Université Clermont Auvergne
Année 2016-2017 C. Samir

Projet modélisation mathématique : système de reconnaissance faciale

Les feuilles de TP et d'autres documents et informations sont disponibles sur la page du cours sur l'ENT

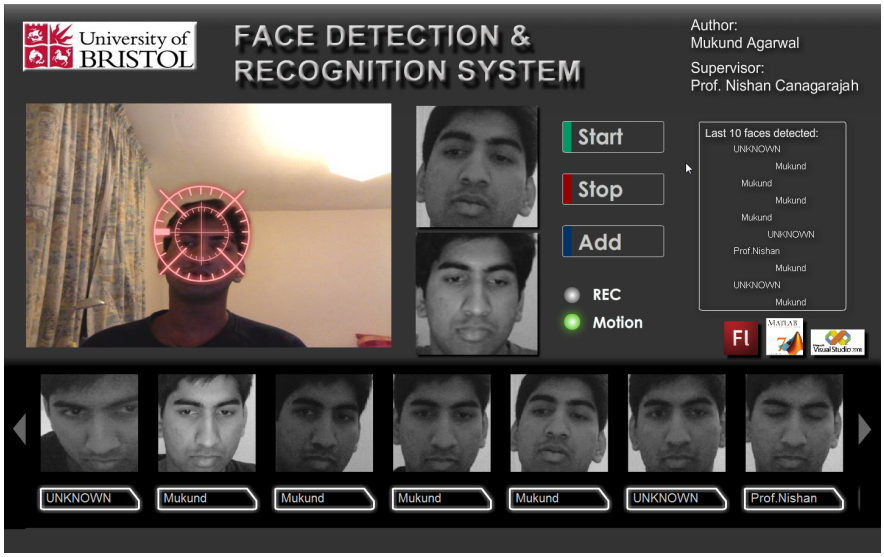

FIGURE 1 – Système de reconnaissance du visage 2D

Exercice 1 (**Images et vecteurs**).
Sur l'image `lena.pgm`.

1. Ā partir d'une matrice I de taille $N \times N$ représentant une image, construire un vecteur-colonne V (de taille $N^2 \times 1$) en transformant la ligne i de I en un morceau de colonne allant de la ligne $(i-1)N+1$ à la ligne iN dans V.
2. Effectuer l'opération inverse : transformer le vecteur-colonne V en matrice carrée J, et vérifier en affichant J.
3. Écrire les deux fonctions `function V=I2Vector(I)` et `function I=Vector2I(V)`, qui seront utiles pour le reste du projet.

Exercice 2 (**Fixer la taille et le format des images**).
Télécharger et décompresser le dossier `Visages.zip` sur l'ENT.

1. Afficher chaque image (il y en a 50) ainsi que sa taille : on constate que certaines des images sont en niveaux de gris tandis que d'autres sont en couleur (RGB).
2. Écrire une fonction **J=RGB2Gray(I)** permettant de transformer une image RGB en image en niveaux de gris à l'aide de la formule : $B(i,j) = round(0.3 * A(i,j,1) + 0.59 * A(i,j,2) + 0.11 * A(i,j,3))$

Exercice 3 (**Visage moyen**).

1. Calculer et afficher le visage moyen (la moyenne de tous les visages).
2. Calculer et afficher la différence (en valeur absolue) entre chaque visage et le visage moyen.
3. Écrire une fonction distance qui permet de comparer deux visages représentés par deux vecteurs.
4. Comparer vos 6 photos avec les images de la base de visages et afficher les 5 images les plus proches.
5. Mettre 3 de vos images dans la base et comparer les 3 qui restent avec la base. Commenter ?

Exercice 4 (**Préparation de la matrice des visages de la base de données**).
Télécharger le dossier `Visages.zip` sur l'ENT et le décompresser.

1. Pour chaque visage, charger et afficher l'image, puis, si nécessaire, la transformer en image en niveaux de gris.
2. Ecrire l'image en niveaux de gris dans un nouveau fichier appelé (par exemple, pour la 25ième image) `ztest25.gif` en utilisant la commande `imwrite(J,"ztest25.gif");`. **Pour le reste du TP, on travaillera sur ces nouveaux fichiers.**
3. Créer une matrice M pleine de zéros, de taille $128^2 = 16384$ par 49.
4. Remplir la matrice M avec les vecteurs-colonnes obtenus à partir des 49 premiers visages en niveaux de gris. (On garde le dernier visage à part pour faire des tests.)

Exercice 5 (**Distance entre deux visages**).

1. Transformer la 50ème image en niveaux de gris en un vecteur VS ("S" pour "sujet"...).
2. Pour chaque colonne M_i de la matrice M, calculer le vecteur différence $D_i = M_i - VS$ puis sa norme d_i (Rappel : c'est un réel positif ou nul...) à l'aide de la fonction `norm`.
3. Déterminer le visage le plus proche de celui du sujet d'après cette approche. Afficher les deux images.
4. Recommencer l'opération en prenant comme sujet une des images de la base de données. À quel résultat doit-on s'attendre et pourquoi ?

Exercice 6 (**Visages "propres"**).

1. Appliquer la commande `[u,s,v]=svd(M,'econ');` (qui signifie "singular values decomposition" = "décomposition en valeurs singulières"). Le résultat est composé de trois matrices, u, s et v qui ont (entre autres) la propriété que $M = us^tv$ et dont seule la première, u, nous intéresse ici. Cette matrice u contient (en colonnes) les vecteurs d'une base particulière du sous-espace vectoriel de $\mathbb{R}^{16384}$ de dimension 49 engendré par les vecteurs de la base de données.
2. Les vecteurs-colonne de u ont la particularité d'être ordonnés dans un ordre d'importance décroissante pour les caractéristiques des visages de la base de données. Extraire les 5 premières colonnes de u dans des vecteur que vous appelerez $VP1, \ldots, VP5$ (pour "visage propre"...).
3. Prendre pour V un des vecteurs-colonne de M. Calculer la projection orthogonale $P1$ de V sur $VP1$, en utilisant la formule $P1 = (V' * VP1) * VP1$. Transformer $P1$ en une image $I1$ et visualiser le résultat.
4. De la même façon, calculer $P2, \ldots, P5$ et visualiser les images obtenues à partir de $P1 + P2$, $P1 + P2 + P3$, $P1 + P2 + P3 + P4$ et $P1 + P2 + P3 + P4 + P5$. Vous devriez observer la reconstruction progressive du visage correspondant au vecteur-colonne V.

Exercice 7 (**Reconnaissance faciale par la méthode des visages propres**).
La méthode de reconnaissance faciale précédente a le défaut de donner le même poids relatif à chacun des pixels des images. Grâce à l'utilisation des visages propres, la méthode présentée dans cet exercice permet de donner plus de poids aux pixels sémantiquement les plus importants (contour du visage, yeux, etc.).

1. Calculer la projection orthogonale P_i de chaque vecteur-colonne M_i de la matrice M sur $VP1$, en utilisant la formule $P_i = (M_i' * VP) * VP$. Ranger l'ensemble dans une matrice P.
2. De même, calculer la projection PS du vecteur VS sur $VP1$.
3. Reprenez les questions 2. à 4. de l'exercice 5 en utilisant la matrice P à la place de la matrice M, et le vecteur PS à la place du vecteur VS.
4. On pourrait aussi appliquer cette méthode en prenant pour P_i la somme des projections orthogonales de M_i sur les k visages propres successifs $VP1, VP2, \ldots VPk$.

Travail à rendre avant le 13/01/17 (au plus tard 23h59)

Comme vous l'avez certainement remarqué, les exercices précédents ont pour objectif de vous donner les outils nécessaires afin de réaliser un système de reconnaissances de visages 2D. En se basant sur ces outils, vous devez rendre les éléments suivants via l'ent :

- Un dossier compressé, avec les noms du binôme, qui contient les fonctions utilisées ainsi que le programme principal (main) de votre projet. A noter qu'il sera tenu compte du temps d'exécution, la documentation des fonctions, l'efficacité des algorithmes, les commentaires au sein des programmes, etc.
- Le programme principal doit fonctionner sur les machines des salles de tp. Il doit permettre de lire une image test, la comparer avec les images de la BD et de trouver un nombre donné des images les plus proches. Pour ce faire, Il doit permettre à l'utilisateur de choisir entre les distances (vecteurs des images, vecteurs propres ou histogrammes (partie bonus)) pour calculer les distances.
- Une interface graphique qui ressemble à la figure 1 : par exemple un bouton pour choisir une image requête, une zone pour afficher les images testées, et des zones pour afficher les images les plus proches à la fin du test.
- Un rapport synthétique et illustré avec des figures pour expliquer le fonctionnement du code et pour commenter les résultats obtenus.

Vous aurez l'occasion de tester votre code et de défendre votre projet lors de la soutenance. Pendant la présentation (transparents), vous aurez 5 mins par binôme pour expliquer l'essentiel de votre projet, 5 mins pour tester le code, et 5 mins pour répondre individuellement aux questions.

1 Partie bonus

- Une autre méthode pour comparer les images consiste à représenter chaque image par son histogramme et d'utiliser une distance (par exemple *chi-squared distance*) entre les histogrammes pour trouver l'image la plus proche dans la base de données.
- Une méthode qui reconnait un visage à partir d'une vidéo (enregistrés ou capturée par webcam).