that gives the best separation between active and inactive voxels, can now be started. To use simple test statistics and hope that the data is normally distributed and independent is no longer necessary.

**5.1. Processing Times.** As can be seen in the tables, a lot of time is saved by using the GPU. Most of the time is saved in the smoothing step. The tables clearly show that the GPU, or an advanced PC cluster, is a must for random permutation tests that include smoothing. To do 100 000 permutations with CCA takes about 7 days with the  $C$  implementation, about a day with the OpenMP implementation, and about 9 minutes with the multi-GPU implementation. The speedup is about 1100 between the  $C$  implementation and the multi-GPU implementation and about 170 between the OpenMP implementation and the multi-GPU implementation.

It should be noted that these processing times are for 80 volumes and 20 000 brain voxels, but it is not uncommon

that an fMRI dataset contains 150 volumes and 30 000 brain voxels, which triples the processing times.

The main problem with a long processing time is the software development. In order to test and verify that a program works correctly, the program has to be launched a large number of times. During the development of the routines and the writing of the paper we ran the complete analysis, with 1000–100 000 permutations, at least 3000 times. For the GLM this means that at least 6000 hours of processing time were saved, compared to the  $C$  implementation. This is equivalent to 750 working days.

With the power of the GPU it is even possible to look at the distributions of the corrected thresholds that otherwise could take as much as 6 months of processing time to estimate.

The processing time for 10 000 permutations with GLM and smoothing is about 3.5 minutes with one GPU. This is perhaps too long for clinical applications, but we believe that it is fast enough for researchers to use it in their daily work.

**5.2. GLM versus CCA.** With the help of the GPU it is finally possible to compare activity maps from GLM and CCA at the same significance level. Even if CCA has a superior detection performance compared to the GLM, its use has been limited. One major reason for this is that it is hard to set a (corrected) threshold for CCA.

The presented activity maps show that the CCA approach in general, due to its spatial adaptivity, finds a higher number of significantly active voxels than the GLM approach. With 2D smoothing CCA finds some significantly active voxels in the left motor cortex and in the left somatosensory cortex that are not detected by the GLM. With 3D smoothing CCA finds some significantly active voxels in the left somatosensory cortex that are not detected by the GLM. We have thereby confirmed previous results that fMRI analysis by CCA can result in a higher number of significantly active voxels [9, 10, 56].

It might seem strange that the corrected canonical correlation thresholds do not decrease as rapidly as the corrected  $t$ -thresholds when the maximum amount of smoothing increases. By using CCA an adaptive smoothing is obtained, such that the amount of smoothing varies between the voxels. The CCA approach will choose the amount (and orientation) of smoothing that results in the highest canonical correlation, as shown in Figure 3. This is one of the main advantages with CCA, since it, for example, prevents that too much smoothing is used in small activity areas. If the maximum canonical correlation is found by only using the small lowpass filter, the maximum canonical correlation might not change significantly when the maximum amount of smoothing is increased since CCA probably will choose to only use the small lowpass filter once again. The consequence is that it is hard to predict how the maximum test value will change as a function of the maximum amount of smoothing.

The corrected thresholds are lower for 3D CCA than for 2D CCA. This is explained by the fact that the 2D version is adaptive in both scale and orientation, and it can thereby find higher correlations than the 3D version that only is adaptive

in scale. With more advanced GPUs, the original 3D CCA approach, with 7 filters, can be used to obtain more spatial adaptivity in 3D.

**5.3. Comparison of Methods for Calculating Corrected Thresholds.** The comparison between the thresholds from Bonferroni correction, Gaussian random field theory, and the random permutation test shows some interesting results. The thresholds from the random permutation test are the highest. For the GLM approach to be valid, the data is assumed to be normally distributed as well as independent. For the multiple testing problem, the parametric approaches also assume a common null distribution for each voxel while the permutation approach does not [24]. There are thus, at least, three sources of error for the parametric approaches.

As a  $t$ -value is calculated for each time series, the normality condition should be investigated for each time series separately [2], for example, by a Kolmogorov-Smirnov test or a Shapiro-Wilk test. These tests are, however, not very reliable if there are only 80 time points for each voxel. The *maximum*  $t$ -distribution is very sensitive to deviations from normality [57], while the standard  $t$ -distribution is rather robust. If a few voxel time series have a distribution that deviates from normality, this is sufficient to affect the maximum  $t$ -distribution and thereby the threshold [24]. This will be captured by the random permutation test but not by the parametric tests.

The distribution of the  $t$ -test values from the *Null* dataset does not strictly follow a Student's  $t$ -distribution, especially if 10 mm smoothing is used. The tails are not longer but slightly thicker than a true  $t$ -distribution. When a mean AR(1) whitening was used for a conventional analysis of the *Null* dataset, the  $t$ -test value that is bigger than 95% of the test values is 1.75 when no smoothing is used, 1.66 when 5 mm of smoothing is used, and 1.59 when 10 mm smoothing is used. The theoretical threshold for uncorrected  $P = 0.05$ , calculated from the Student's  $t$ -distribution, is 1.66. This explains why the thresholds from the random permutation test are higher than the thresholds from Bonferroni correction.

It is commonly assumed that the noise in MRI is normally distributed, but due to the fact that only the magnitude of the MRI data is used, the noise is actually Rician distributed [1]. The original (complex valued) noise in MRI is normally distributed, but the magnitude operation after the inverse Fourier transform in the image reconstruction process is not linear and thereby changes the distribution of the noise. The distribution of the fMRI noise is more complicated as there are several sources of artefacts and the difference between images is used to calculate the test statistics [2–5]. The consequence for fMRI is that the residuals from the GLM might not be normally distributed, even if the model is valid. For the model to be valid, all possible artefacts that can arise have to be modelled. This includes motion-related artefacts, breathing artefacts, pulse artefacts, and MR scanner imperfections. To make a perfect model for all these artefacts is a big challenge on its own.

Another problem for the random field theory approach is that the activity map has to be sufficiently smooth in order to approximate the behaviour of a continuous random field. The smoothness also has to be estimated from the data and it is assumed that it is constant in the brain. These assumptions and several others [24, 39] have to be met in order for the random field theory approach to be valid. For the random permutation test some assumptions also have to be made, for example that the time series are correctly whitened before the permutations. The number of necessary assumptions for the random permutation test is, however, significantly lower than that for the parametric approaches.

**5.4. Future Work.** In this paper we have only described what is known as a *single-threshold* permutation test, but other types of permutation tests can be more powerful. Examples of this are the so-called *step-down* and *step-up* permutation tests. These permutation tests are even more computationally demanding; it can, for example, be necessary to reestimate the maximum null distribution for each voxel. It is also possible to use the mass of a cluster [28, 58] instead of the voxel intensity, or a combination of the voxel intensity and the cluster extent [25]. The random permutation tests can also be used in order to calculate significance thresholds for functional connectivity analysis [31, 32].

The GPU can of course also be used to speed up permutation tests for multi-subject fMRI and multi-subject PET, and not only for single-subject fMRI. The only drawback with the GPU that has been encountered so far is that some test statistics, like 3D CCA, are harder to implement on the GPU than on the CPU, due to the current limitations of the GPU. It must also be possible to calculate the test statistics in parallel, otherwise the GPU will not provide any speedup.

## 6. Conclusions

We have presented how to apply random permutation tests for single-subject analysis of fMRI data by using the graphics processing unit (GPU). Our work enables objective evaluation of arbitrary methods for single-subject fMRI analysis. As a pleasant side effect, the problem of multiple testing is solved in a way that significantly reduces the number of necessary assumptions. To our knowledge, our implementation is the first where the smoothing is done *in each permutation*. In previous papers about permutation tests in fMRI, it is neglected that the smoothing has to be done in each permutation for the analysis to be correct.

## Acknowledgments

This work was supported the Linnaeus center CADICS, funded by the Swedish research council. The fMRI data was collected at the Center for Medical Image Science and Visualization (CMIV). The authors would like to thank the NovaMedTech project at Linköping University for financial support of our GPU hardware and Johan Wiklund for support with the CUDA installations.

## References

- [1] H. Gudbjartsson and S. Patz, "The Rician distribution of noisy MRI data," *Magnetic Resonance in Medicine*, vol. 34, no. 6, pp. 910–914, 1995.
- [2] W. L. Luo and T. E. Nichols, "Diagnosis and exploration of massively univariate neuroimaging models," *NeuroImage*, vol. 19, no. 3, pp. 1014–1032, 2003.
- [3] O. Friman, I. Morocz, and C.-F. Westin, "Examining the whiteness of fMRI noise," in *Proceedings of the ISMRM Annual Meeting*, p. 699, 2005.
- [4] T. E. Lund, K. H. Madsen, K. Sidaros, W. L. Luo, and T. E. Nichols, "Non-white noise in fMRI: does modelling have an impact?" *NeuroImage*, vol. 29, no. 1, pp. 54–66, 2006.
- [5] A. M. Wink and J. B. T. M. Roerdink, "BOLD noise assumptions in fMRI," *International Journal of Biomedical Imaging*, vol. 2006, Article ID 12014, 2006.
- [6] K. J. Friston, O. Josephs, E. Zarahn, A. P. Holmes, S. Rouquette, and J. B. Poline, "To smooth or not to smooth? Bias and efficiency in fMRI time-series analysis," *NeuroImage*, vol. 12, no. 2, pp. 196–208, 2000.
- [7] K. J. Worsley, C. H. Liao, J. Aston et al., "A general statistical analysis for fMRI data," *NeuroImage*, vol. 15, no. 1, pp. 1–15, 2002.
- [8] O. Friman, J. Cedefamn, P. Lundberg, M. Borga, and H. Knutsson, "Detection of neural activity in functional MRI using canonical correlation analysis," *Magnetic Resonance in Medicine*, vol. 45, no. 2, pp. 323–330, 2001.
- [9] O. Friman, M. Borga, P. Lundberg, and H. Knutsson, "Adaptive analysis of fMRI data," *NeuroImage*, vol. 19, no. 3, pp. 837–845, 2003.
- [10] R. Nandy and D. Cordes, "A novel nonparametric approach to canonical correlation analysis with applications to low CNR functional MRI data," *Magnetic Resonance in Medicine*, vol. 49, pp. 1152–1162, 2003.
- [11] J. Mourão-Miranda, A. L. W. Bokde, C. Born, H. Hampel, and M. Stetter, "Classifying brain states and determining the discriminating activation patterns: support Vector Machine on functional MRI data," *NeuroImage*, vol. 28, no. 4, pp. 980–995, 2005.
- [12] N. Kriegeskorte, R. Goebel, and P. Bandettini, "Information-based functional brain mapping," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 10, pp. 3863–3868, 2006.
- [13] F. D. Martino, G. Valente, N. Staeren, J. Ashburner, R. Goebel, and E. Formisano, "Combining multivariate voxel selection and support vector machines for mapping and classification of fMRI spatial patterns," *NeuroImage*, vol. 43, no. 1, pp. 44–58, 2008.
- [14] M. B. Åberg and J. Wessberg, "An evolutionary approach to the identification of informative voxel clusters for brain state discrimination," *IEEE Journal on Selected Topics in Signal Processing*, vol. 2, no. 6, pp. 919–928, 2008.
- [15] Y. Hochberg and A. C. Tamhane, *Multiple Comparison Procedures*, John Wiley & Sons, New York, NY, USA, 1987.
- [16] S. Siegel, "Nonparametric statistics," *The American Statistician*, vol. 11, pp. 13–19, 1957.
- [17] A. P. Holmes, R. C. Blair, J. D. G. Watson, and I. Ford, "Nonparametric analysis of statistic images from functional mapping experiments," *Journal of Cerebral Blood Flow & Metabolism*, vol. 16, no. 1, pp. 7–22, 1996.
- [18] E. Bullmore, M. Brammer, S. C. R. Williams et al., "Statistical methods of estimation and inference for functional MR image analysis," *Magnetic Resonance in Medicine*, vol. 35, no. 2, pp. 261–277, 1996.
- [19] J. J. Locascio, P. J. Jennings, C. I. Moore, and S. Corkin, "Time series analysis in the time domain and resampling methods for studies of functional magnetic resonance brain imaging," *Human Brain Mapping*, vol. 5, no. 3, pp. 168–193, 1997.
- [20] M. J. Brammer, E. T. Bullmore, A. Simmons et al., "Generic brain activation mapping in functional magnetic resonance imaging: a nonparametric approach," *Magnetic Resonance Imaging*, vol. 15, no. 7, pp. 763–770, 1997.
- [21] M. Belmonte and D. Yurgelun-Todd, "Permutation testing made practical for functional magnetic resonance image analysis," *IEEE Transactions on Medical Imaging*, vol. 20, no. 3, pp. 243–248, 2001.
- [22] E. Bullmore, C. Long, J. Suckling et al., "Colored noise and computational inference in neurophysiological (fMRI) time series analysis: resampling methods in time and wavelet domains," *Human Brain Mapping*, vol. 12, no. 2, pp. 61–78, 2001.
- [23] T. E. Nichols and A. P. Holmes, "Nonparametric permutation tests for functional neuroimaging: a primer with examples," *Human Brain Mapping*, vol. 15, no. 1, pp. 1–25, 2002.
- [24] T. Nichols and S. Hayasaka, "Controlling the familywise error rate in functional neuroimaging: a comparative review," *Statistical Methods in Medical Research*, vol. 12, no. 5, pp. 419–446, 2003.
- [25] S. Hayasaka and T. E. Nichols, "Combining voxel intensity and cluster extent with permutation test framework," *NeuroImage*, vol. 23, no. 1, pp. 54–63, 2004.
- [26] M. Breakspear, M. J. Brammer, E. T. Bullmore, P. Das, and L. M. Williams, "Spatiotemporal wavelet resampling for functional neuroimaging data," *Human Brain Mapping*, vol. 23, no. 1, pp. 1–25, 2004.
- [27] O. Friman and C. F. Westin, "Resampling fMRI time series," *NeuroImage*, vol. 25, no. 3, pp. 859–867, 2005.
- [28] L. Tillikainen, E. Salli, A. Korvenoja, and H. J. Aronen, "A cluster mass permutation test with contextual enhancement for fMRI activation detection," *NeuroImage*, vol. 32, no. 2, pp. 654–664, 2006.
- [29] R. Nandy and D. Cordes, "A semi-parametric approach to estimate the family-wise error rate in fMRI using resting-state data," *NeuroImage*, vol. 34, no. 4, pp. 1562–1576, 2007.
- [30] A. Eklund, M. Andersson, and H. Knutsson, "fMRI analysis on the GPU—possibilities and challenges," *Computer Methods and Programs in Biomedicine*. In press.
- [31] D. Gembris, M. Neeb, M. Gipp, A. Kugel, and R. Männer, "Correlation analysis on GPU systems using NVIDIA's CUDA," *Journal of Real-Time Image Processing*, pp. 1–6, 2010.
- [32] A. Eklund, O. Friman, M. Andersson, and H. Knutsson, "A GPU accelerated interactive interface for exploratory functional connectivity analysis of fMRI data," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 1621–1624, 2011.
- [33] A. R. Ferreira da Silva, "A bayesian multilevel model for fMRI data analysis," *Computer Methods and Programs in Biomedicine*, vol. 102, pp. 238–252, 2011.
- [34] I. Shterev, S.-H. Jung, S. George, and K. Owzar, "permGPU: using graphics processing units in RNA microarray association studies," *BMC Bioinformatics*, vol. 11, p. 329, 2010.
- [35] J. L. V. Hemert and J. A. Dickerson, "Monte Carlo randomization tests for large-scale abundance datasets on the GPU," *Computer Methods and Programs in Biomedicine*, vol. 101, no. 1, pp. 80–86, 2011.

- [36] K. J. Friston, P. Jezzard, and R. Turner, "Analysis of functional MRI time-series," *Human Brain Mapping*, vol. 1, no. 2, pp. 153–171, 1993.
- [37] K. J. Friston, A. P. Holmes, K. J. Worsley, J. P. Poline, C. D. Frith, and R. S. J. Frackowiak, "Statistical parametric maps in functional imaging: a general linear approach," *Human Brain Mapping*, vol. 2, no. 4, pp. 189–210, 1994.
- [38] S. J. Kiebel, J. B. Poline, K. J. Friston, A. P. Holmes, and K. J. Worsley, "Robust smoothness estimation in statistical parametric maps using standardized residuals from the general linear model," *NeuroImage*, vol. 10, no. 6, pp. 756–766, 1999.
- [39] R. S. Frackowiak, K. Friston, and C. Frith, *Human Brain Function*, Academic Press, New York, NY, USA, 2004.
- [40] M. Dwass, "Modified randomization tests for nonparametric hypotheses," *The Annals of Mathematical Statistics*, vol. 28, pp. 181–187, 1957.
- [41] A. M. Smith, B. K. Lewis, U. E. Ruttimann et al., "Investigation of low frequency drift in fMRI signal," *NeuroImage*, vol. 9, no. 5, pp. 526–533, 1999.
- [42] O. Friman, M. Borga, P. Lundberg, and H. Knutsson, "Detection and detrending in fMRI data analysis," *NeuroImage*, vol. 22, no. 2, pp. 645–655, 2004.
- [43] A. R. Laird, B. P. Rogers, and M. E. Meyerand, "Comparison of Fourier and wavelet resampling methods," *Magnetic Resonance in Medicine*, vol. 51, no. 2, pp. 418–422, 2004.
- [44] T. Gautama and M. M. Van Hulle, "Optimal spatial regularisation of autocorrelation estimates in fMRI analysis," *NeuroImage*, vol. 23, no. 3, pp. 1203–1216, 2004.
- [45] B. Lenoski, L. C. Baxter, L. J. Karam, J. Maisog, and J. Debbins, "On the performance of autocorrelation estimation algorithms for fMRI analysis," *IEEE Journal on Selected Topics in Signal Processing*, vol. 2, no. 6, pp. 828–838, 2008.
- [46] H. Knutsson and C.-F. Westin, "Normalized and differential convolution: methods for interpolation and filtering of incomplete and uncertain data," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 515–523, June 1993.
- [47] G. M. Ljung and G. E. P. Box, "On a measure of lack of fit in time series models," *Biometrika*, vol. 65, no. 2, pp. 297–303, 1978.
- [48] H. Hotelling, "Relation between two sets of variates," *Biometrika*, vol. 28, pp. 322–377, 1936.
- [49] T. K. Nguyen, A. Eklund, H. Ohlsson et al., "Concurrent volume visualization of real-time fMRI," in *Proceedings of the 8th IEEE/EG International Symposium on Volume Graphics*, pp. 53–60, Norrköping, Sweden, May 2010.
- [50] A. Constantine, "Some non-central distribution problems in multivariate analysis," *Annals of Mathematical Statistics*, vol. 34, pp. 1270–1285, 1963.
- [51] S. Das and P. K. Sen, "Restricted canonical correlations," *Linear Algebra and Its Applications*, vol. 210, no. C, pp. 29–47, 1994.
- [52] O. Friman, "Subspace models for functional MRI data analysis," in *Proceedings of the 2nd IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 1–4, April 2004.
- [53] D. S. Moore, G. P. McCabe, and B. A. Craig, *Introduction to the Practice of Statistics*, W. H. Freeman & Company, 2007.
- [54] Nvidia, *CUDA Programming Guide*, Version 4.0, 2010.
- [55] D. Kirk and W. Hwu, *Programming Massively Parallel Processors, A Hands on Approach*, Morgan Kaufmann, 2010.
- [56] M. Ragnehed, M. Engström, H. Knutsson, B. Söderfeldt, and P. Lundberg, "Restricted canonical correlation analysis in functional MRI-validation and a novel thresholding technique," *Journal of Magnetic Resonance Imaging*, vol. 29, no. 1, pp. 146–154, 2009.
- [57] R. Viviani, P. Beschoner, K. Ehrhard, B. Schmitz, and J. Thöne, "Non-normality and transformations of random fields, with an application to voxel-based morphometry," *NeuroImage*, vol. 35, no. 1, pp. 121–130, 2007.
- [58] E. T. Bullmore, J. Suckling, S. Overmeyer, S. Rabe-Hesketh, E. Taylor, and M. J. Brammer, "Global, voxel, and cluster tests, by theory and permutation, for a difference between two groups of structural MR images of the brain," *IEEE Transactions on Medical Imaging*, vol. 18, no. 1, pp. 32–42, 1999.