ImageJ - TP 1

`ImageJ` est un logiciel libre écrit en JAVA permettant de traiter et analyser des images 2D ou 3D. Il offre de nombreuses fonctionnalités qu'on peut compléter par le biais de macros ou de plugins. Le site web d'`ImageJ` se trouve à l'adresse `http://rsbweb.nih.gov/ij/`.

Pour l'exécuter, il suffit d'appeler le script `run` qui vous est fourni. Ce script contient essentiellement la ligne suivante où `<PLUGINDIR>` est le chemin d'accès aux éventuels plugins et `<IMAGEDIR>` le chemin de l'archive java de `ImageJ` :

```
java -Dplugins.dir=<PLUGINDIR> -cp <IMAGEJDIR>/ij.jar ij.ImageJ
```

Exercice 1.1 *Utiliser ImageJ*

De nombreuses images sont disponibles pour les tests via [`File/Open Samples`]. Vous trouverez toute la documentation nécessaire à partir de l'item *Menu Commands* de la partie *Documentation* des pages web d'`ImageJ` .

1. L'objectif est de visualiser les artefacts de discrétisation. Récupérez des images de pont ou de bâtiment et testez [`Image/Zoom`]. Que voyez-vous ? Y a-t-il une différence entre image acquise et image de synthèse ?
2. Regardez et testez les fonctionnalités présentes dans les menus, notamment dans [`Image`] et [`Process`]. Pourquoi toutes les fonctionnalités ne s'appliquent-elles pas à toutes les images ?

Exercice 1.2 *Ecrire un plugin pour ImageJ*

Vous pouvez consulter la documentation de l'API `ImageJ` sur `http://rsbweb.nih.gov/ij/developer/api/`. Vous trouverez également un tutorial complet à l'adresse :
`http://imagingbook.files.wordpress.com/2013/06/tutorial171.pdf`

Un plugin est une classe Java. Avec votre éditeur préféré créez le fichier Premier_Plugin.java qui contiendra le code suivant :

```
import ij.*;
import ij.process.*;
import ij.plugin.filter.*;

public class Premier_Plugin implements PlugInFilter {
   public void run(ImageProcessor ip){
      ip.flipHorizontal();
   }

   public int setup(String arg, ImagePlus imp){
      return DOES_RGB;
   }
}
```

Exécutez ce code (Menu `Plugins/Compile and Run` de ImageJ, le plugin doit se trouver dans le même répertoire que le fichier `run`). Sur quel type d'images fonctionne-t-il ? Que fait-il ?

Vous pouvez également travailler avec l'environnement de développement Eclipse en suivant les étapes suivantes :

— Configurer Eclipse pour Java
— Créer un projet correspondant à l'ensemble des travaux pratiques de traitement d'image. L'assistant permet de séparer les sources (.java) des fichiers compilés (.class). Vous choisirez `plugins` au lieu de `bin` comme valeur de `default output folder`.
— Ajouter l'archive ImageJ. Après un clic droit sur le nom du projet, suivre les menus `Build Path` puis `Add external archives` et donner le chemin de l'archive java `ImageJ/ij.jar`.

Exercice 1.3 *Accéder aux valeurs des pixels d'une image*

Dans un plugin `ImageJ` de type `PlugInFilter` on accède à l'image ouverte par l'utilisateur via le paramètre de type `ImageProcessor` de la méthode `run`. La classe ImageProcessor possède les méthodes suivantes :

— `getWidth()` qui retourne la largeur de l'image (x $\in [0, getWidth()[$).
— `getHeight()` qui retourne la hauteur de l'image (y $\in [0, getHeight()[$).
— `getPixel(x, y)` qui retourne un entier correspondant à la valeur du pixel de coordonnées (x, y).
— `putPixel(x, y, val)` qui permet de donner une nouvelle valeur au pixel (x, y).

Ecrivez une nouvelle version du plugin de l'exercice 1 sans utiliser la méthode `flipHorizontal` et donc en implémentant l'algorithme correspondant.

Exercice 1.4 *Images couleur : création / accès aux canaux RGB*

Dans cet exercice vous utiliserez la méthode `int[] getPixel(int x, int y, int[] iArray)` qui s'applique à un objet de type `ImageProcessor` correspondant à une image couleur. Cette méthode remplit et retourne un tableau de trois entiers correspondant aux valeurs des canaux R, G, et B du pixel de coordonnées `(x, y)`. Vous utiliserez également la méthode `void putPixel(int x, int y, int[] iArray)`

Complétez le code suivant de façon à créer une image contenant uniquement le canal rouge de l'image initiale.

```
public class Canal_Rouge implements PlugInFilter {

   public int setup(String arg, ImagePlus imp) {
      return DOES_RGB;
   }

   public void run(ImageProcessor ip) {
      int w = ip.getWidth();
      int h = ip.getHeight();
      ImagePlus imR = NewImage.createRGBImage ("Canal rouge", w, h, 1, NewImage.FILL_BLACK);
      ImageProcessor ipR = imR.getProcessor();
      ...
      imR.show();
   }
}
```

Ecrire une autre version de ce plugin en utilisant uniquement les méthodes `int getPixel(int, int)` et `void putPixel(int, int, int)`.