PROJET

1

Projet d'analyse d'images : Moteur OCR

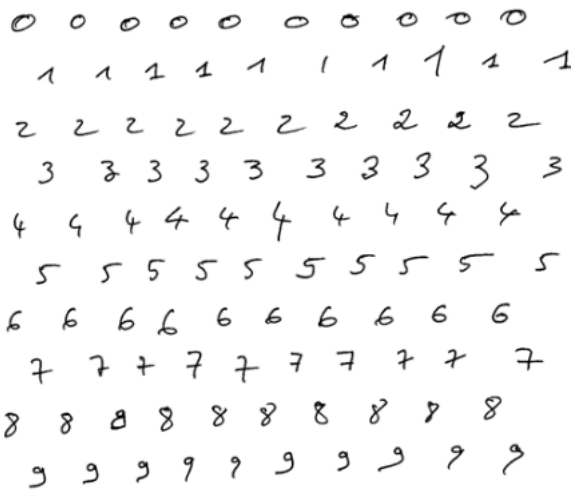
Nicholas Journet - projet d'analyse d'images - IUT -

Objectifs du projet

- L'objectif de ce projet est de réaliser, avec ImageJ, un moteur d'OCR de chiffres manuscrits
- N'hésitez pas à vous référer aux supports de cours pour trouver de bonne idées à mettre en place
- Ce projet est à réaliser pendant les 4 dernières séances d'analyse d'images
- A la fin des 4 séances vous rendrez un rapport d'une dizaine de pages maximum dans lequel vous expliquerez clairement le travail réalisé (objectifs, algorithmes mis en place, tests effectués, résultats obtenus...). Vous pouvez vous inspirer du plan de ce support.
- Vous rendrez également une archive comportant le code source et les images utilisées pour les tests (si vous en avez ajouté d'autres)
- Vous n'êtes pas évalués sur la qualité de votre code JAVA, cependant un minimum de bon sens et de rigueur dans le code est requis. Essayez, entre autre, de vous conformer aux recommandantions de codage données dans ce sujet.

1.1 Présentation des données

Vous avez à votre disposition l'archive `baseProjetOCR.tar.gz` contenant 100 chiffres (dix 0, dix 1,..., dix 9).



Chaque chiffre a été préalablement segmenté. On retrouve donc dans l'archive qui vous est fournie 100 images (une image par caractère).
Le nom de chaque fichier est construit selon le masque suivant : chiffre_numéroImage Vous remarquerez que tous les chiffres d'une même classe ne se ressemblent pas. Vous remarquerez également que les images n'ont pas toutes la même taille.

1.2 Travail préalable

Afin de vous aidez a réaliser votre moteur d'OCR, vous allez commencer par inplémenter quelques méthodes qui vous seront utiles.

Question 1

Tout d'abord, créez une nouvelle classe Image qui permettra de manipuler plus simplement des images. Cette classe fera partie du package `aimage`. Tous les algorithmes travaillant sur le contenu de l'image (binariser, filtrer,...) seront à implémenter dans la classe `OCRImage`

N'oubliez pas de créer les méthodes `get` et `set` de chaque attribut ainsi qu'un constructeur à 3 arguments.

```
1  package aimage;
2  import ij.*;
3  public class OCRImage{
4    private ImagePlus img;//contenu de l'image
5    private char label;//correspond au label de l'image (donne par le nom
         du fichier--> 0,1,... ou 9)
6    private String path;//path du fichier image
7    private char decision;//affectation du label apres classification
8    private ArrayList<Double> vect;//Vecteur de caracteristiques de
         l'image
9  public OCRImage(ImagePlus img, char label, String path)
10 {
11   this.img=img;
12   this.label=label;
13   this.path=path;
14   this.decision='?';
15   vect = new ArrayList<Double>();
16 }
17 public void setImg(ImagePlus img){this.img=img;}
18 public ImagePlus getImg(){return img;}
19 public void setVect(int i,double val){this.vect.set(i,val);}
20 public Double getVect(int i){return vect.get(i);}
21
22 }
```

Question 2

Créez une classe principale pour votre moteur d'OCR. Cette classe doit importer la package `aimage`. Cet import permet ainsi de gérer indépendamment les algorithmes de traitements d'images et l'appel à ces traitements. Cette classe principale possède un attribut de classe permettant de gérer une liste d'`OCRImage` : `ArrayList<OCRImage> listeImg`

On vous donne la méthode `public void createListeImage(String path)` qui permet de remplir votre `ArrayList` d'`OCRIMAGE`. Cette méthode est implémentée dans la classe principale de votre moteur d'OCR.

```
1  public void createListeImage(String path,ArrayList<OCRImage> listeImg)
2  {
3  File[] files=listFiles(path);
4  if (files.length!=0)
5    {
6    for (int i=0;i <files.length;i++){
7      ImagePlus tempImg = new ImagePlus(files[i].getAbsolutePath());
8      new ImageConverter(tempImg).convertToGray8();
9      listeImg.add(new OCRImage(tempImg,
           files[i].getName().substring(0,1).charAt(0),
           files[i].getAbsolutePath()));
10   }
11
12   }
13 }
```

Question 3

Dans le package `aimage` créez la classe `calculMath` permettant d'effectuer des calculs sur des vecteurs de doubles. Toutes les méthodes de cette classe seront `static`.

Dans `calculMath.java`, écrivez la méthode `public static double distEucli( ArrayList <Double> vect1,ArrayList <Double>vect2)` permettant de retourner la distance euclidienne entre deux vecteurs de même taille.

Question 4

Dans `CalculMath.java`, écrivez la méthode `public static int PPV(ArrayList<Double> vect,ArrayList<ArrayList <Double» tabVect,int except)` qui retourne l'indice du vecteur de tabVect correspondant au vecteur le plus similaire à vect (vous devez utiliser `distEucli` pour calculer une similarité entre vecteurs).

Pour ne pas fausser les résultats, l'entier `except` permet de spécifier l'indice d'un vecteur de `tabVect` à ne pas comparer avec `vect`.

Dans votre classe principale, si vous exécutez le code suivant la valeur retournée est 1 car tab2 est le vecteur le plus similaire à tab0.

```
1 ArrayList <Double>tab0 = new ArrayList<Double>(); tab0.add(new
     Double(1.0));tab0.add(new Double(1.0));
2 ArrayList <Double>tab1 = new ArrayList<Double>(); tab1.add(new
     Double(5.0));tab1.add(new Double(-1.0));
3 ArrayList <Double>tab2 = new ArrayList<Double>(); tab2.add(new
     Double(2.0));tab2.add(new Double(1.0));
4 ArrayList <Double>tab3= new ArrayList<Double>(); tab3.add(new
     Double(-1.0));tab3.add(new Double(0.0));
5 ArrayList<ArrayList<Double>> myList = new ArrayList<ArrayList<Double>>();
6 myList.add(tab1);myList.add(tab2);myList.add(tab3);
7 IJ.showMessage("dist= "+calculMath.PPV(tab0,myList));
```

Question 5

Dans votre classe principale, écrivez la méthode `public void LogOCR(String pathOut)` qui écrit dans le fichier `pathOut` la matrice de confusion. Cette méthode est donc à lancer uniquement si la reconnaissance des caractères a été opérée et que l'attribut `decision` a été affecté d'une valeur (entre 0 et 9).

Une matrice de confusion est une mise en forme des résulats elle permet de visualiser dans quelles classes ont été affectées chacune des images testées. Pour construire cette matrice il vous suffit donc de comparer, pour un objet de la liste, si le label (connu grace au nom du fichier) est le même que celui de l'attribut `decision`.

La figure suivante vous montre le résultat obtenu. Pour afficher la date du résultat il faut utiliser `new Date()`.

Dans l'exemple donné :

- 3 chiffres O ont été identifiés comme étant des chiffres 4
- 1 chiffre 2 a été correctement identifié comme étant un chiffre 2
- 3 chiffres 8 ont été identifiés comme étant des chiffres 7
- Aucun chiffre 4 n'a été bien identifié

Sur la diagonale de la matrice on trouve donc les cas de figure où les chiffres ont été correctement classés. Pour connaître le taux de reconnaissance du test, il suffit donc de diviser la trace de la matrice par le nombre total d'images.

```
Test OCR effectués le Mon Nov 17 17:40:06 CET

      0     1     2     3     4     5     6     7     8     9
-------------------------------------------------------------------------------
0|    0     2     1     0     3     2     0     1     1     0
1|    0     1     3     0     1     2     0     1     2     0
2|    1     0     1     2     1     2     1     1     1     0
3|    2     2     2     1     0     0     1     2     0     0
4|    0     1     1     0     0     0     4     4     0     0
5|    0     0     3     1     2     0     1     2     1     0
6|    1     2     0     2     1     1     1     0     2     0
7|    1     1     2     2     0     0     3     1     0     0
8|    1     2     1     1     2     0     0     3     0     0
9|    2     1     0     2     2     1     0     2     0     0
-------------------------------------------------------------------------------
Le taux de reconnaissance est de 5%
```

Vous ne pouvez pas encore tester la méthode qui génère la matrice de confusion car vous n'avez pas encore implémenté les algorithmes d'extraction de caractéristiques. Cependant, en modifiant légèrement le constructeur de la classe `OCRimage` on peut affecter aléatoirement une valeur à l'attribut `decision`. Vous pouvez ainsi tester votre méthode générant un fichier résultat.

```
int val=(int)(Math.random()*10.0);
String i = Integer.toString(val);
char a = i.charAt(0);
this.decision=(char)a;
```

1.3 Premiers tests sur la base d'images

Avant de vous lancer dans la mise en place d'algorithmes de reconnaissances des formes, il est nécéssaire de bien étudier les images qui vont sont proposées afin d'éviter de faire des erreurs de débutants.

1.3.1 Niveau de gris global

Question 6
Reprenez le code de la méthode calculant le niveaux de grix moyen d'une image (tp précédent) et écrivez, dans `OCRImage` la méthode `public double AverageNdg()` qui permet de retourner le niveau de gris moyen d'une image.

Question 7
Dans `OCRImage`, écrivez la méthode `public void setFeatureNdg()` permettant d'initialiser le vecteur caractéristique d'une image à une unique valeur (sa moyenne de niveaux de gris).

Question 8
Dans votre classe principale, écrivez la méthode `public void setFeatureNdgVect()` permettant d'intialiser le vecteur de chaque image contenue dans ArrayList<OCRImage> listeImg

Question 9
Générez la matrice de confusion et analysez les résultats obtenus.

1.3.2 Mise à l'échelle des images

Question 10
Dans votre rapport, à l'endroit où vous présenterez vos données (les images), vous ferez apparaître un récapitulatif des caractéristiques des images composant la base (taille, format d'image, poids, nombre d'images...).
Un graphique peut sembler être une bonne idée pour présenter de manière synthétique des données.

Question 11
En toute logique, après avoir rigoureusement étudié les caractéristiques des images, vous observez qu'elles n'ont pas toutes la même taille. Cette différence peut entrainer des erreurs lors de l'étape de la comparaison des images (dans notre cas, il faut comparer des images de même taille).
En adaptant la méthode `resize` qui vous est fournie, faites en sorte que toutes les images de la base soient analysées à la même taille (20 pixels de largeur sur 20 pixels de hauteur).

```
public void resize(ImagePlus img,int larg,int haut)
{
    ImageProcessor ip2 = img.getProcessor();
    ip2.setInterpolate(true);
    ip2 = ip2.resize(larg, haut);
    img().setProcessor(null, ip2);
}
```

Question 12
Après avoir appliqué la méthode `AverageNdg` sur chaque image, générez la matrice de confusion sur des images ayant toutes la même taille et analysez les résultats obtenus.

1.3.3 Chaine de traitements

Suite aux tests que vous venez de réaliser, proposez une chaine de traitements expliquant comment vous faites pour reconnaître un chiffre. Cette chaîne de traitements est à présenter sous la forme d'un schéma et doit comporter toutes les étapes importantes en spécifiant les précédences entre ces étapes.

1.4 Algorithmes à réaliser

Comme vous l'avez remarqué, une simple moyenne des niveaux de gris calculée sur toute l'image ne semble pas permettre de décrire précisément une forme (taux de reconnaissance très faible).
C'est pourquoi, dans cette section, vous devez implémenter 3 nouveaux algorithmes d'extraction de caractéristiques. Ces trois algorithmes sont décrits dans le cours d'analyse d'images

Question 13
Pour chacun de ces algorithmes, générez la matrice de confusion et analysez les résultats.

1.4.1 Vecteur profil

Question 14
Implémentez la méthode `setFeatureProfilH` qui intialise le vecteur caractéristique de chaque image avec son profil horizontal. La forme est donc décrite par un vecteur dont la dimension est égale à la hauteur de l'image.

Question 15
Implémentez la méthode `setFeatureProfilV` qui intialise le vecteur caractéristique de chaque image avec son profil vertical. La forme est donc décrite par un vecteur dont la dimension est égale à la largeur de l'image.

Question 16
Implémentez la méthode `setFeatureProfilHV` qui intialise le vecteur caractéristique de chaque image avec la concaténation de son profil vertical et de son profil horizontal. La forme est donc décrite par un vecteur dont la dimension est égale à la largeur+hauteur de l'image.

1.4.2 Rapport isopérimétrique

Question 17
Implémentez la méthode `rapportIso` qui intialise le vecteur caractéristique de chaque image avec son rapport isopérimétrique.
Pour estimer la surface d'un chiffre, il faut compter le nombre de pixels qui ne sont pas blancs.
Pour estimer le périmètre d'un chiffre, il faut compter le nombre de pixels noirs qui ont au moins un pixel voisin blanc.
La forme est donc décrite par un vecteur d'un seul réel.

1.4.3 Zoning

Question 18
Implémentez la méthode `Zoning()` permettant de découper l'image en 9 blocs de tailles identiques. Sur chacun de ces blocs, la méthode calcule la moyenne des niveaux de gris et intialise le vecteur de caractéristiques de chaque image.
La forme est donc décrite par un vecteur de 9 réels.

1.5 Synthèse des expérimentations

Question 19
Rédigez un paragraphe de synthèse du travail effectué. Dans cette synthèse vous expliquerez quel algorithme de description de formes vous avez finalement choisi et détaillerez les résultats obtenus.

1.6 TRAVAIL BONUS

1.6.1 Testez votre propre base de chiffres

Les tests que vous avez réalisés sont censés avoir mis en évidence un taux de reconnaissance relativement élevé. Ce taux s'explique par la simplicité de la base de chiffres que je vous ai fournie.
En utilisant la base construite en TP avec le scanner, recommencez les tests avec les descripteurs ayant donnés les meilleurs résultats sur la base initiale.

1.6.2 Faire grimper le taux de reconnaissance

Il est potentiellement possible de faire grimper le taux de reconnaissance moyen en améliorant l'étape de classification. Dans la version actuelle de notre moteur d'OCR, la forme inconnue est classée en fonction de l'image de la base qui lui est la plus proche. Ce choix rend la classification sensible à une forte variabilité intra-classe et une faible variabilité inter-classes.

Une alternative consiste donc à construire des centres de classes (un vecteur moyen pour la classe des 0, un vecteur moyen pour la classe des 1,...). La forme inconnue sera donc comparée à seulement 10 vecteurs au lieu d'être comparée à l'ensemble de la base.

Question 20
Dans la classe `calculMath`, implémentez la méthode `public static ArrayList <Double> avgVector(ArrayList<ArrayList <Double» tabVect)` qui calcule et retourne le vecteur moyen de l'ensemble des vecteurs de `tabVect`

Question 21
Dans la classe principale, implémentez la méthode `public void generateCentreClasse()` qui permet de calculer les 10 centres de classes correspondant aux 10 chiffres que l'on souhaite reconnaitre.

Question 22
Générez la matrice de confusion et analysez les résultats.