Reconnaissance de visages

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 place une approche permettant de reconnaître les visages. Nous construirons notre propre base d'images mais utiliserons également des bases d'images de référence afin de valider la méthode proposée. La base d'images "ORL Database of Faces" mise à disposition par l'université de Cambridge illustre bien le challenge à relever. La figure 1 illustre une partie de cette base d'images.

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

1 Création des bases de données image

Dans ce TP, nous proposons de créer notre propre base d'images de visage. Pour cela nous mettons à contribution l'ensemble des étudiants présents dans le groupe de TP.

Pour chaque étudiant, 7 images où le visage aura une expression neutre mais une orientation différente vont être acquises (cf. figure 2).

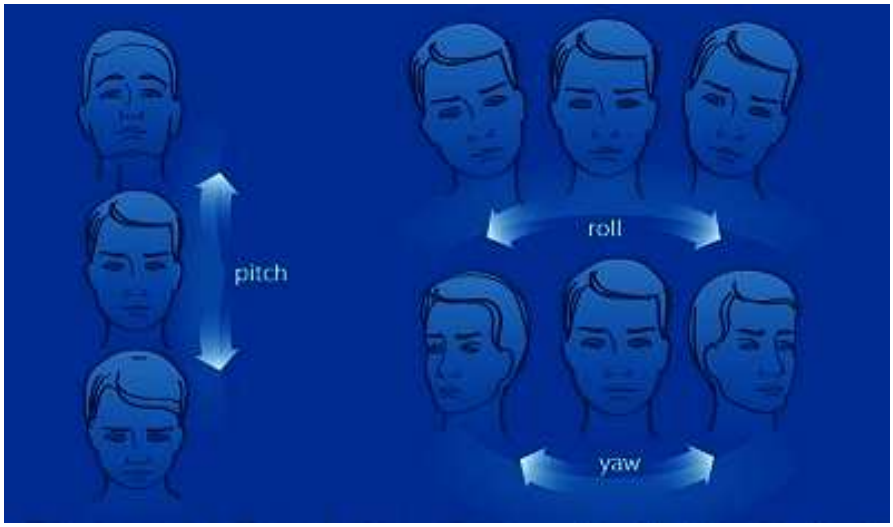


Figure 2 – Les différentes orientations possibles du visage.

La base sera donc composée de C classes, C étant le nombre d'étudiants présents dans le groupe de TP, et de 7 images par classes.

1) A l'aide du banc d'acquisition à votre disposition, acquérir pour chacun des étudiants les 7 images demandées ci-dessus. Veillez à ce que les conditions d'acquisition (fond, éclairage) soient identiques pour tous. Enregistrer les images au format .png en suivant l'incrémentation suivante :

- de 001.png à 007.png pour le premier étudiant pris sous différentes orientations,
- de 008.png à 014.png pour le deuxième étudiant pris sous différentes orientations,
- etc

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.

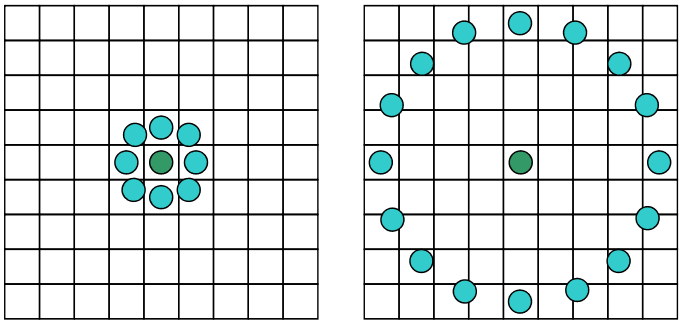
2 Caractérisation : extraction des attributs de texture

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).

2.1 Motifs locaux binaires

La fonction *lbp* permet de calculer les motifs locaux binaires d'une image monochrome avec différents paramètres :

- 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 3 illustre deux voisinages, avec différentes valeurs de N et R.



N=8, R=1 *N=16, R=4*

Figure 3 – 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* :
  — 0 : LBP classique,
  — 'u2' : LBP dits "uniformes", qui sont une version réduite des LBP classiques,
  — 'ri' : LBP invariants en rotation,
  — 'riu2' : LBPuniformes 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,
  — l'image des LBP est retournée lorsque le paramètre n'est pas renseigné.

2) Écrire un script Matlab permettant d'ouvrir et afficher une image, de la convertir en image en niveaux de gris et de calculer l'histogramme des LBP correspondant.

3 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 la méthode du plus proche voisin qui consiste à assigner chaque image requête à la classe de l'image d'apprentissage la plus similaire. Cette similarité entre images est mesurée en comparant les vecteurs d'attributs.

3.1 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.

```
clear all;
close all;
clc;

nb_classe = C; % défini le nombres de classes
nb_image = 7; % défini le nombre d'images par classe
chemin = ('E:\...\');
nb_ima = nb_classe*nb_image;
nb_ima_train = (floor(nb_image/2)+1);
% Pour chaque classe, 4 images parmis les 7 sont utilisées pour l'apprentissage
Attributs = zeros(nb_classe*nb_ima_train, 256);

%% Aprentissage
for i_classe = 1:nb_classe
    for i_train = 1:nb_ima_train
        % Enregistrement du numéro de la classe dans un vecteur
        num_image = i_train + (i_classe - 1) * nb_image;
        num_classe(num_image) = i_classe;

        % Concaténation des chaines de caractères
        % pour constituer le chemin d'accès au fichier image
        if(num_image/10 < 1)
            fichier_train = [chemin '00' int2str(num_image) '.png'];
        else
            if (num_image/100 < 1)
                fichier_train = [chemin '0' int2str(num_image) '.png'];
            else
                fichier_train = [chemin int2str(num_image) '.png'];
            end
        end

        % Affichage du numéro de la classe
        disp([fichier_train ' Classe ' int2str(num_classe(num_image))]);

        % Ouverture de l'image
        Ima_train = imread(fichier_train);
    end
end
```

3) Compléter le programme ci-dessus en intégrant l'extraction des attributs de texture et réaliser l'apprentissage du processus de classification.

3.2 Décision

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

Le classifieur utilisé pour cela sera le classifieur 1-ppv (algorithme du plus proche voisin). Cet algorithme nécessite de mesurer la distance entre le vecteur d'attributs de l'image requête 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 d'apprentissage pour laquelle la distance est maximale est alors l'image la plus ressemblante à l'image requête et sa classe indique la classe de l'image requête.

4) Compléter le programme précédent afin d'ouvrir une image de la base test et classer cette image par un algorithme 1-ppv.

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

5) 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%.

4 Utilisation de la couleur

Nous proposons de voir si la couleur permet d'améliorer la classification.

6) 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.

Matlab dispose de quelques fonctions de conversion d'espace couleur :
- *rgb2hsv* : RGB → HSV,
- *rgb2ntsc* : RGB → YIQ,
- *rgb2ycbcr* : RGB → YCbCr,
- *rgb2gray* : RGB → intensité,
- *applycform* : RGB → XYZ, Yxy, Luv, Lu'v', Lab, Lch, sRGB.

7) Transformer l'image couleur dans différents espaces couleur et observer les histogrammes des LBP dans chacun de ces espaces. Analyser les résultats de classification obtenus dans ces différents espaces.

8) Écrire deux fonctions permettant les transformations suivantes et tester votre approche avec ces nouveaux espaces :
- *rgb2i1i2i3* : RGB → I1I2I3,
- *rgb2rngnbn* : RGB → RGB normalisé.

5 Autres essais

9) Poursuivre les essais expérimentaux en utilisant d'autres attributs de texture, d'autres classifieurs, d'autres mesures de similarité, ... Relever à chaque fois les taux de classification, les temps de traitement, la matrice de confusion, ...