HMIN318M – TP 1 (3 heures)

Ce TP a pour objectif de vous faire programmer un algorithme de segmentation automatique du cerveau à partir d'images IRM en pondération T1. Il peut être effectué seul ou en binôme.

- **Le TP est noté : le compte-rendu doit être envoyé** sur le moodle avant **le mercredi 2 octobre 2019** (minuit)
- Le compte-rendu doit inclure **vos noms** et être composé d'1 à 2 pages de texte **décrivant l'algorithme** ainsi que les commandes lancées **AVEC quelques captures d'écran** pour évaluer le résultat **ET la source** de votre programme ou la liste des fonctions macros utilisées.
- Le tout doit être sous la forme d'un **unique fichier pdf**.
- Tout compte-rendu de TP non envoyé sous la bonne forme dans les délais sera sanctionné par un 0.
- Le TP peut se faire seul ou en binôme
- La participation active pendant le TP pourra aussi être prise en compte.
- **Tout plagiat sera lourdement sanctionné**.

  - **Master 2 Informatique** : obligatoirement C/C++ avec la bibliothèque CImg (http://cimg.sourceforge.net/). Voir Annexe 1.
  - **Autres Masters** : au choix soit en C/C++ avec la bibliothèque CImg, soit en utilisant des macros du logiciel Fiji (http://fiji.sc/). Voir Annexe 2.

1. Visualiser l'image 3D *MR_head.Coronal* (fournie avec le TP) et *brainseg* (depuis le site Web), avec Fiji.

2. Faites apparaître la fenêtre histogramme (Analyze/Histogram ou CTRL+H avec Fiji).
   a. Pouvez-vous expliquer la forme de l'histogramme ?
   b. Le bouton « Auto/Otsu » de Fiji utilise la méthode d'Otsu vue en cours et donne un seuil. A quoi correspond cette valeur ?

3. Essayer de trouver un seuil qui permet de dissocier « au mieux » le cerveau puis la matière blanche. Que remarquez-vous si on seuille à ces valeurs ?

4. En vous inspirant de l'article qui est associé à ce TP, mettez au point un algorithme de segmentation du cerveau. Les paramètres pourront être fixés manuellement.

5. Programmer cet algorithme en C/C++ en utilisant CImg (obligatoire pour les Master 2 Informatique) ou des macros Fiji (au choix pour les autres Masters 2). On trouvera respectivement en Annexe 1 et 2, quelques fonctions Cimg et Fiji qui pourront être utiles.

6. Le tester sur *MR_head.Coronal* et *brainseg*. Discuter le choix des paramètres et évaluer le résultat.

7. Essayer d'automatiser ou de rendre interactif le choix de certains des paramètres pour avoir un programme le plus automatique possible.

Annexe 1

```
CImg<T>& resize ( const int          size_x,
                  const int          size_y = -100,
                  const int          size_z = -100,
                  const int          size_c = -100,
                  const int          interpolation_type = 1,
                  const unsigned int boundary_conditions = 0,
                  const float        centering_x = 0,
                  const float        centering_y = 0,
                  const float        centering_z = 0,
                  const float        centering_c = 0
                )
```

Resize image to new dimensions.

Parameters

```
CImg<T>& threshold ( const T    value,
                     const bool soft_threshold = false,
                     const bool strict_threshold = false
                   )
```

Threshold pixel values.

Parameters:

value	Threshold value
soft_threshold	Tells if soft thresholding must be applied (instead of hard one).
strict_threshold	Tells if threshold value is strict.

Example

```
const CImg<float> img("reference.jpg"), res = img.get_threshold(128);
(img,res.normalize(0,255)).display();
```

```
CImg<T>& erode ( const unsigned int sx,
                 const unsigned int sy,
                 const unsigned int sz = 1
               )
```

Erode image by a rectangular structuring element of specified size (choisir sx=sy=sz=2).

Parameters:

sx	Width of the structuring element.
sy	Height of the structuring element.
sz	Depth of the structuring element.

CImg<T>& dilate (const unsigned int *sx*,
const unsigned int *sy*,
const unsigned int *sz* = 1
)

Dilate image by a rectangular structuring element of specified size (choisir sx=sy=sz=2)..

Parameters:

sx Width of the structuring element.
sy Height of the structuring element.
sz Depth of the structuring element.

CImg<T>& label (const bool *is_high_connectivity* = false,
const Tfloat *tolerance* = 0
)

Label connected components.

Parameters:

is_high_connectivity	Boolean that choose between 4(false)- or 8(true)-connectivity in 2d case, and between 6(false)- or 26(true)-connectivity in 3d case.
tolerance	Tolerance used to determine if two neighboring pixels belong to the same region.

CImg<T>& load_analyze (const char *const *filename*,
float *const *voxel_size* = 0
)

Load image from an ANALYZE7.5/NIFTI file.

Parameters:

	filename	Filename, as a C-string.
[out]	voxel_size	Pointer to the three voxel sizes read from the file.

const CImg<T>& save_analyze (const char *const *filename*,
const float *const *voxel_size* = 0
) const

Save image as an ANALYZE7.5 or NIFTI file.

Parameters:

filename	Filename, as a C-string.
voxel_size	Pointer to 3 consecutive values that tell about the voxel sizes along the X,Y and Z dimensions.

Annexe 2

Resize: Image/Adjust/Size

Threshold: Image/Adjust/Threshold

3D Erode: Plugins/Process/Erode (3D)

3D Dilate: Plugins/Process/Dilate (3D)

Connected regions: Plugins/Process/Find Connected Regions
http://www.longair.net/edinburgh/imagej/find-connected-regions/

Allow diagonal connections?

If you check the box for "Allow diagonal connections?" then two voxels that meet all the other criteria will be considered to be connected even if they only touch at the corners or along one edge. Otherwise, the voxels must share a face. To put that in less sloppy language, the search will be 26-connected rather than 6-connected if this option is selected.

Display an image for each region?

If this is selected then the plugin will generate and display a new image stack each time it finds a connected region. If you are finding lots of regions, you may not have enough memory for this, and you may just be interested in the counts of points in each region. The images produced are masks, rather than being the original values in the stack, since this has been typically more useful in my experience. (The original values can be trivially recovered from the mask, of course.) The exception to this is if you have selected "Regions must have the same value?" in which case original values are kept, so you'll only have two colours in the image anyway.

Display results table?

If this is selected then the plugin will display information about the regions found (sorted by size of region) in the standard ImageJ results dialog.

Regions must have the same value?

In some situations (such as the first example above) you want to restrict the regions to have the same value. In those situations, you should select this box, otherwise (e.g. if you have an image with a smooth range of values) leave it unchecked.

Start from point selection?

By selecting this option, you can specify that the search for the first cluster should start at a particular point by creating a point selection in the image stack and leaving the stack in on the right slice before starting the plugin. (Point selections and other regions of interest in ImageJ are not specific to slices). Otherwise one of two strategies is used for finding the starting point for the next cluster search:

If you specified that all regions must have the same value, then the first voxel that matches the criteria you specified is used in each case.

If you did not specify that all regions must have the same value then the search always starts from point that's left with the maximum value in the image.

Regions for values over:

Points will not be included in any cluster unless the value at the point is over this value. You probably need to tune this a little to get the kind of clusters you want.

Minimum number of points in a region:

In many situations you're only interested in large clusters of points, so you can use this to specify the minimum number of points you want to consider as a cluster.

Stop after this number of regions are found:

You may know that you only want two large clusters in an image, for example, so you can use this option to stop the plugin after it has found those.