TP 3

TP - Introduction à la programmation sous ImageJ

Nicholas Journet - Traitement d'images - IUT - [1]

3.1 Programmation de plugins

ImageJ est un logiciel dont les fonctions peuvent être étendues par le biais de plugins. Les plugins sont des classes Java placées dans un dossier précis : le dossier `plugins` d'ImageJ (ou un de ses sous dossiers) [2].
Si vous travaillez avec `eclipse`, n'oubliez-pas d'importer le jar en tant que librairie externe : *Project* → *Properties* → *Java Build Path* → *Librairies* → *Add External JARs*.
Nous allons créer un premier script afin d'étudier la structure d'un plugin.

Question 1
Créez le fichier MonScript.java en recopiant le code suivant :

```
1  import ij.*;
2  import ij.process.*;
3  import ij.plugin.filter.PlugInFilter;
4
5  public class MonScript implements PlugInFilter{
6
7      public void run(ImageProcessor ip){
8          IJ.showMessage("Affichage de la Joconde");
9          ip.invert();
10     }
11
12     public int setup(String arg, ImagePlus imp){
13         if (arg.equals("about"))
14             return DONE;
15
16         new ImageConverter(imp).convertToGray8();
17         return DOES_8G + DOES_STACKS + SUPPORTS_MASKING;
18     }
19
20 }
```

Question 2
Juste histoire de se rafraîchir un peu la mémoire : expliquez les lignes de code 1, 5, 7, 8 et 9.

Question 3
Compilez et exécutez ce code (le fichier java doit être placé dans le répertoire **plugins** d'imageJ). Que fait-il ?

Sachant que la classe ImageProcessor possède les méthodes suivantes :

- `getPixels` dont voici un exemple d'utilisation : `byte[] pixels = (byte[]) monProcessor.getPixels();` et qui permet de récupérer les niveaux de gris de l'image dans un tableau mono-dimensionnel.
- `getHeight()` qui retourne la hauteur de l'image ($i \in [0, getHeight()[$).
- `getWidth()` qui retourne la largeur de l'image ($j \in [0, getWidth()[$).

1. Support inspiré du tutoriel ImageJ de l'IJM.
2. Si vous travaillez avec l'archive `ij.jar`, placez vos plugins dans un répertoire `plugins`, que vous aurez préalablement créé dans le même répertoire que `ij.jar`.

Question 4
Quelle est la taille du tableau `pixels[]` ?

Question 5
En vous aidant de ce que nous avons vu en cours, indiquez les lignes de commandes permettant d'accéder aux pixels suivants du tableau `pixels[]`. On précise que le pixel `(i=0,j=0)` est en haut à gauche de l'image.

- $(0, 0)$
- $(0, 10)$
- $(10, 0)$
- $(234, 132)$

Question 6
Déduisez une formule générale permettant d'accéder à n'importe quel pixel d'une image.
`ndg[i][j] = p[?]`

Question 7
Le type `byte` en Java est signé et prend sa valeur entre -128 et 127, alors que nous voudrions un niveau de gris sur 8 bits entre 0 et 255. Si l'on caste un `byte` vers un autre type, il faut s'assurer que le bit de signe est éliminé. Ceci est fait très facilement avec un **et** logique

```
int pix = pixels[i] & 0xff;    // conversion en int
...
pixels[i] = (byte) pix;        // re-conversion en byte
```

Complétez le code suivant afin de permettre la binarisation d'une image.
Binariser signifie que l'intensité d'un pixel est mis à 0 si le niveau de gris de ce pixel est inférieur à un seuil. Ce pixel est mis à 255 si il est supérieur ou égal à ce seuil.

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

public class Binarize implements PlugInFilter{

    public void run(ImageProcessor ip){
        binarize(ip, 127);
    }

    public void binarize(ImageProcessor ip, int threshold){
        byte[] pixels = (byte[]) ip.getPixels(); // Notez le cast en byte ()

        int height =  _____ ;
        int width =  _____ ;

        for (int i=0; i <  _____ ; i++)
          for (int j=0; j <  _____ ; j++){
                int pix = pixels[ _____ ] &  _____ ;

                if(  _____ )
                    pixels[ _____ ] = (byte)  _____ ;
                else
                    pixels[ _____ ] = (byte)  _____ ;
            }
    }

```

```
    public int setup(String arg, ImagePlus imp){
        if (arg.equals("about"))
            return DONE;

        new ImageConverter(imp).convertToGray8();
        return DOES_8G + DOES_STACKS + SUPPORTS_MASKING;
    }

}
```

Question 8

Complétez le code suivant afin qu'il permette de calculer la moyenne des niveaux de gris d'une image ; Testez-le en vous inspirant du code des questions précédentes.

```
// Retourne la moyenne des intensités d'une image en Niveaux de Gris
public double meanImage(ImageProcessor ip){
    byte[] pixels = (byte[]) ip.getPixels();

    int height =  _____ ;
    int width =  _____ ;

      ____  sum =  __ ;

    for (int i=0; i <  _____ ; i++)
        for (int j=0; j <  _____ ; j++)
            _____ & 0xff;

    return  _____ ;
}
```

Question 9

Complétez le code suivant afin qu'il permette de connaître quelle image parmi celles d'un répertoire ressemble le plus à l'image ouverte avec imageJ. Le code permettant d'obtenir les fichiers présents dans un répertoire vous est donné (voir `listFiles`).

```
import java.io.File;

...

public class CompareImages implements PlugInFilter{

    public void run(ImageProcessor ip){
        String path = "  _____ ";
        File[] files =  _____ ;

        if ( files !=  ____ ){
            double gap = Double.MAX_VALUE;

                  _____
                  _____

            for(int i=0; i <  _____ ; i++ ){
                if( ! files[i].isHidden() ){
                    // Création d'une image temporaire
                    String filePath =  _____ .getAbsolutePath();
                    ImagePlus tempImg = new ImagePlus(filePath);

                    new ImageConverter(  _____ ).convertToGray8();

                    ImageProcessor ipTemp = tempImg.  _____ ;
```

```
25
26                      // Calcul du niveau de gris moyen de l'image
27                      double avgTemp =                    ;
28
29                      // Différence par rapport à l'image d'origine
30                      double dif = Math.abs(              );
31
32                      if ( dif <      ){
33
34                      }
35                  }
36              }
37
38              String closestImageName =                          ;
39              IJ.showMessage("L'image la plus proche est " + closestImageName
                    + " avec une distance de " + gap + ".");
40
41          }
42      }
43
44      public File[] listFiles(String directoryPath){
45        File[] files = null;
46        File directoryToScan = new File(directoryPath);
47        files = directoryToScan.listFiles();
48        return files;
49      }
50
51      public double meanImage(ImageProcessor ip){
52        ...
53      }
54
55      public int setup(String arg, ImagePlus imp){
56          ...
57      }
58  }
```

Si on ne souhaite pas utiliser le plugin avec une image de départ, il faut implémenter l'interface `Plugin` à la place de l'interface `PlugInFilter`. Du coup, la fonction `setup` n'est plus à redéfinir.

Question 10

Avec ImageJ, ajoutez du bruit à une image de la base fournie (*Process* $\rightarrow$ *Noise* $\rightarrow$ *Add Noise*). Le plugin de la question précédente `CompareImages` arrive t-il a retrouvé l'image d'origine dans la base, et pourquoi ? Même question, mais cette fois-ci avec du bruit *Salt and Pepper*.

Question 11

S'il vous reste du temps, inspirez vous du cours pour trouver d'autres caractéristiques pertinentes permettant de caractériser le contenu d'une image :

- ➤ Profil horizontal
- ➤ Profil vertical
- ➤ Moments d'ordre 2 et 3

Sont-elles plus tolérantes aux différents bruits ?