

Reconnaissance de visages TP distanciel - Scilab

La reconnaissance de visages est de plus en plus utilisée de nos jours, notamment pour des aspects sécuritaires. Cette reconnaissance peut être effectuée par des attributs décrivant la forme, la couleur et/ou la texture. Dans ce TP, nous proposons de mettre en oeuvre une approche permettant de reconnaître les visages. La base d'images "ORL Database of Faces" mise à disposition par l'université de Cambridge illustre bien le challenge à relever (cf. Figure 1).



FIGURE 1 – Base d'image "ORL Database of Faces".

1 Logiciel utilisé

Le logiciel qui sera utilisé dans le cadre de ce TP en distanciel est Scilab. Il est téléchargeable à l'adresse suivante : <https://www.scilab.org/download/6.1.0>. Une fois le téléchargement effectué, lancez l'exécutable et suivez l'assistant d'installation en sélectionnant une installation complète.

Tout comme Matlab, Scilab est associé à des boîtes à outils appelées toolbox permettant d'accéder à des fonctions spécifiques comme celles utilisées pour le traitement d'images par exemple. Ce TP nécessite ainsi la toolbox "Image Processing and Computer Vision". Pour installer cette toolbox, il suffit d'exécuter la commande `atomsInstall("IPCV")`. Il faut enfin redémarrer Scilab pour que l'installation de la toolbox soit effective.

2 Quelques généralités sur Scilab

La fenêtre Scilab est décomposée en plusieurs onglets :

- un onglet d'édition des commandes,
- un onglet de visualisation de l'espace des variables,
- un onglet de visualisation des fichiers du répertoire de travail,
- un onglet de visualisation de l'historique des commandes,
- et un onglet permettant de visualiser le contenu des variables.

Les fonctions sont éditées dans la fenêtre de commandes et exécutées en appuyant sur la touche Entrée. Le point virgule à la fin d'une fonction permet d'éviter d'afficher les données résultats de la fonction exécutée ou de séparer plusieurs fonctions sur une même ligne de commande. Plusieurs fonctions et commandes peuvent être saisies dans un fichier qui sera enregistré avec l'extension `.sci`. Il est également possible d'y créer des fonctions en utilisant la commande `function`.

Les commandes du logiciel Scilab sont très proches de celles de Matlab. Ainsi, on retrouve avec Scilab les fonctions `imread`, `imshow`, `rgb2gray`, ... D'autres fonctions sont quant à elles nommées de manière légèrement différente, comme par exemple la fonction `graythresh` dans Matlab qui a pour équivalent `imgraythresh` avec Scilab. Scilab étant un logiciel libre, il n'est néanmoins pas aussi fourni que Matlab et certaines fonctions comme `imclearborder` ou `bwareaopen` ne trouvent pas leur équivalent dans Scilab.

3 Base d'images considérée

Dans ce TP, nous proposons d'utiliser une base d'images composée de 50 classes, avec 12 images par classe. Nous allons travailler ici dans un contexte supervisé. Cela nécessite de disposer d'une sous-base d'apprentissage et d'une sous-base de test. Pour cela, nous choisissons une décomposition de type Holdout 1/2 - 1/2. Cela signifie que la moitié des images sera utilisée pour construire la sous-base d'apprentissage, les images restantes étant utilisées afin de tester la pertinence de la caractérisation. Nous proposons de considérer les images impaires comme images d'apprentissage et les images paires pour constituer la base de test.

4 Caractérisation : extraction des attributs de texture

Afin de classer nos images, il est nécessaire de les caractériser grâce à des attributs. Nous proposons dans ce TP de caractériser les visages par des attributs de texture : les motifs locaux binaires (LBP : Local

Binary Pattern).

4.1 Motifs locaux binaires

La fonction *lbp* permet de calculer les motifs locaux binaires d'une image en niveaux de gris. Elle peut être utilisée pour calculer les LBP sous leur forme classique, telle que vu en cours. L'instruction suivante permet d'accéder au LBP classique :

```
Attributs = lbp(Ima_gray);
```

Il est également possible de calculer les motifs locaux binaires avec différents paramètres, comme dans l'exemple suivant :

```
mapping = getmapping(8, 'u2');  
Attributs = lbp(Ima_gray, 1, 8, mapping, 'h');
```

La définition d'une texture doit impliquer un voisinage spatial. La taille de ce voisinage dépend du type de texture ou de la surface occupée par le motif définissant cette dernière. Dans la fonction *lbp*, le voisinage est défini par 2 paramètres : *N*, le nombre de voisins à analyser et *R*, le rayon du cercle sur lequel ces voisins se situent. La figure 2 illustre deux voisinages, avec différentes valeurs de *N* et *R*.

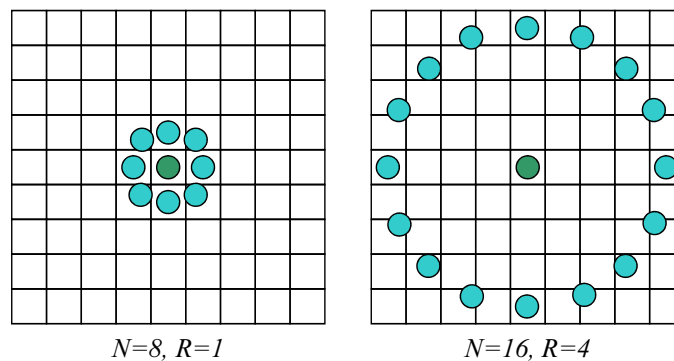


FIGURE 2 – Exemples de voisinages utilisés pour le calcul des LBP.

Le paramètre *mapping* permet quant à lui d'avoir accès aux variantes des motifs locaux binaires grâce à la fonction *getmapping* :

- 'u2' : LBP dits "uniformes", qui sont une version réduite des LBP classiques,
- 'ri' : LBP invariants en rotation,
- 'riu2' : LBP uniformes invariants en rotation.

Le dernier paramètre est le *mode* :

- 'h' ou 'hist' : pour obtenir l'histogramme des LBP,
- 'nh' : pour obtenir l'histogramme normalisé des LBP,
- '' : pour obtenir l'image des LBP.

1) Créer un nouveau fichier *TP.sci*. Au sein de ce fichier, lire et afficher une image de la base et la convertir en niveaux de gris.

2) Extraire de l'image en niveaux de gris précédemment obtenue l'histogramme des LBP sous sa forme classique et observer le résultat obtenu. Noter qu'il est nécessaire d'exécuter au préalable la commande *exec*

lbp.sci pour que la fonction *lbp* puisse être prise en compte.

3) Comparer maintenant les différentes formes de LBP disponibles en faisant varier les paramètres N , R , *mapping* et *mode*.

5 Classification

Maintenant qu'il est possible de calculer un vecteur d'attributs à partir d'une image, nous allons mettre en place la procédure de classification. Cette procédure va nous permettre d'analyser la pertinence de nos attributs en mesurant le taux d'images bien classées.

Le processus de classification est divisé en deux étapes successives :

1. l'apprentissage : l'objectif est de "construire" des classes à partir de l'ensemble d'images d'apprentissage. Pour cela, les visages présents dans les images d'apprentissage sont décrits par un ensemble d'attributs.
2. la classification : durant cette seconde phase, nous utiliserons un classifieur afin d'assigner chaque image test à une classe en fonction de sa similarité. Cette similarité entre images est mesurée en comparant les vecteurs d'attributs.

5.1 Apprentissage

L'apprentissage consiste à ouvrir chacune des images de la base d'apprentissage, calculer et enregistrer dans une variable le vecteur d'attributs de chaque image d'apprentissage ainsi que la classe correspondante.

Pour cela, nous allons utiliser une boucle répétitive telle que présentée sur la page suivante.

4) Compléter le programme proposé en intégrant l'extraction des attributs de texture et réaliser l'apprentissage du processus de classification.

5.2 Décision

Nous allons maintenant mettre en place la procédure permettant de classer une image requête de la base test, l'objectif étant de reconnaître la personne correspondant à l'image analysée.

Le classifieur utilisé pour cela sera l'algorithme du plus proche voisin. Cet algorithme nécessite de mesurer la distance entre le vecteur d'attributs de l'image à classer avec chacun des vecteurs d'attributs des images de la base d'apprentissage. La mesure utilisée sera ici l'intersection d'histogrammes. Plus les images sont similaires, plus l'intersection est importante. L'image à classer sera alors assignée à la classe de l'image d'apprentissage pour laquelle la distance est maximale.

5) Compléter le programme précédent afin d'ouvrir une image de la base test et classer cette image par l'algorithme du plus proche voisin.

Le taux de classification correspond au rapport entre la somme des images test bien classées et le nombre total d'images test.

6) En vous inspirant du programme permettant l'apprentissage, compléter le programme afin de calculer le taux de classification. Un moyen de valider votre programme est de classer les images de la base d'apprentissage et de vérifier que vous obtenez bien un taux de 100%.

```

clear;
xdel(winsid()); // close all;
clc;

exec lbp.sci
exec getmapping.sci

nb_classe = 50; // défini le nombre de classes
nb_image = 12; // défini le nombre d'images par classe
nb_ima_train = 6; // défini le nombre d'images d'apprentissage par classe
nb_bins = 255; // défini la taille de l'histogramme des LBP considéré
Attributs = zeros(nb_ima_train*nb_classe, nb_bins);

// Apprentissage
comp_train = 1;
for i=1:nb_image*nb_classe
    if(modulo(i,2)~=0) // les images impaires constituent les images d'apprentissage
        // Enregistrement du numéro de la classe dans un tableau
        num_classe(comp_train) = floor((i-1)/nb_image) + 1;
        // Détermination du numéro de l'image
        num_image = 1 + modulo(i-1,12);
        // Concaténation des chaînes de caractères
        // pour constituer le chemin d'accès au fichier image
        if (num_image < 10)
            fichier_train = strcat(['Base\ ', sprintf('%d',num_classe(comp_train)),
                                    '-0', sprintf('%d',num_image), '.jpg']);
        else
            fichier_train = strcat(['Base\ ', sprintf('%d',num_classe(comp_train)),
                                    '-', sprintf('%d',num_image), '.jpg']);
        end
        // Affichage du numéro de la classe
        disp([fichier_train 'Classe' sprintf('%d',num_classe(comp_train))]);

        // Ouverture de l'image
        Ima_train = imread(fichier_train);

        // Conversion en niveaux de gris
        Ima_gray_train = rgb2gray(Ima_train);

        // Extraction des attributs de texture

        comp_train = comp_train + 1;
    end
end

```

Les commandes *tic* et *toc* permettent de mesurer le temps nécessaire pour exécuter un code placé entre ces 2 instructions.

7) Mesurer le temps nécessaire à l'extraction des attributs d'apprentissage et de test et le temps mis pour la phase de classification.

8) Comparer les temps et les taux de classification obtenus en fonction des différentes variantes de LBP. La variante qui permet d'obtenir le meilleur compromis entre temps de traitement et taux de classification

sera conservée pour la suite du TP.

9) Analyser et comparer les taux obtenus lorsque l'intersection d'histogrammes est remplacée par la distance de Manhattan.

6 Utilisation de la couleur

Dans cette partie, nous proposons de voir si la couleur permet d'améliorer la classification.

10) Dans un nouveau script, modifier le programme précédent afin d'extraire les composantes R, G, B des images couleur et de calculer l'histogramme des LBP de chacune des 3 images-composante. Concaténer au sein d'un même vecteur les histogrammes issus des 3 images-composante R, G et B et tester votre approche.

Scilab dispose de quelques fonctions de conversion d'espace couleur :

- *rgb2hsv* : RGB \rightarrow HSV,
- *rgb2ntsc* : RGB \rightarrow YIQ,
- *rgb2ycbcr* : RGB \rightarrow YCbCr,
- *rgb2lab* : RGB \rightarrow Lab.

11) Transformer l'image couleur dans ces différents espaces et analyser les résultats de classification obtenus.

7 Analyse centrée sur le visage

La base de données image mise à votre disposition contient des images dont le cadre comprend le visage de la personne, mais aussi ses cheveux et le haut de son buste. La couleur des vêtements portés et la prise en compte des cheveux peuvent influencer favorablement la classification. Cependant, dans la réalité, une personne ne porte pas tous les jours les mêmes vêtements, et peut décider de se coiffer différemment d'un jour à l'autre. Il est donc nécessaire de centrer l'analyse uniquement sur le visage.

Scilab étant un logiciel libre, il n'est pas aussi fourni que Matlab et certaines fonctions proposées par Matlab ne trouvent pas leur équivalent dans Scilab. C'est le cas de l'outil *vision.CascadeObjectDetector* qui permet de détecter les visages. Pour pouvoir utiliser cet outil, Matlab va être temporairement utilisé (une version d'évaluation de 30 jours est disponible gratuitement).

12) A l'aide de l'outil *vision.CascadeObjectDetector* et des fonctions *imcrop* et *imwrite* de Matlab, construire une nouvelle base d'images correspondant aux images de la base initiale, centrées et rognées pour ne considérer que visage de la personne. Noter que les images 01, 02, 09 et 10 de chaque classe ne sont pas considérées ici, étant donné que le visage n'est pas détectable avec l'outil dans ces images.

13) Revenir au logiciel Scilab et tester votre approche de reconnaissance de visages sur la nouvelle base d'images rognées.

Afin d'améliorer la qualité de la reconnaissance, de nombreux auteurs proposent de diviser l'image du visage en imagerie, comme le montre la figure 3.

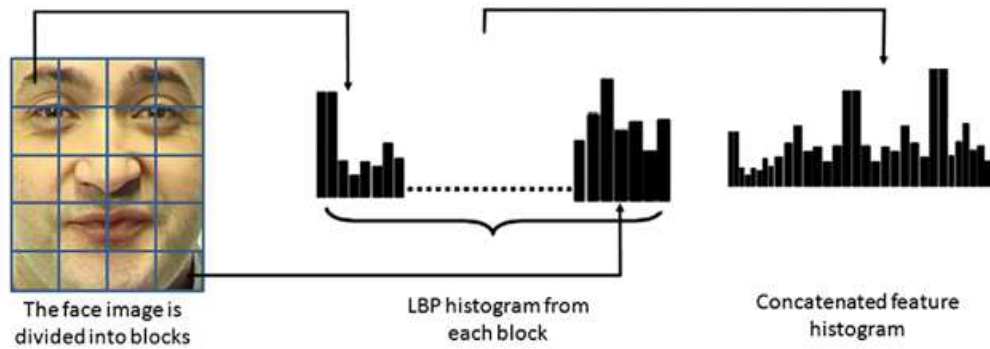


FIGURE 3 – Division en imgettes pour le calcul des LBP.

- 14) Dans un nouveau script, modifier le programme précédent afin de diviser chaque image en 25 imgettes.
- 15) Pour chaque image de la base, extraire l'histogramme des LBP de chaque imgette et concaténer au sein d'un même vecteur les histogrammes obtenus.
- 16) Tester et comparer votre approche en termes de résultats et de temps de traitement.