INSTRUCTIONS for Division of Cortical Labels Using divide_parcellation.py

Date: 13/7/2015
Institution: NBE, Aalto University
Author: Laitio, P. <pekka.laitio@aalto.fi>

The algorithm divides cortical labels from Freesurfer annotation file into smaller sublabels using MNE Python and PySurfer packages [1,2]. Both PCA and region growing (RG) algorithms with some homemade developments are utilized in the process. The region growing algorithm is partly modified to correspond to the algorithm introduced by L. Cammoun et al in [3].

1. Installation

The NBE-distributed Python module is **divide_parcellation.py**. Required third party packages are listed in Table 1. It is recommended to first install Canopy Enthought environment (www.enthought.com/products/canopy) and then install PySurfer (pysurfer.github.io) and MNE Python (martinos.org/mne/stable/mne-python) separately. All the other packages are included in Canopy Enthought (also those required for MNE with Python).

Package name	Additional information
numpy	
scipy	
ipython	
nibabel	
mayavi	
matplotlib	
mne	Not included in Canopy Enthought
pysurfer	Not included in Canopy Enthought

Table 1: Required third-party Python packages

2. Methods

2.1. Overview

The key function is **split_annotation** in divide_parcellation.py, where all the parameters needed are explained after each function. The first and most important decision is whether to use PCA or region growing (RG) algorithm in the subdivision. Function parameters are listed in Table 5. The pipeline follows:

1. Divide the cortical labels using alternatively
   (i) PCA, or
   (ii) RG.
2. Smooth the subdivided labels with RG.

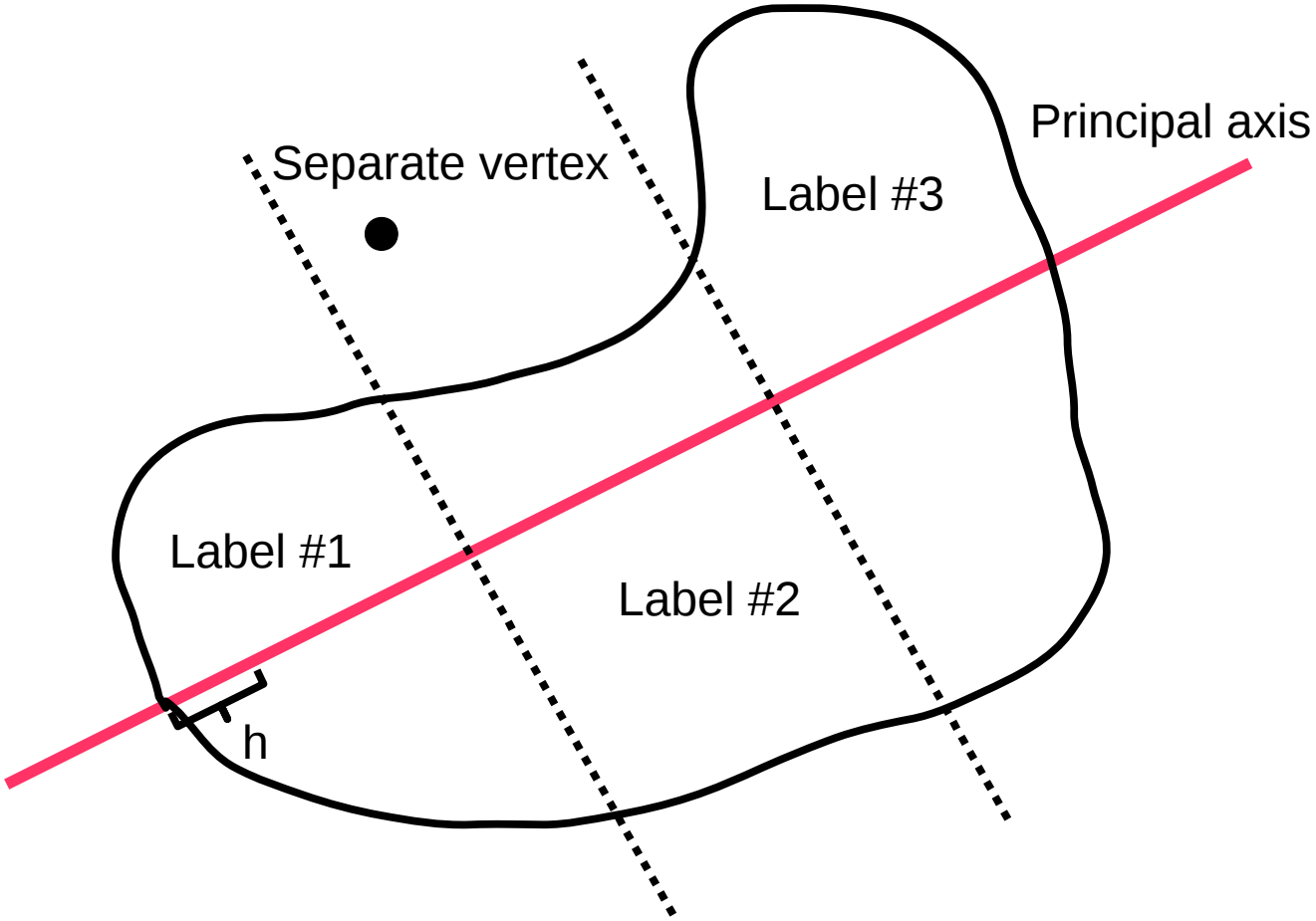
Computing time varies from 1 up to 10 minutes depending on the number of labels and method (PCA or RG).

2.2. Division methods

(i) PCA

In PCA, the vertices inside a label are projected on their principal component. Then, propagation with step size *h* is done along the principal axis, subsequently checking the area occupied by vertices in

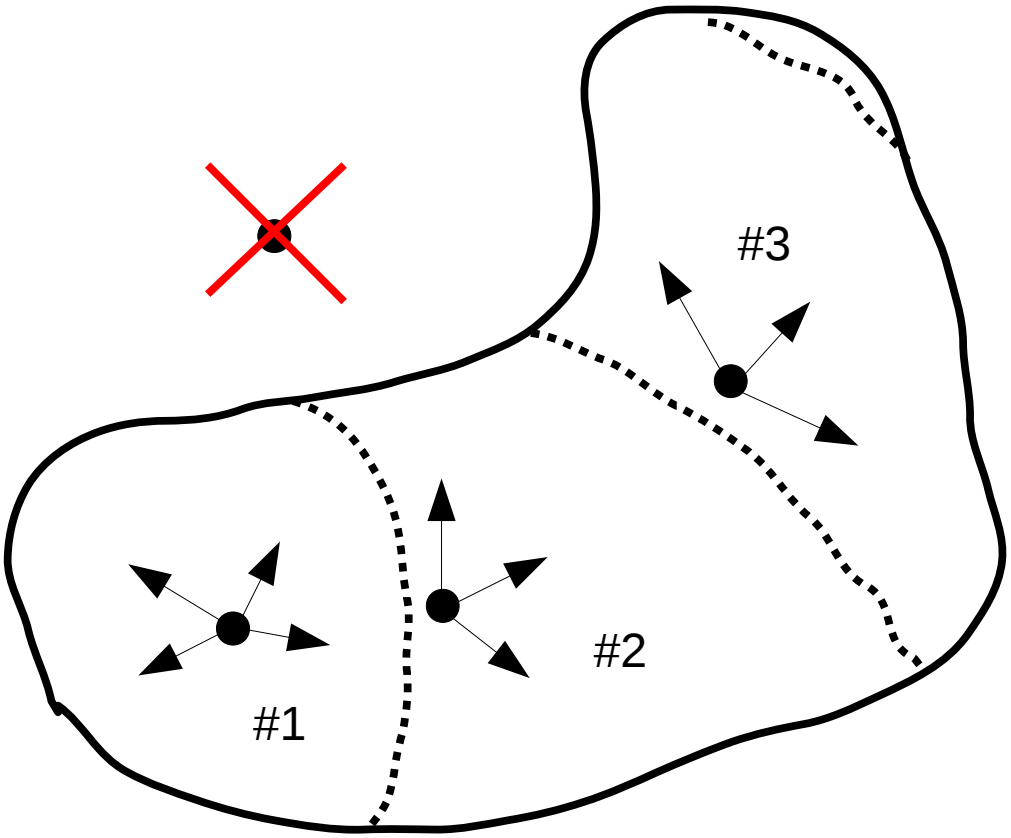
projection length *kh*. A new sublabel is settled every time the target area is reached.



(ii) Region growing (RG)

The RG method is based on the publication by L. Cammoun et al. in [3]. The first sublabel is computed by aggregating neighboring vertices from an initial seed point until the target area is reached. Next, the same procedure is applied for the rest of the label in order to compute the second sublabel. This is repeated until sublabels cover 95 % of the label area.

The first seed point is set as the mean coordinate of the first, imaginary, PCA-divided sublabel. The next seed point is always the nearest point of the rest of the label from the mean vertex of the previously obtained sublabel.



2.3. Smoothing the results with region growing

RG starts simultaneously from the mean coordinates of each sublabel. The algorithm stops when non-overlapping sublabels cover the entire label area despite of the few vertices possibly outside the bounded area (notice that region growing proceeds one edge at a time and cannot jump off the boundary). This procedure results in smooth and continuous labels.

2.4. Label and annotation file naming procedure

A '_sub' extension is added to the name of the sublabels, accompanied with a corresponding index (1, 2, etc.). The new annotation file name ends with '_subdivision' extension.

3. Merge and split features

3.1. Using 'mergefile'

Function **merge_labels** can be used to merge selected labels in an annotation file, creating a new annotation file with '_merged' extension in the file name. In addition, a template splitfile is created from the merged annotation file, making it easier to use a splitfile in split_annotation function (see section 3.2). Mergefile is a special text file. The format is explained in Table 5.

Mergefile is a .txt file with as many rows as the expected number of merged labels. Each row defines a group of labels to merge. Names must be separated with tabs to work correctly.
LABEL_TO_MERGE#1.1 LABEL_TO_MERGE#1.2 ... LABEL_TO_MERGE#1.M
LABEL_TO_MERGE#2.1 ...
...
LABEL_TO_MERGE#N ...

Table 5: Mergefile format

Default path for creating template splitfiles is $SUBJECT/label/splitfiles/

3.2. Using 'splitfile'

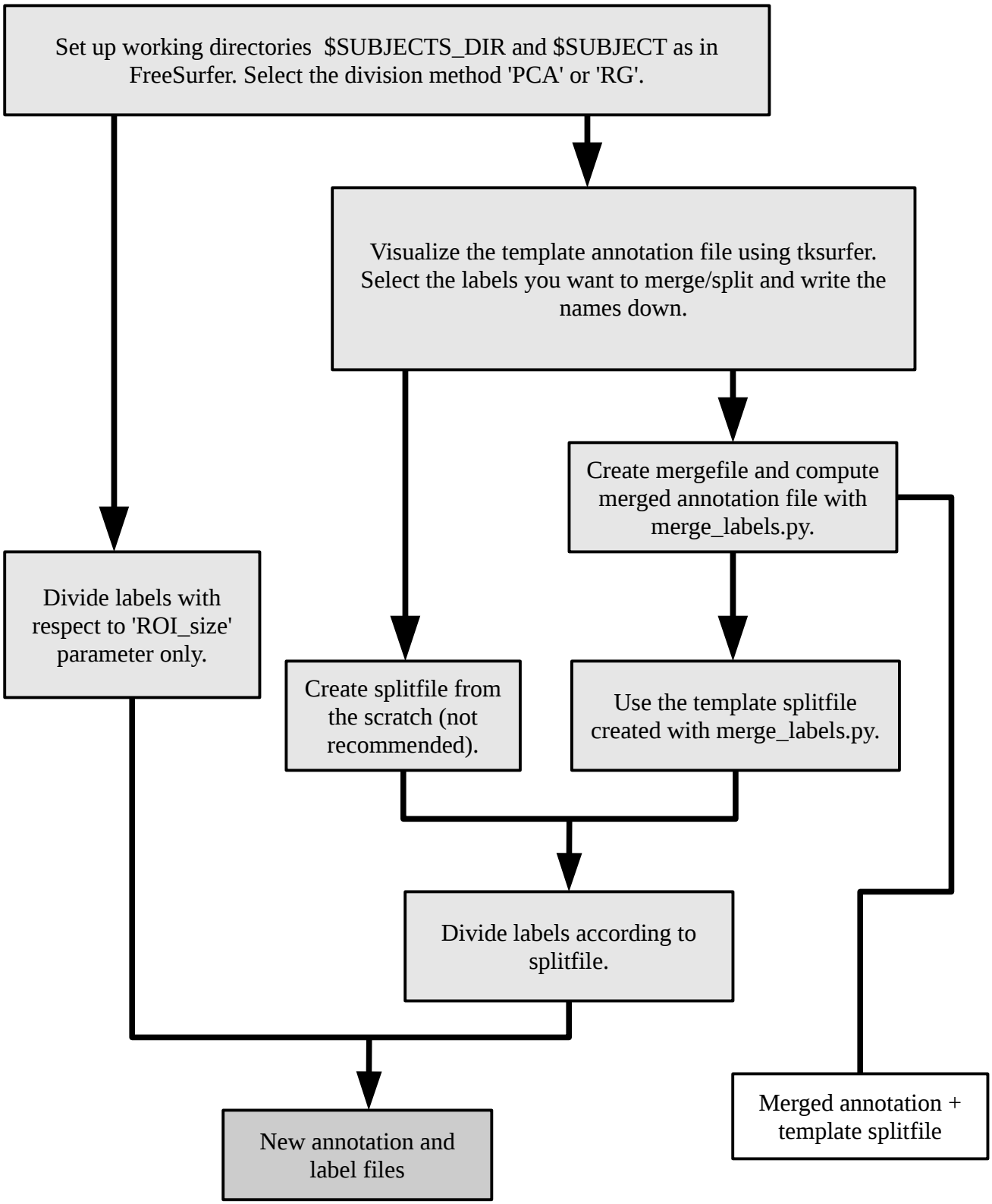
In function **split_annotation**, 'splitfile' parameter can be used to determine the number of subdivision labels in the same manner as in Freesurfer function mris_divide_parcellation.c. If parameter 'splitfile' is 'False', the number of sublabels for each label is computed based on the 'ROI_size' parameter. Else, a specific text file, splitfile, must be used instead. The format is explained in Table 6.

Splitfile is a .txt file with each row defining a label name and a corresponding number of expected sublabels (integer > 0). Names and integers must be separated with tabs to work correctly. For labels not specified in the splitfile, default number of sublabels will be 1.
LABEL_TO_SPLIT#1 NUMBER_OF_LABELS#1
LABEL_TO_SPLIT#2 NUMBER_OF_LABELS#2
...
LABEL_TO_SPLIT#N NUMBER_OF_LABELS#N

Table 6: Splitfile format

Tip: It is easier to modify the template splitfile produced with merge_labels function than start from the scratch. The number of sublabels can be chosen using e.g. tksurfer or other visualization software. An empty mergefile can also be used in order to produce a splitfile with each number of divisions to make corresponding to the 'ROI_size' parameter.

4. Workflow examples



5. References

[1] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, M. Hämäläinen, "MNE software for processing MEG and EEG data", NeuroImage, Volume 86, 1 February 2014, Pages 446-460, ISSN 1053-8119.

[2] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, M. Hämäläinen, "MEG and EEG data analysis with MNE-Python", Frontiers in Neuroscience, Volume 7, 2013, ISSN 1662-453X.

[3] L. Cammoun, X. Gigandet, D. Meskaldji, J. P. Thiran, O. Sporns, K. Q. Do, ... & P. Hagmann, “Mapping the human connectome at multiple scales with diffusion spectrum MRI.” Journal of neuroscience methods, Volume 203.2, 2012. Pages 386-397.