

## libFCL (v1.0)

A new MVPA toolbox for brain state classification,  
functional connectivity and feature analysis.

Orhan Firat  
*orhan.firat@ceng.metu.edu.tr*

November 26, 2012

## Contents

1. Introduction.....	3
2. Installation .....	4
3. Algorithm Overview.....	5
Step 1: Load Data.....	6
Step 2: Clustering .....	7
Step 3: Functional Connectivity Analysis .....	8
Step 4: Extract LRF / FC-LRF .....	9
Step 5: Classification .....	10
4. Sample Run .....	11
A. Acknowledgements .....	16
B. License .....	17

# 1. Introduction

This document provides a detailed description of the design and usage of the **libFCL**. It includes an overview of the various components of the library, installation instructions and a sample run using provided toy data along with toolbox. The intended audience is users and developers of the **libFCL**. The document assumes that the user is familiar with MATLAB and has a grasp on Multi-Voxel Pattern Analysis (MVPA) methods using functional Magnetic Resonance Imaging (fMRI) data.

The **libFCL** is a library of research code initially developed to support the Pattern Analysis of Functional Magnetic Resonance Imaging Project (<http://neuro.ceng.metu.edu.tr>) accomplished by the collaboration of Department of Computer Engineering - Middle East Technical University and Department of Psychology - Koç University. The **libFCL** is a new MVPA toolbox for brain state classification, functional connectivity and feature analysis.

The library is developed and tested under 64-bit Windows environment using MATLAB 7.11.0.584 (R2010b) and is actually a collection of software libraries (see Appendix A) some of which are third-party open source projects including **Functional Connectivity Toolbox**, **Parallel Spectral Clustering**, **libSVM**. We have released the library under the BSD license (see Appendix B).

Developed by Orhan Firat,  
Department of Computer Engineering,  
Middle East Technical University, Turkey

Contact information:  
[orhan.firat@ceng.metu.edu.tr](mailto:orhan.firat@ceng.metu.edu.tr)

## 2. Installation

I. Download the toolbox archive file ([libFCL.rar](#)) and unzip it to anywhere in your file system (ex: 'C:/libFCL/').

II. Run MATLAB. Either,

- Add libFCL path to MATLAB search path manually:

Type

```
>>addpath('C:/libFCL/');
```

means the path of libFCL on the machine

or

- Go to the folder where you extracted the libFCL using MATLAB “current folder” browser.

III. Call script *start\_libFCL.m* to start libFCL,

Type

```
>>start_libFCL;
```

This script automatically adds all the folders under the libFCL directory to the MATLAB path and calls the opening figure of toolbox (see Figure 1 Starting the libFCL below).

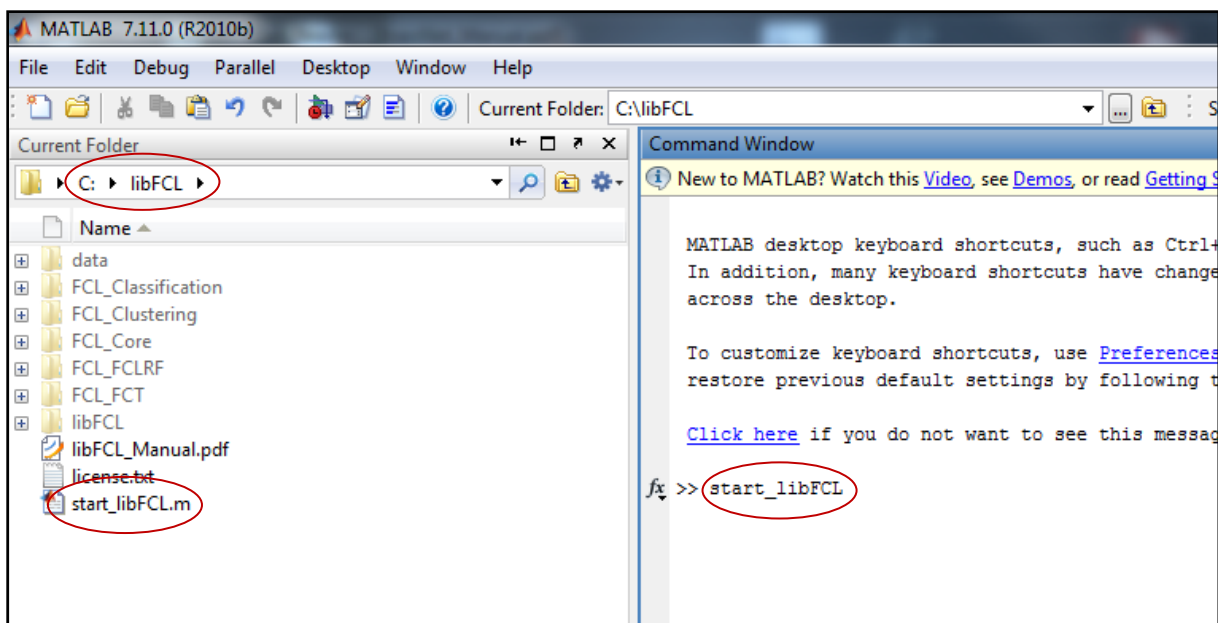


Figure 1 Starting the libFCL

### 3. Algorithm Overview

After successfully starting **libFCL**, an opening panel will popup. The libFCL consists of five consecutive panels that you start with loading data and corresponding labels and completing a classification task in the fifth step.

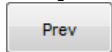

The very first panel of the **libFCL** is the data loading panel. In data loading panel, training data, class labels for training data, test data and class labels for test data is expected to be loaded.

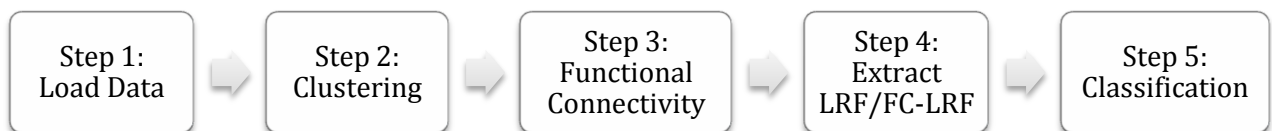
The second panel of the **libFCL** is the clustering panel where user is expected to load 3D-voxel locations file and partition the region of interest into a predefined number of clusters.

Functional connectivity is calculated in the third panel of the **libFCL**. This step calculates functional connectivity within clusters if conducted in the second step or assumes all the voxels in the region of interest form a huge cluster and calculates functional connectivity within one cluster.


Local Relational Features (LRF) or Functional Connectivity Aware Local Relational Features (FC-LRF) (if functional connectivity is pre-calculated in the third step) are extracted in the fourth step using training and test data loaded in the first step. The **libFCL** uses linear prediction filter coefficients (lpc) function of the MATLAB-Signal Processing Toolbox in order to determine the coefficients of a forward linear predictor by minimizing the prediction error in the least squares sense. Therefore, make sure Signal Processing Toolbox is available in your MATLAB.

The final step of the **libFCL** is the classification step using features extracted in the fourth step. Two well-known classification methods are provided for the use via toolbox namely, Support Vector Machine (using libSVM implementation) and k-nearest neighbors (using MATLAB-Bioinformatics Toolbox).

The major transition pattern of the **libFCL** is illustrated in Figure 2 Panel Transition Pattern of libFCL below and also the current step of the algorithm is indicated as a top frame banner in each panel during execution. Transition between panels is done by the  and  buttons on the bottom-right of each panel.



**Figure 2 Panel Transition Pattern of libFCL**

It is also possible to reset the current list, radio button or check box selections by using  button on the bottom-right.

Each panel has an algorithm description text-box as the right-most column frame which explains the current step and panel usage.

## Step 1: Load Data

The first panel of the **libFCL** is data definition panel for the overall algorithm. This panel consists of three frame columns, input data and labels loading boxes on the left, input data description boxes in the middle and algorithm description box on the right (see Figure 3 Load Data Panel of libFCL below).

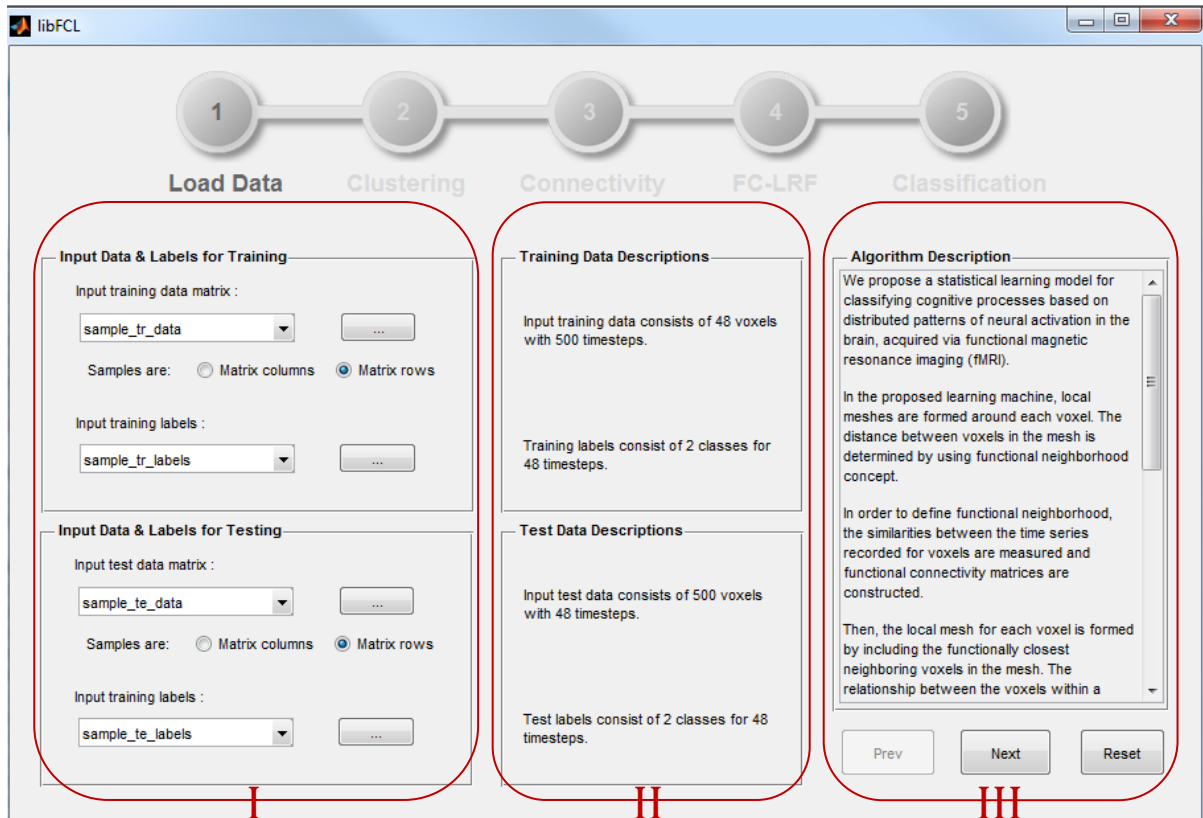



Figure 3 Load Data Panel of libFCL

User has to specify training data and corresponding class labels files. Training data must be a matrix of  $N \times M$  or  $N \times M$  where  $N$  is the total number of time steps and  $M$  is the total number of voxels in the region of interest. The class labels can be specified in two different ways, first by an  $N \times C$  matrix where  $C$  is the number of classes. When the  $i^{th}$  row and  $j^{th}$  column is 1 that states  $i^{th}$  time point is in  $j^{th}$  class and the rest of the  $i^{th}$  row is all zero. The second format for class labels is using a  $N \times 1$  vector representation where each non-zero element of the vector indicates the corresponding class label. User can transpose data matrices by using radio buttons to indicate samples are distributed either in the rows or in the columns. For details of the data representations user may find it informative to examine sample data provided with the toolbox under the folder "libFCL/data/".

The middle frame consists of two description boxes that are designed to inform user about how **libFCL** translates the inputs and/or selections on the left frame column. User do not have to specify test data and test labels but for training.

## Step 2: Clustering

The second panel of the **libFCL** is the clustering panel. Clustering is basically the task of assigning a set of objects (voxels in our case) into groups (called clusters) so that the objects in the same cluster are more similar (anatomically closer in our case) to each other than to those in other clusters. The main advantage of clustering the region of interest along anatomical locations is relaxation of the computational cost of functional connectivity which will be conducted in the third step. The clustering provided in the toolbox is a generic clustering algorithm and is not enforced by the overall algorithm. User can also load a pre-defined clustering result by using  button, in which loaded clustering file may indicate functional clusters or anatomical region of interests that groups voxels (see Figure 4 Clustering Data Panel of libFCL below).

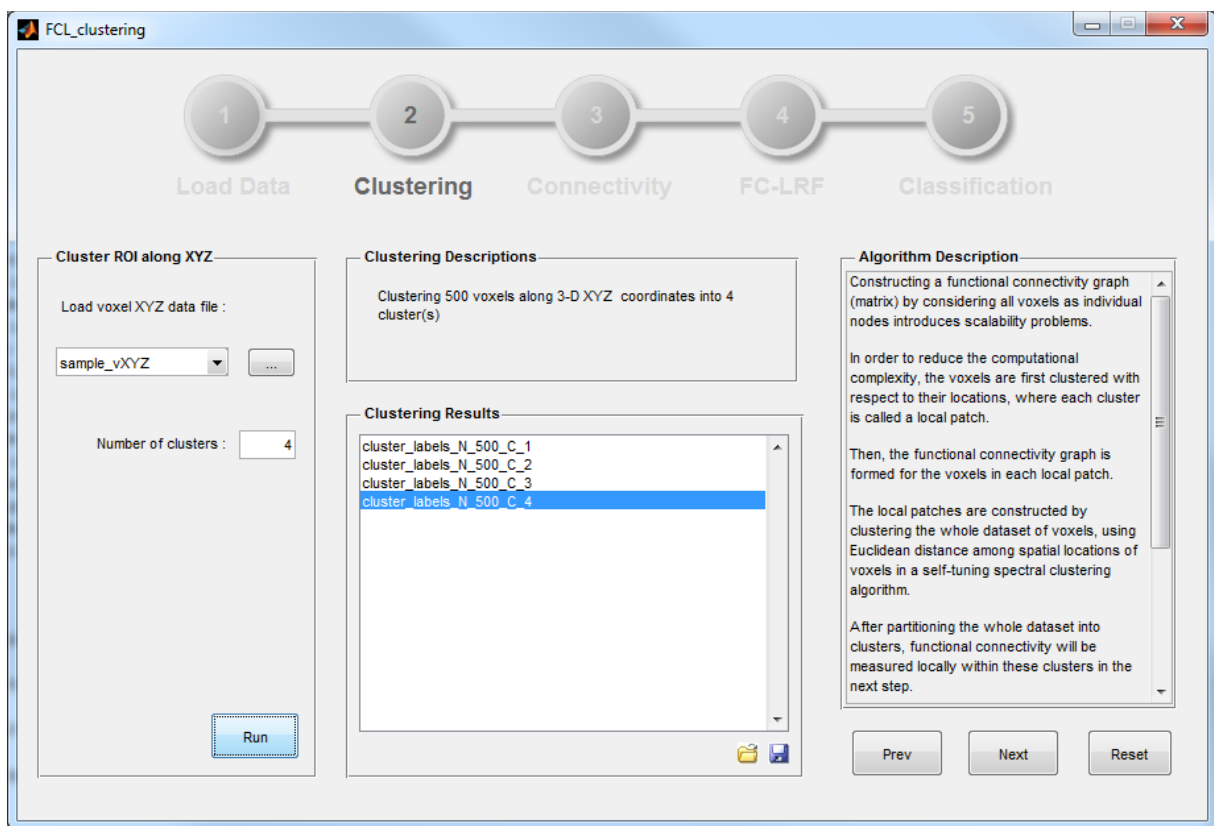



Figure 4 Clustering Data Panel of libFCL

This panel also consists of three frame columns, voxel positions loading and clustering parameters on the left, input data description box and clustering results list in the middle and algorithm description box on the right. User should take into account that the time required by the clustering algorithm grows rapidly as the number of voxels increases, though provided clustering algorithm is tested up to 80.000 voxels.

In order to make it possible to use clustering results in the future, middle panel allows user to save clustering results to disk by using  button in the middle frame below clustering results list. The selected clustering results will be carried to and further used in the next panel (step 3). Note that “Clustering Results” list allows user to select multiple clustering results.

### Step 3: Functional Connectivity Analysis

The third panel of the **libFCL** is the within-cluster functional connectivity analysis panel. Functional connectivity captures deviations from statistical independence between distributed and often spatially remote neuronal units (voxels in our case). Statistical dependence may be estimated by measuring correlation or covariance. The libFCL provides three correlation variants as a measure of connectivity, namely, pearson correlation, peak correlation and scan correlation. User may refer to the **Functional Connectivity Toolbox** for details of the provided correlation measures.

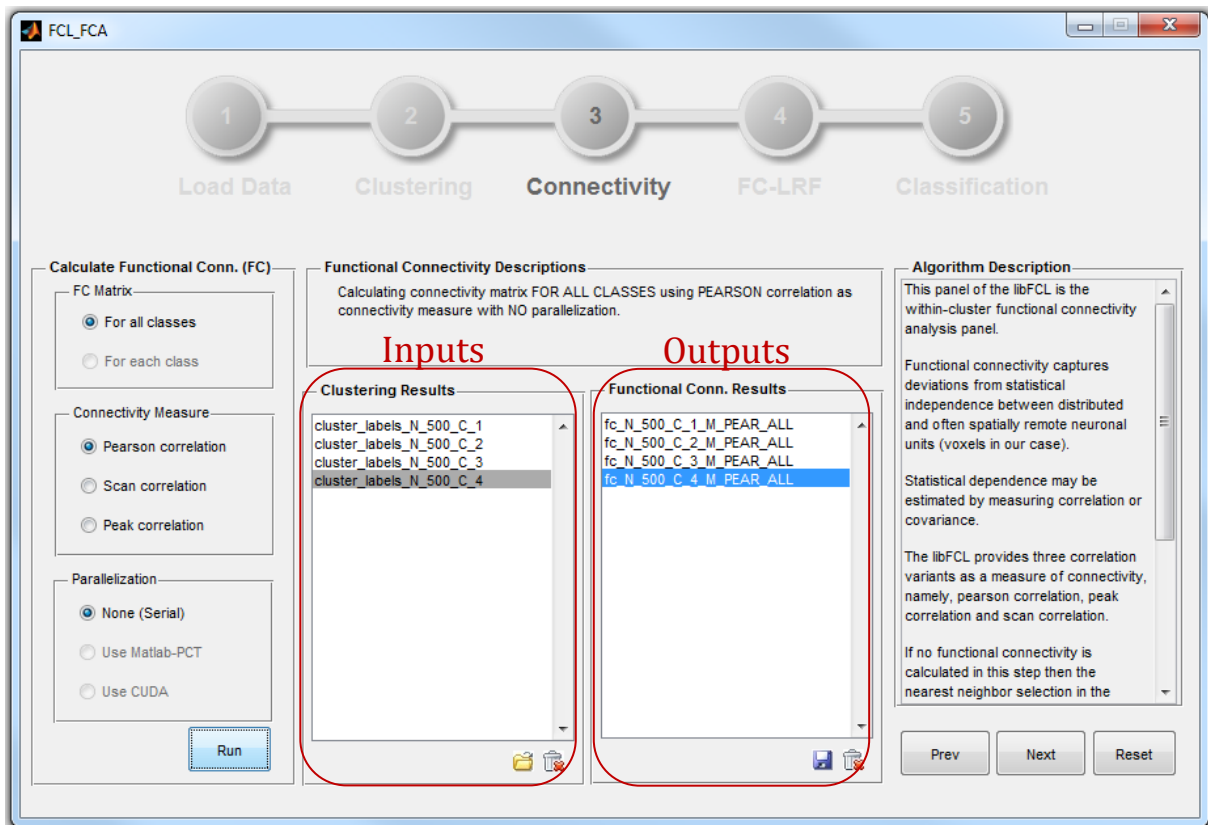



Figure 5 Functional Connectivity Analysis Panel of libFCL

This step is also not enforced by the overall algorithm. If no functional connectivity is calculated in the third step then the nearest neighbor selection in the fourth step will be conducted according to the voxel positions provided in the second step.

Different from the first two panels, middle frame of the third panel comprises two lists “Clustering Results” list on the left functions as input list and “Functional Connectivity Results” list on the right functions as output list. This structure allows user to analyze several inputs by either loading from file or calculating in the previous step and obtaining corresponding outputs for functional connectivity all compactly in a single panel. Resulting functional connectivity matrices will be carried to the fourth step if they are selected in the “Functional Connectivity Results” list. Note that “Functional Connectivity Results” list also allows multiple selections (see Figure 5 Functional Connectivity Analysis Panel of libFCL above).



The disabled buttons and their functionalities are expected to be available in the future versions of **libFCL**. A new button  is introduced in this panel which clears the list when clicked. User should be aware of deleting unsaved results may cause recalculating corresponding results.

#### Step 4: Extract LRF / FC-LRF

Feature extraction is one the most important step in the classification tasks and the fourth step of **libFCL** completes is responsible for feature extraction. User may select three different features, LRF, FC-LRF using positive correlation and FC-LRF using negative correlation. In order to extract LRF user is expected to load voxel positions in the second step and for FC-LRF options user is expected to calculate functional connectivity in the third step.

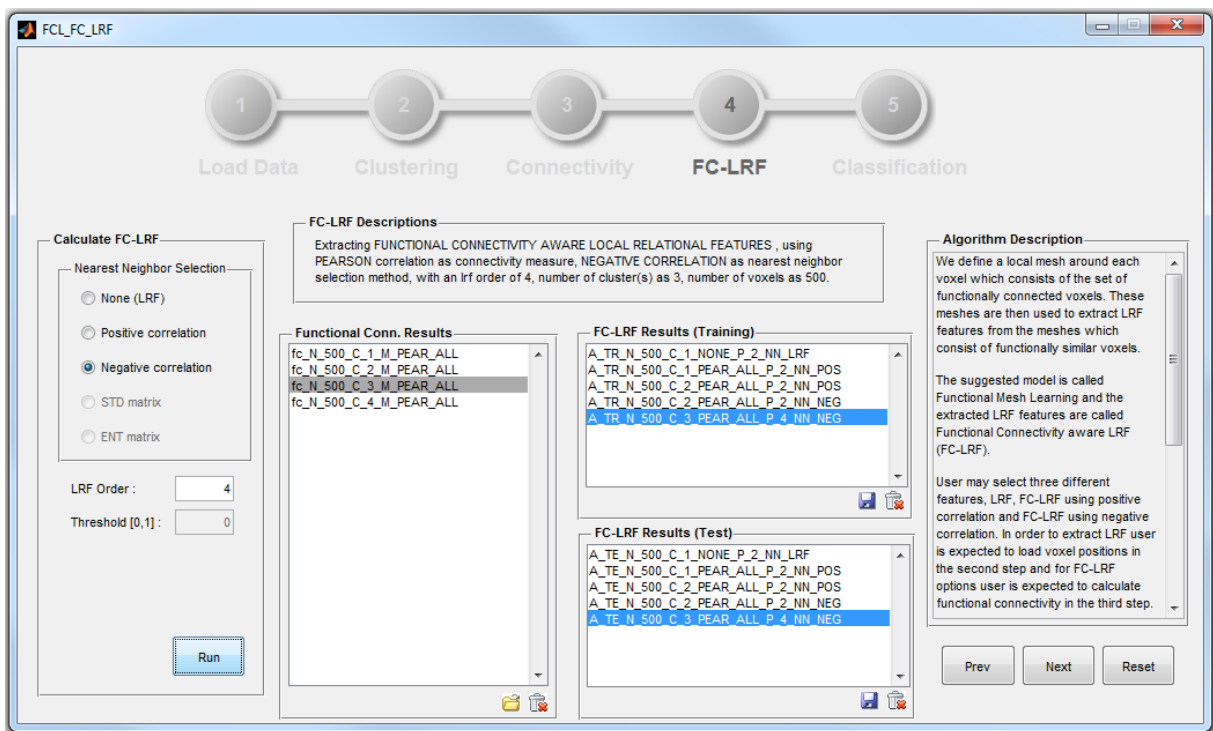


Figure 6 Feature Extraction Panel of **libFCL**

Similar to the convention of third panel, middle frame comprises lists “Functional Connectivity Results” list on the left functions as an input list and “FC-LRF Results (training) / (test)” lists on the right functions as output lists. Features are extracted according to the loaded data for training and test in the first panel of **libFCL**. Note that loading test data and labels is optional in the first panel and therefore determines the extraction of test features in feature extraction step. LRF Order parameter on the left frame column, determines the size of a mesh formed around each voxel and must be between 1 and minimum number of voxels in all clusters. The disabled buttons and their functionalities are expected to be available in the future versions of **libFCL**.

User is provided with the same load, save and clear operations for the input/output lists also multiple selections are allowed for the output lists to transfer feature extraction results to the final step, the “Classification” step, of the **libFCL**.

## Step 5: Classification

The **libFCL** encapsulates two well known classification methods, k-nearest neighbor and support vector machine. When user selects training data to train a classifier and test data in order to generalization test, and clicks the run button corresponding parameter menu for selected classifier will popup. After specifying classifier parameters (or leaving it as default) popup menu will close and classifier runs.

Results of the classification are summarized in the “Status Summary” box. Performance measures used by **libFCL** are precision, recall and f-score. User can also visualize the classification results on a confusion matrix by clicking the corresponding button in the “Performance Results” box. Save button at the bottom right of this box is also allows user to save class labels predicted by the classifier to the disk.

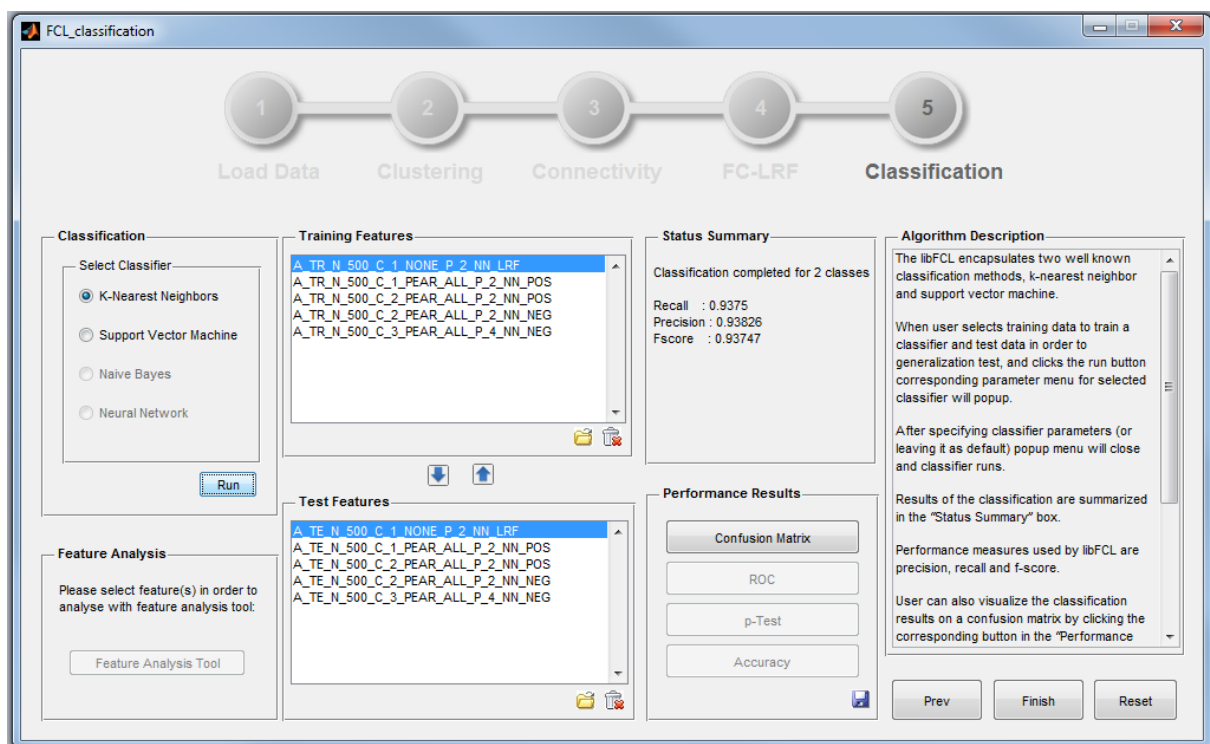


Figure 7 Classification Panel of libFCL

User is encouraged to tune parameters of classifiers by providing training and cross validation sets to “training features” list and “test features” list respectively before actually loading test features, as a best practice in machine learning. The disabled buttons and their functionalities are expected to be available in the future versions of **libFCL**.

Note that SVM option of **libFCL** uses libSVM library, provided SVM mex files are compiled under a 64-bit Windows environment, Linux or MAC-OS users should compile corresponding mex files under ‘/libFCL/FCL\_Classification/libsvm/matlab/’ directory or just type following line on libFCL root:

```
>> run('C:\libFCL\FCL_Classification\libsvm\matlab\make.m')
```

## 4. Sample Run

After installing and starting **libFCL** user is introduced with the first panel, data loading panel. **libFCL** provides a sample data located under directory “libFCL/data/”. Data for training and test along with class labels are loaded using this panel (see Figure 8 Loading Sample Data below).

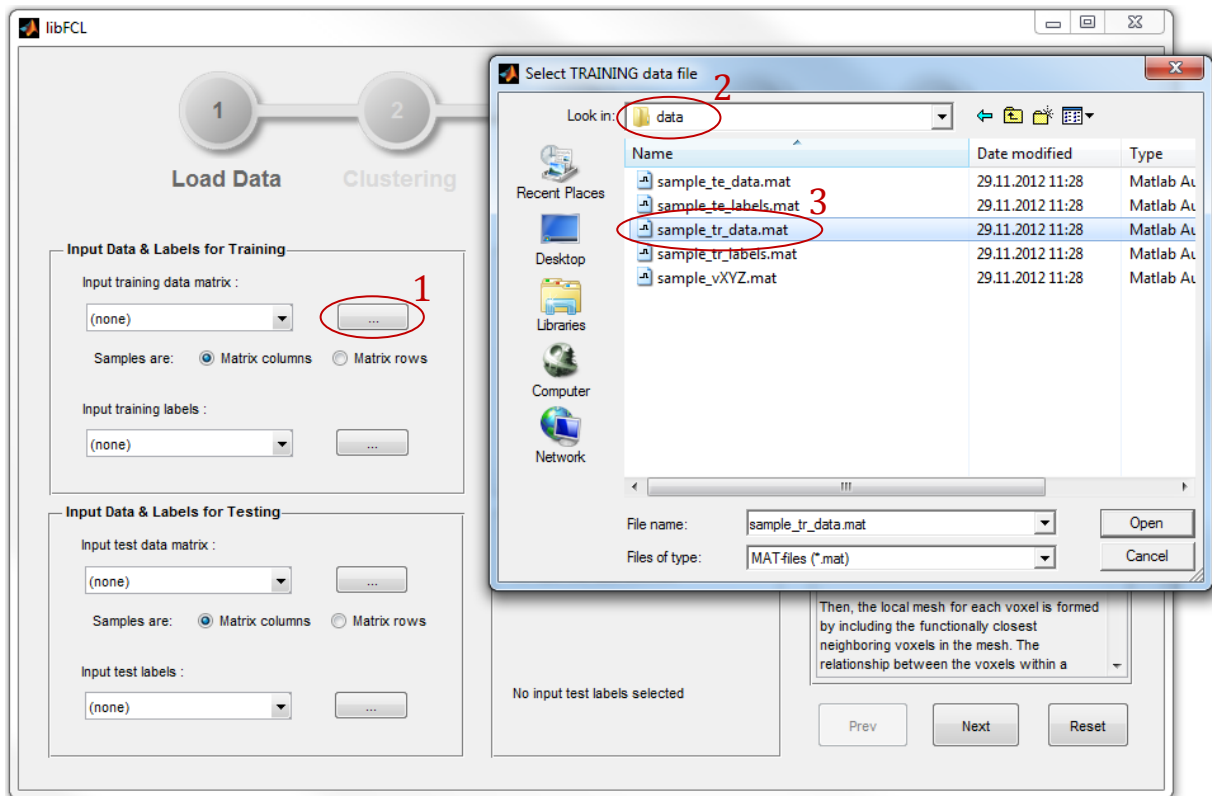
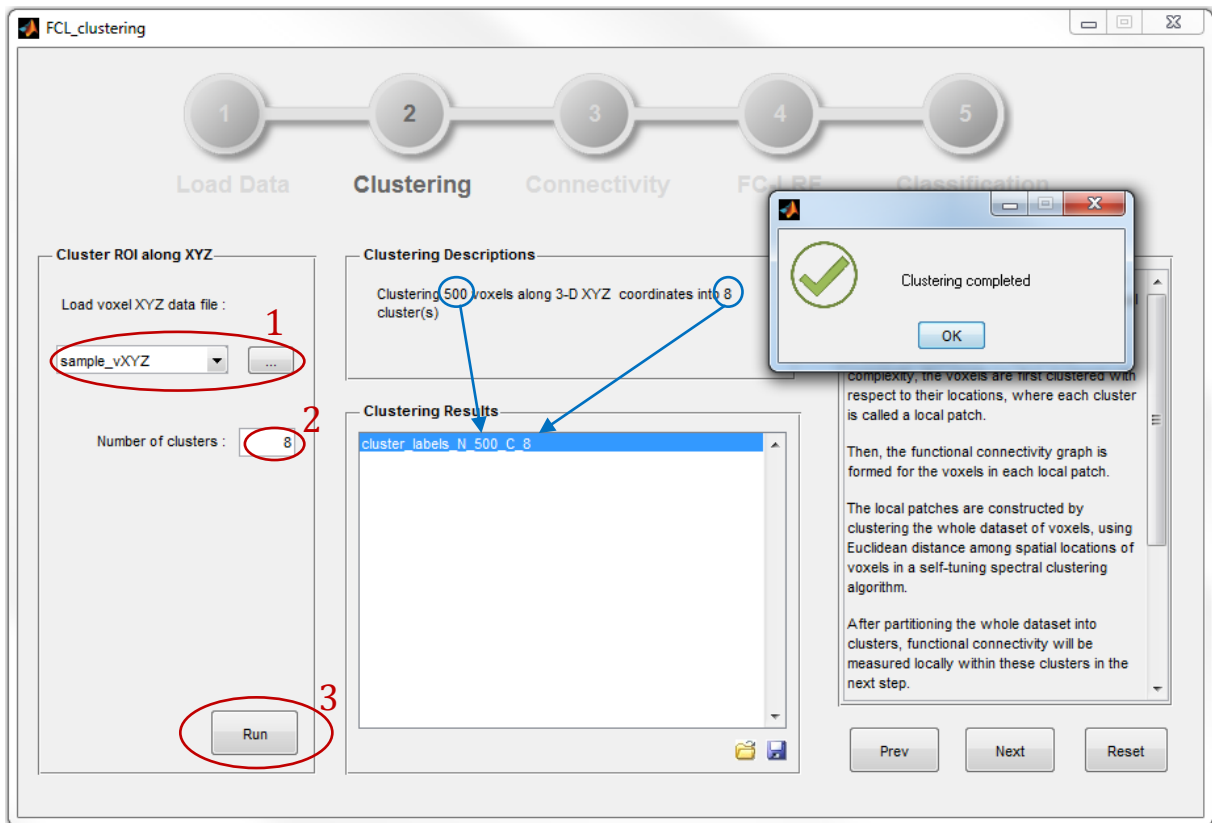


Figure 8 Loading Sample Data

After successfully loading provided training and test data along with corresponding class labels, user has to check middle column for **libFCL** translation. Provided data samples are distributed in matrix rows therefore “Matrix rows” radio button must be selected for both training and test data matrices (see Figure 3 Load Data Panel of libFCL above). Clicking next button ignites several validations and user is prompted with warnings or errors when loading data is not successful.

Second panel of **libFCL** will be opened when all validations checks were successful. The **libFCL** waits loading voxel position file and specifying cluster number. This step is conducted same as the first step, provided “sample\_vXYZ.mat” should be selected for loading. Set number of clusters as 8 and click run. Successful completion of clustering will be prompted with a dialog popup and result of the clustering will be loaded to the “Clustering Results” list (see Figure 9 Clustering Region of Interest Using Sample Voxel Positions below). File name of clustering result indicates number of voxels in the region of interest as N\_\* and number of clusters C\_\*. For example *cluster\_labels\_N\_500\_C\_8* means that number of voxels is 500 and number of cluster is 8.



**Figure 9 Clustering Region of Interest Using Sample Voxel Positions**

User may repeat this procedure multiple times with differing number of clusters and select multiple clustering results then click next button to transfer results for connectivity analysis.

After clicking next button Functional Connectivity Analysis panel will be opened and selected clustering results are transferred to the input list (Clustering Results list) of the third panel. In order to initiate a functional connectivity analysis, first select the clustering result from the “Clustering Results” list, then choose correlation measure and click run. If all the input validations are accomplished connectivity algorithm will start and user will be prompted the successful completion of functional connectivity analysis routines. The result of the functional connectivity analysis will be loaded into the “Functional Connectivity Results” list. File name of functional connectivity result indicates number of voxels in the region of interest as  $N_*$ , number of clusters  $C_*$ , correlation measure  $M_*$  and a trailing  $_{ALL}$  indicating connectivity matrix is calculated using time steps from all classes. For example  $fc\_N\_500\_C\_8\_M\_PEAR\_ALL$  means that number of voxels is 500, number of cluster is 8, correlation measure used is pearson correlation and connectivity matrix is calculated using time steps from all classes.

The functional connectivity analysis will be followed by feature extraction. Click next in order to start feature extraction.

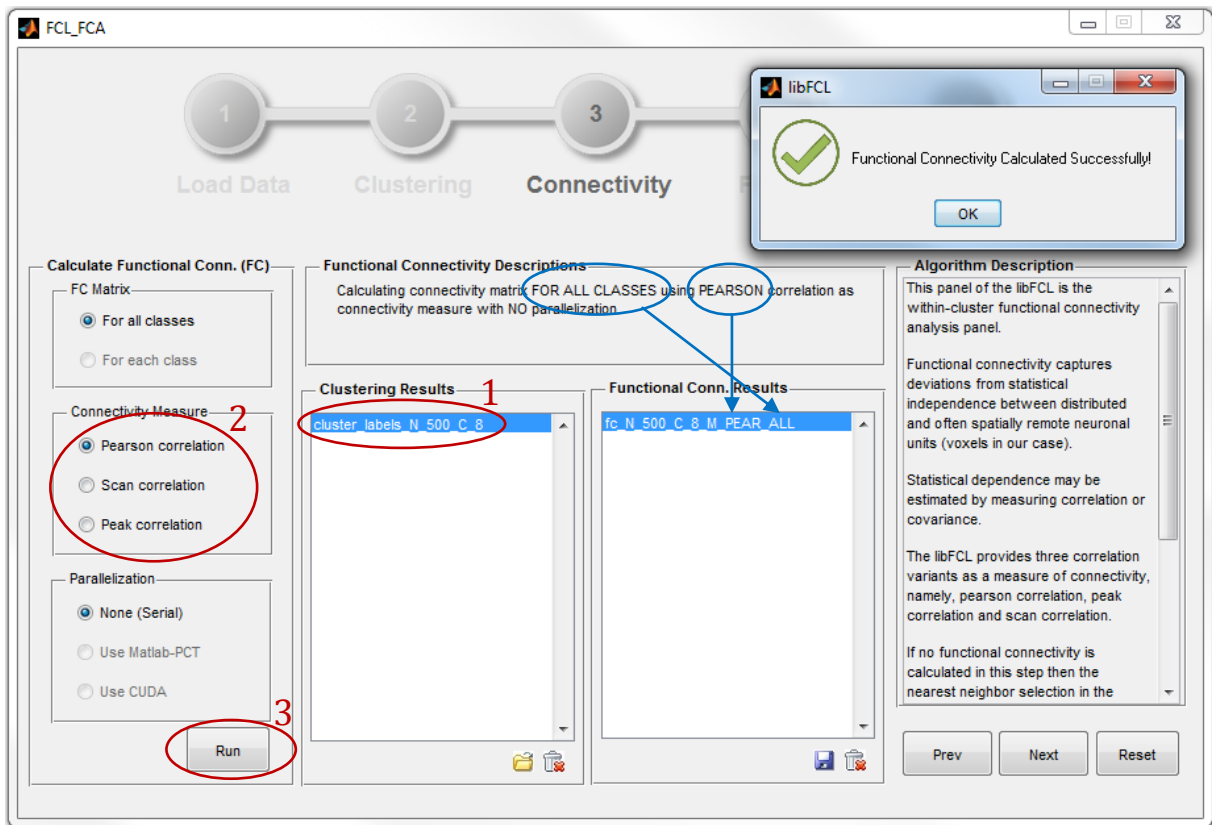


Figure 10 Calculating Functional Connectivity using Sample Data

The feature extraction panel will be started with “Functional Connectivity Results” list is filled with the selected functional connectivity analysis results in the previous step. In order to extract features for training and test first choose one the functional connectivity results, then select one of the nearest neighbor selection algorithm, set LRF order a non-zero integer and click run button. When all the input validations are accomplished feature extraction algorithm will start and user will be prompted the successful completion. The result of the extracted features will be loaded into the “FC-LRF Results (Training)” and “FC-LRF Results (Test)” lists. Repeat this process by changing LRF order, nearest neighbor selection algorithm and input functional connectivity results and transfer selected features to the classification panel. Note that when using LRF as nearest neighbor selection algorithm, **libFCL** does not use functional connectivity results or clustering results, it selects nearest neighbor only considering the Euclidean distance between voxels. Therefore resulting feature filenames will have cluster number as one. File name of extracted features indicate number of voxels in the region of interest as  $N_*$ , number of clusters  $C_*$ , correlation measure  $_{NONE}$  for raw LRF,  $PEAR\_ALL$  for pearson correlation,  $PEAK\_ALL$  for peak correlation,  $SCAN\_ALL$  for scan correlation, LRF order  $P_*$  and nearest neighbor selection algorithm  $NN\_LRF$  for raw LRF,  $NN\_POS$  for positive correlation based neighbor selection,  $NN\_NEG$  for negative correlation based neighbor selection. For example  $A\_TR\_N\_500\_C\_8\_M\_PEAR\_ALL\_P\_4\_NN\_POS$  means that number of voxels is 500, number of cluster is 8, correlation measure used is pearson correlation and connectivity matrix is calculated using time steps from all classes, FC-LRF order is 4 and nearest neighbor is selected using positive correlation (see Figure 11 Feature Extraction Using Sample Data below).

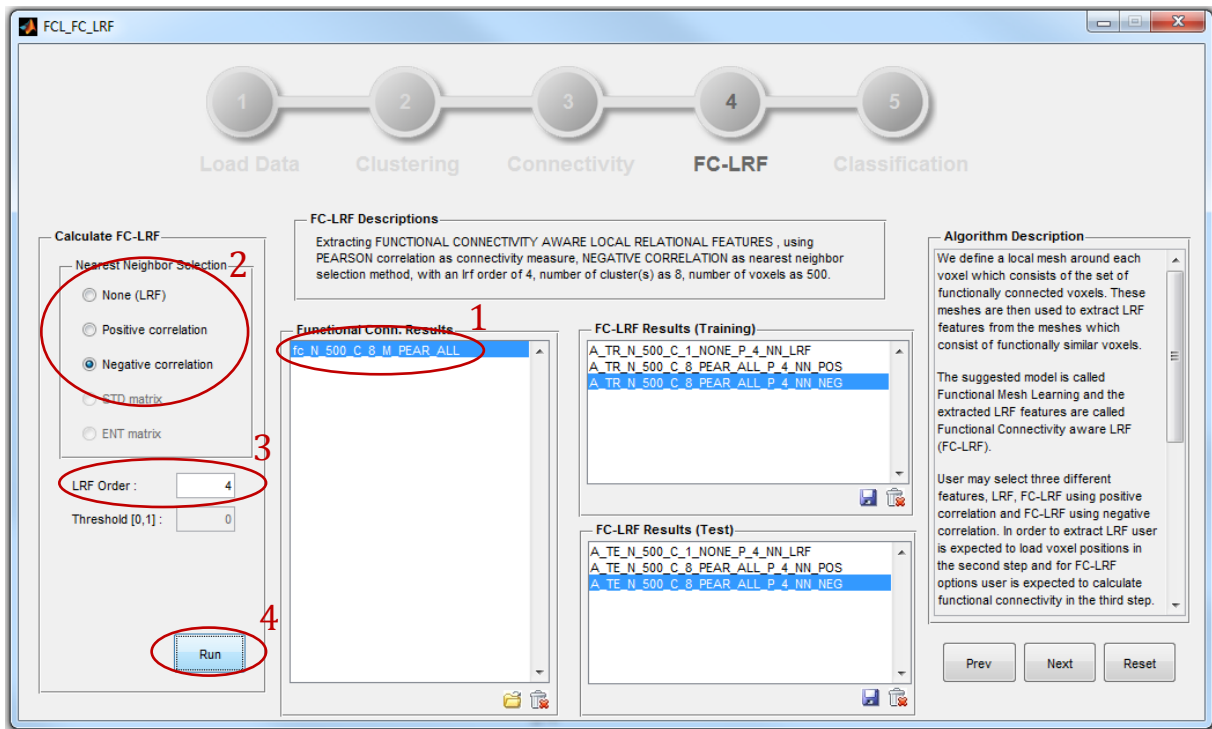


Figure 11 Feature Extraction Using Sample Data

Selecting extracted features and clicking next button will open up the final panel of the **libFCL** where “Training Features” and “Test Features” lists are loaded with features selected in the previous step. In order to conduct a classification task first select one item from the “Training Features” list and one item from “Test Features” list, then choose a classifier and click run. User will be prompted to enter classifier specific parameters, use them as default and click run (see Figure 12 Classification Using Sample Data below).

The results will be updated in the “Status Summary” box with performance measures as precision, recall and f-score. Clicking the “Confusion Matrix” button draws the confusion matrix for the current classification result.

The finish button closes the figure and exits the **libFCL**.

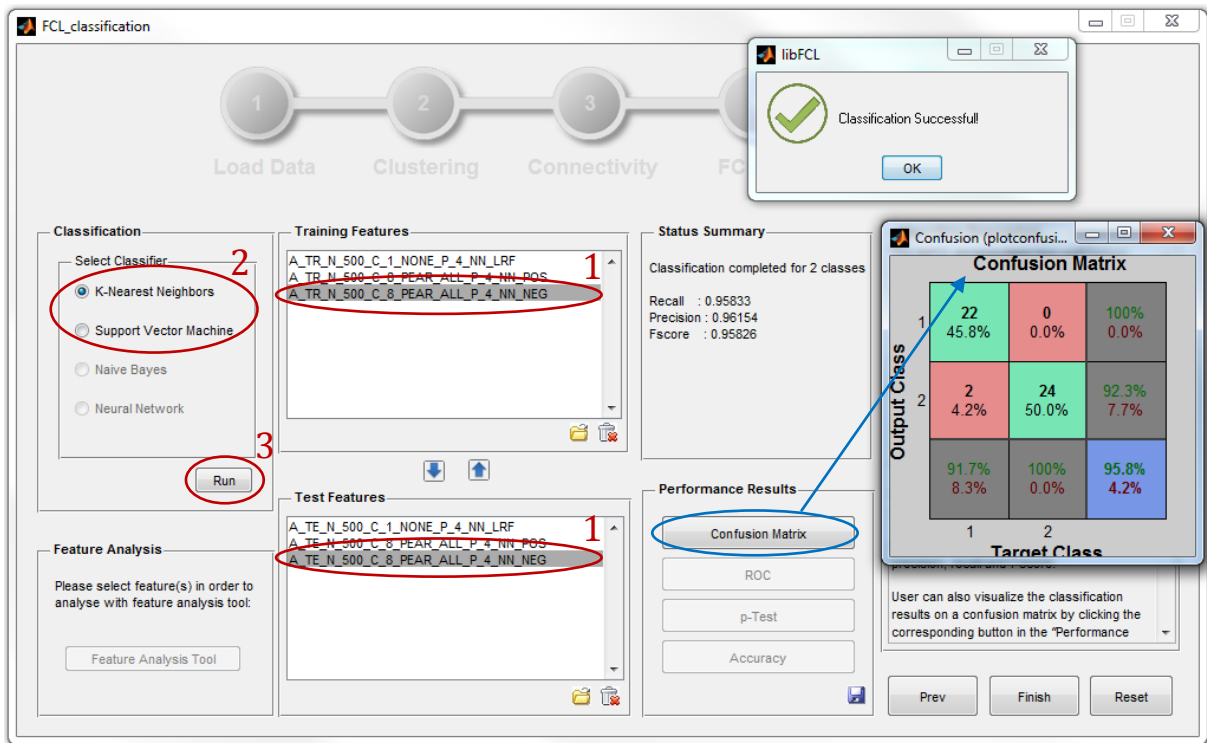


Figure 12 Classification Using Sample Data

## A. Acknowledgements

We thank the developers of the following softwares and toolboxes whose source code or file formats were referenced during our package development:

MATLAB:

<http://www.mathworks.com/products/matlab/>

Functional Connectivity Toolbox:

<https://sites.google.com/site/functionalconnectivitytoolbox/>

libSVM:

<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

Parallel Spectral Clustering:

<http://alumni.cs.ucsb.edu/~wychen/sc.html>

This work was partially supported by the Science and Technological Research Council of Turkey (TÜBİTAK).



## B. License

Copyright (c) 2012, Orhan FIRAT  
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution
- \* Neither the name of Middle East Technical University and Koç University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.