MASTER EN SCIENCES INFORMATIQUES

Reconnaissance de visages par décomposition en visages propres

De Meyere, Didier

Award date:
2000

Reconnaissance de visages par décomposition en visages propres

Didier DE MEYERE

Promoteur : Madame M. NOIRHOMME

Travail de fin d'études présenté
par Didier De Meyère

en vue de l'obtention du diplôme de
Licencié en Informatique

Année académique 1999 – 2000

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

CHAPITRE 1

CHAPITRE 1 : Introduction

1.1 Introduction

La capacité de l'être humain à reconnaître des visages est chose tellement courante qu'on oublie qu'elle est tout à fait remarquable. En effet, l'être humain est capable de différencier une multitude de visages, de les stocker en mémoire et de reconnaître d'un seul coup des visages qui lui sont familiers même après de longs mois de séparation. Au vu du visage, il est capable également de classer les personnes suivant leur sexe, leur âge ou encore leur race. Il peut aussi décoder les expressions du visage, révélateurs des humeurs , des intentions, etc et qui jouent un rôle prépondérant dans les relations sociales. Les visages sont pourtant des stimuli visuels complexes, qui ne peuvent pas simplement être décrits en utilisant des formes ou des caractéristiques comme nous le verrons plus loin.

Depuis longtemps, les scientifiques tentent de comprendre les processus du système de vision de l'être humain dans le but entre autres d'élaborer des techniques qui permettront la création de systèmes automatiques de reconnaissance.

Ces 20 dernières années, des psychophysiciens, des neuroscientistes et des ingénieurs ont mené une recherche intensive sur différents aspects de la reconnaissance de visages par les êtres humains et par les machines [5]. Les principaux problèmes qui ont été étudiés sont:

- L'unicité des visages.
- La reconnaissance est-elle fait de manière holistique ou par l'analyse de caractéristiques locales du visage.
- L'analyse et l'utilisation des expressions faciales pour la reconnaissance.
- Comment les enfants perçoivent-ils les visages?
- L'organisation de la mémoire des visages.
- La difficulté à reconnaître parfaitement un visage à l'envers.
- Le rôle de l'hémisphère droit dans la perception des visages.

Les travaux menés ne fournissent pas directement une méthodologie pour la reconnaissance mais offrent néanmoins quelques pistes intéressantes dont certaines sont enumérées ci-dessous:

- La reconnaissance des visages appelle une procédure spécifique. En effet, les visages sont plus facilement mémorisés que d'autres objets. Des patients atteints de propagnose reconnaissent des gens par leur voix, la couleur de leurs cheveux ou leurs habits mais ne peuvent distinguer deux visages différents. Cependant les mêmes patients reconnaissent sans peine d'autres objets.
- Certaines zones ou caractéristiques du visage sont plus significatives que d'autres: Les yeux, la bouche, le contour du visage par exemple jouent un rôle prépondérant dans la reconnaissance faciale.
- Le contenu en informations d'une image varie avec la fréquence spatiale de ces informations.
- Les capacités de reconnaissance augmentent en fonction de la qualité de l'image visualisée, mais il existe un plafond dans les performances une fois qu'une certaine résolution a été dépassée. Il ne sert donc à rien d'utiliser des images de qualité excessive.

1.2 Motivation et Présentation du travail

Le travail proposé a pour but de choisir et d'étudier une méthode de reconnaissance de visages utilisant des images de visages en vue frontale et d'implémenter ensuite cette méthode. La méthode choisie est la méthode dite des visages propres, basée sur la technique de l'analyse en composantes principales.

Le présent travail est subdivisé en quatre chapitres. Le premier chapitre est consacré à la définition d'un système automatique de reconnaissance de visages et à l'analyse de certaines techniques qui ont été développées, sans toutefois être exhaustif. Le second chapitre aborde les différentes approches de l'analyse en composantes principales. L'utilisation de cette technique dans la méthode des visages propres est

étudiée au cours du chapitre 3. Le chapitre 4 traite de l'expérimentation de la méthode des visages propres. Il reprend notamment le matériel et les outils de travail ainsi que les diverses étapes du programme réalisé et les principaux résultats obtenus. Le travail se termine par quelques conclusions et par les améliorations à apporter.

1.3 Applications des systèmes de reconnaissance de visages

Avant d'entamer l'étude proprement dite des techniques de reconnaissance, il est intéressant de se demander où sont utilisés ces systèmes de reconnaissance et à quelles fins.

Les applications des systèmes de reconnaissance de visage sont assez nombreuses et variées. On peut ainsi trouver des systèmes automatiques de reconnaissance pour la vérification de photos d'identités se trouvant sur les cartes d'identités, sur les passeports, sur les cartes de crédit, sur les permis de conduire, etc. Une entreprise, un magasin, une banque peuvent mettre en place un système automatique de reconnaissance de visages à des fins de sécurité pour permettre ou refuser l'accès de certains bâtiments, de certains locaux à leurs employés ou aux clients. La police peut également utiliser un système de reconnaissance pour la recherche de criminels. Suivant le domaine d'utilisation, on distingue généralement les applications commerciales des applications visant à faire respecter la loi (dans un cadre policier ou judiciaire par exemple). Une autre classification consiste à différencier les applications utilisant la mise en correspondance statique des applications utilisant la mise en correspondance dynamique (ou en 'temps réel') par la surveillance vidéo de foules dans un aéroport ou autre lieu public par exemple.

Les systèmes de reconnaissance de visages utilisés seuls sont efficaces lorsque les données d'entrée, les images, sont de bonne qualité et bien contrôlées (taille, orientation des visages, lumière, ...). Dans le cas où ces conditions ne seraient pas toujours vérifiées, d'autres techniques comme la reconnaissance vocale par exemple sont également utilisées et une fusion est opérée entre les données des différentes techniques afin d'augmenter la probabilité de reconnaissance d'individus. Il existe d'autres méthodes d'identification telles que les empreintes digitales, la séquence ADN

ou l'inspection de l'iris. Par rapport à ces techniques qui ont fait leurs preuves, la reconnaissance faciale possède les avantages suivants :

- Il s'agit d'un processus passif : La reconnaissance peut s'effectuer sans la participation active du sujet.
- Il s'agit souvent de la seule biométrie disponible : De nombreuses situations se présentent où les photos de visages sont les seules informations disponibles.
- Meilleure acceptabilité sociale : nous sommes habitués à être photographiés, que ce soit pour un permis de conduire ou pour se faire délivrer un passeport.
- Aucun équipement spécial ou excessivement coûteux n'est requis.

1.4 Composantes d'un système automatique de reconnaissance de visages

1.4.1 Système de vision humain

Avant de mettre au point un système automatique de reconnaissance de visages et d'analyse, il faut décider quelles doivent être les fonctionnalités d'un tel système. La meilleure référence que l'on puisse avoir est le système de vision humain.

a) Représentation

La première étape dans la reconnaissance de visages est l'encodage structurel du stimulus visuel.

b) Détection

Les humains détectent la forme d'un visage dans une image par inspection de la scène formant l'image. On peut donc envisager une unité spéciale destinée à la détection ou effectuer cette tâche en parallèle avec d'autres.

Un visage est perçu comme un tout et non comme une collection de caractéristiques. Ainsi si une partie du visage est cachée, nous percevons quand même un visage entier comme si notre système percepteur remplissait la partie manquante. Nous détectons également sans trop d'efforts dans une vaste gamme de conditions, par

exemple dans des conditions de mauvais éclairage, à grande distance, .. Nous détectons mieux certains visages en louchant ou à distance. Ces processus ont pour effet de brouiller, de rendre floue l'image, ce qui revient en fait à filtrer les hautes fréquences spatiales.

c) Identification

Lorsqu'une forme a été perçue comme un visage, l'étape suivante consiste à l'identifier. L'identification est le processus d'association d'un nom à un visage. Une identification correcte signifie que différentes images d'un même visage sont identifiées comme correspondant à un seul visage. Malgré notre aptitude à identifier des visages pratiquement sans effort, il est pourtant difficile de décrire un visage humain. La méthode la plus courante est d'énumérer les différents traits du visage comme par exemple des cheveux noirs, des yeux bruns, un nez plat. Ces caractéristiques jouent un rôle très important dans le processus d'identification. Les caractéristiques le plus couramment utilisées sont: les yeux, le nez, les lèvres, les oreilles et les cheveux. Les relations spatiales entre ces caractéristiques sont également très importantes. On note que les caractéristiques du visage qui ressortent, décroissent en partant du sommet vers la base, ce qui implique que le rôle des caractéristiques telles que les cheveux ou les yeux est plus important dans le processus d'identification que celui des caractéristiques telles que le menton ou les lèvres. Cependant, des visages avec des caractéristiques inhabituelles telles qu'un menton proéminent sont facilement identifiés.

1.4.2 Composantes d'un système de reconnaissance de visages

Les deux composantes principales d'une unité de reconnaissance sont:

- La détection: Elle a pour but de localiser un visage présent dans une image. Dans certains cas, les conditions (lumière, arrière plan, position et orientation du visage) dans lesquelles l'image, la photo est prise, sont contrôlées (prise de photos de criminels par exemple). La localisation du visage peut alors être facilement déterminée. Dans la plupart des cas, par contre, la localisation d'un visage n'est pas connue à priori. La première

étape consiste donc à déterminer si la scène de l'image comprend des visages et si un visage est présent à le localiser. Plusieurs facteurs peuvent rendre ce problème plus complexe notamment la présence de cheveux, de maquillage,... qui peuvent ainsi masquer certains traits du visage. Un autre problème est celui de la variation d'échelle et de l'orientation du visage dans l'image. Le visage peut ainsi apparaître sous différentes tailles et sous différentes orientations. Enfin la présence de bruit accroît également la difficulté de détection.

- L'identification: Elle est la deuxième opération et la plus importante d'un système de reconnaissance de visage. Elle vise à mettre en correspondance le visage détecté dans l'image avec un visage 'connu' enregistré dans une base de données. Il y a deux aspects liés à l'identification:
  - Un nom doit être associé au visage.
  - Plusieurs instances d'un même visage, de tailles ou d'orientations différentes, doivent être identifiées comme appartenant à une seule et même personne.

Parfois certaines unités de reconnaissance de visages possèdent d'autres composantes telles que:

- L'analyse des expressions faciales: Elle a pour objectif de déterminer à partir de l'information contenue dans le visage si l'expression de celui-ci est une expression de bonheur, de regret, de surprise, ... Le problème consiste à modéliser les émotions humaines et à les corréler avec les traits du visage et leurs variations.
- La classification basée sur les caractéristiques physiques: La classification peut être basée sur de nombreux aspects. Ainsi, elle peut être basée sur le sexe: homme ou femme, sur l'âge: enfant, jeune, âgée, ... sur la race: noir, oriental, ..., etc.

1.5 Techniques de reconnaissance automatique de visages

Les deux classes traditionnelles de techniques pour la reconnaissance d'images digitales de vue frontales de visages sont:

- les techniques basées sur le calcul d'un ensemble de caractéristiques géométriques extraites d'une image de visages.
- les techniques basées sur la mise en correspondance avec des gabarits.

1.5.1 Représentation

Deux types de représentations sont généralement utilisés dans la reconnaissance de visage et la recherche d'identification:

- l'image d'intensité en 2D.
- le vecteur de caractéristiques.

L'approche la plus simple pour représenter un visage est d'utiliser un tableau 2D de valeurs d'intensité. Ce n'est pas la représentation la plus compacte mais elle s'avère très utile lorsque la robustesse est prioritaire. Les systèmes qui utilisent d'autres représentations gardent souvent des tableaux de valeurs d'intensité bien qu'ils ne soient pas utilisés explicitement. Dans de très larges systèmes avec des milliers de visages par exemple, une telle représentation n'est pas appropriée. Cependant avec une compression adéquate, par exemple par la réduction de la résolution spatiale ou en niveaux de gris, cette approche peut être pratique. Certains auteurs dans [8] affirment qu'avoir des images 32 x 32 avec 4 bits par pixel est suffisant aussi bien pour la détection que pour l'identification. Ainsi une image peut être représentée par seulement 512 bytes et le stockage de milliers d'images requiert alors quelques mégabytes.

La représentation la plus largement utilisée est le vecteur de caractéristiques. Le tableau ci-après reprend certaines caractéristiques utilisées:

Catégorie	Caractéristiques
Yeux	Gabarit, forme, couleur, surface, centre, intensité autour de la pupille, distance entre les centres des yeux, distances entre les extrémités des yeux.
Sourcils	Epaisseur, distance entre sourcils.
Lèvres	Epaisseur, forme, gabarit, largeur.
Bouche	Gabarit, largeur, hauteur, surface, surface de l'ouverture.
Nez	Largeur, longueur, forme, surface des narines.
Oreilles	Longueur, forme, surface.
Cheveux	Intensité, couleur, longueur, texture
Distances	D(menton, ligne des yeux), D(centre des lèvres, menton), D(yeux, naissance des cheveux), D(yeux, centre du nez), D(extrémité du menton, centre du visage), D(centre de la bouche, extrémité du nez),...
Surfaces	Extrémités intérieures des yeux et centre de la bouche Centre de la bouche et centre du front
Rapports	D(centre du visage, ouverture de la bouche) / D(menton, ouverture de la bouche) D(menton, ouverture de la bouche) / D(extrémité du nez, ouverture de la bouche)
Autres	Contour du menton, taille et contour du front

Il peut y avoir parfois une combinaison des caractéristiques et des données d'intensité pour représenter les visages.

1.5.2 Détection des visages

La détection de visages peut être faite suivant différentes approches:

- Une approche dite holiste qui considère le visage comme une entité globale. Govindaraju [5] propose par exemple une méthode de calcul pour localiser des visages. La technique proposée, utilise un gabarit déformable basé sur le contour de la tête. Ce gabarit est obtenu à partir de l'image des contours. Le visage est ainsi représenté par trois courbes (pour les deux côtés du visage et pour le contour des cheveux). A chacune de ces trois courbes est assigné un quadruplet comprenant la longueur de la courbe, la corde, la surface comprise entre cette corde et la courbe et enfin le centre de la surface. Pour

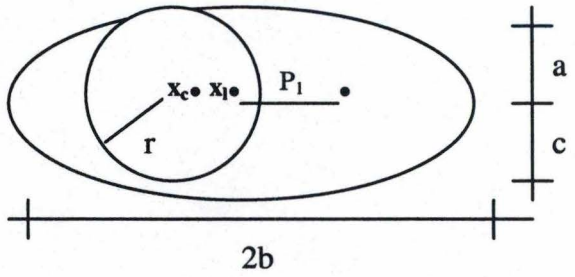
déterminer la présence d'une tête, chacun de ces trois segments doit se trouver dans une orientation particulière. Le centre de ces trois segments détermine le centre du visage. Les gabarits peuvent changer d'échelle, subir des rotations ou des translations. Une fonction de coût est élaborée pour déterminer les éventuels candidats. Dans d'autres cas, le gabarit utilisé à la forme d'une ellipse, forme se rapprochant le plus du contour du visage.

- Une autre approche consiste à extraire certaines caractéristiques telles que celles énoncées ci-avant. La technique utilisée pour extraire ces caractéristiques consiste à opérer différents pré-traitement sur l'image digitale afin de localiser les caractéristiques comme les yeux, le nez ou la bouche. Ainsi à partir des images de contour, on détermine à l'aide des gradients horizontaux les limites gauches et droites du visage et du nez tandis que les gradients verticaux permettent de détecter le sommet de la tête, les yeux, le base du nez et la bouche. Dès que les yeux ont été détectés (en utilisant la technique de projection intégrale ou à l'aide de gabarits déformables comme expliqué ci-dessous), la recherche des autres caractéristiques peut s'appuyer sur la connaissance de leur forme moyenne . On applique une fenêtre sur une partie de l'image pour affiner la recherche. Dans cette fenêtre, en utilisant la technique de projection, on peut déterminer la largeur du nez, la largeur et la hauteur de la bouche, l'épaisseur des sourcils, ... Ces caractéristiques formeront les composantes du vecteur de caractéristiques.
  NB: Certaines de ces techniques (opérateurs de gradient, projection intégrales) sont utilisés dans la normalisation des images (cfr. chap 4).

L'extraction des caractéristiques peut également se faire à l'aide de gabarits déformables. Ces gabarits sont définis par un ensemble de paramètres qui véhiculent une connaissance sur la forme attendue des caractéristiques pour pouvoir guider le processus de détection. Dès que les caractéristiques sont identifiées, la localisation globale du visage est

déterminée en utilisant l'information géométrique, c'est-à-dire la position relative des caractéristiques. Bien que n'importe quelle caractéristique du visage puisse être localisée en premier lieu, les yeux sont généralement utilisés. Yuille [5] propose par exemple une approche pour détecter les yeux dans les images en utilisant des gabarits déformables. Le gabarit choisi pour l'œil comporte entre autres les caractéristiques suivantes:

- Un cercle de rayon r centré en un point $\overline{x}_c$. Il correspond à la frontière entre l'iris et le blanc de l'œil.
- Un contour externe de l'œil modélisé par deux parties paraboliques représentant le contour supérieur et le contour inférieur. Ce contour a un centre $\overline{x}_l$ et une largeur 2b, une hauteur maximale a pour le contour supérieur et une hauteur maximale c pour le contour inférieur.
- Deux points (un pour chaque œil) correspondant au centre des blancs des yeux désignés par $\overline{x}_l + p_1$ et $\overline{x}_l + p_2$.



Au total, on dénombre 11 paramètres. Ces paramètres sont liés entre eux par certaines contraintes qui imposent par exemple que $\overline{x}_c$ et $\overline{x}_l$ soient proche l'un de l'autre, que la largeur 2b soit environ 4 fois le rayon r de l'iris, que le centre du blanc de l'œil $\overline{x}_l + p_1$ soit à mi-chemin entre le centre de l'œil $\overline{x}_l$ et le contour. Les paramètres sont alors introduits dans une fonction d'énergie qui doit être minimisée pour avoir un le meilleur ajustement des paramètres au candidat potentiel.

1.5.3 Identification des visages

Une procédure typique d'étapes dans la procédure d'identification est la suivante:

1. Déterminer un ensemble de caractéristiques indépendantes pour représenter un visage.
2. Représenter les visages connus par leurs caractéristiques dans la base de données.
3. Déterminer les valeurs de caractéristiques des nouveaux visages.
4. Utiliser une méthode de mise en correspondance pour obtenir le meilleur ajustement des deux visages.

Méthode de mise en correspondance

Différentes approches sont utilisées pour l'étape de mise en correspondance, parmi celles-ci citons:

- La Distance euclidienne: C'est la mesure la plus couramment utilisée. Le but est de calculer la distance entre le vecteur des caractéristiques du visage à tester et celui de chaque visage de la base de données.
- Le clustering: Les techniques de 'clustering' peuvent être utilisées pour trouver le degré de ressemblance entre un visage inconnu et un visage connu.
- L'ensemble partionnant: Les vecteurs de caractéristiques de différents visages diffèrent de manière significative pour au moins une caractéristique. Les visages connus peuvent être réduits à un petit sous-ensemble en rejetant chaque visage qui diffère de manière significative pour au moins une caractéristique.
- La corrélation: Des gabarits de visage sont utilisés pour calculer la ressemblance entre un visage connu et un visage inconnu.
- Autres: Parfois une combinaison de ces méthodes est utilisée. Une autre mesure utilisée pour trouver la ressemblance entre deux visages est la

différence entre les sommes de leurs caractéristiques. Les arbres de décision et les moindres carrés sont ainsi utilisés.

1.6 Synthèse et commentaires

A côté des deux classes de techniques énoncées, d'autres techniques spécifiques ont été développées. Citons entre autres les techniques utilisant les réseaux neuronaux, l'expansion de Karhunen-Loeve, les moments algébriques ou encore les lignes d'iso-densité.

Un des problèmes majeurs dans la reconnaissance est de trouver une manière de représenter des visages. Traditionnellement, les modèles de calcul en reconnaissance de visages représentent les visages en termes

- de descripteurs géométriques incluant des distances, des angles et des surfaces entre des caractéristiques élémentaires telles que les yeux, le nez ou le menton.

ou

- de paramètres de gabarits ou de lignes d'isodensité.

La difficulté dans la représentation de visage par un ensembles de caractéristiques est qu'elle suppose une connaissance préalable sur ce que sont les caractéristiques et sur les relations entre ces caractéristiques.

Les modèles connectionnistes fournissent une voie différente pour représenter les visages en utilisant des caractéristiques déduites de la structure statistique d'un ensemble de visages. Cette approche est généralement connue comme l'approche en composantes principales. En gros, ces modèles représentent les visages, explicitement ou via une architecture de réseaux neuronaux, par une somme pondérée de vecteurs propres de la matrice de covariance des images de départ. Cette approche est étudiée dans le chapitre 2 et formera la base du travail réalisé.

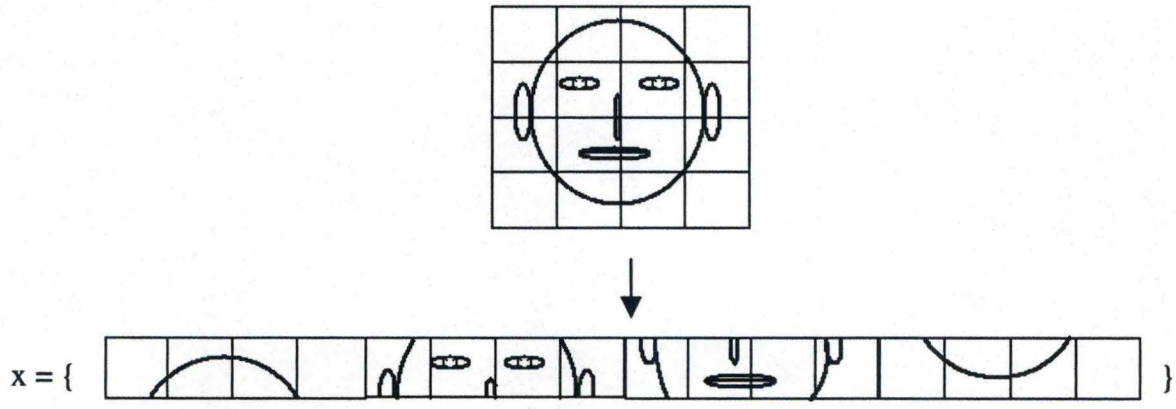
CHAPITRE 2

CHAPITRE 2 Analyse en composantes principales

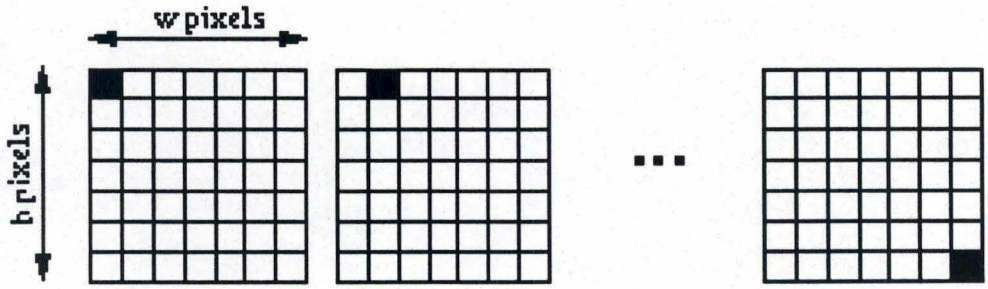
2.1 L'espace des images et l'espace des visages

Une image digitale peut être modélisée par une fonction de 2 ou 3 variables. Dans le cas le plus simple, les arguments sont les coordonnées (x,y) d'un plan et la valeur de la fonction correspond à l'intensité lumineuse au point considéré. On appelle une telle image 2-D une image d'intensité et on la note I(x,y). L'ensemble des valeurs que peut prendre la fonction est limité. Par convention, dans les images monochromatiques, la valeur la plus basse correspond au noir et la plus haute au blanc. Les valeurs entre ces limites sont appelées niveaux de gris. Dans la plupart des cas, le nombre de niveaux de gris choisi est égal à 256 (0 .. 255).

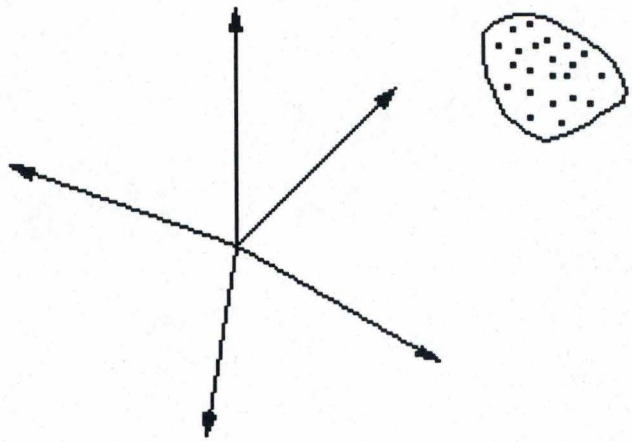
Une image qui doit être traitée par un ordinateur doit être représentée par une structure de données appropriée. La matrice est la structure de données la plus courante pour la représentation de l'image. Les éléments de la matrice sont des nombres entiers correspondant à l'intensité (au niveau de gris) ou à une autre propriété du pixel (contraction de picture element) correspondant dans la grille d'échantillonnage. L'image peut également être représentée sous forme de vecteur. Si la taille des images est w*h pixels, le nombre de composants du vecteur sera w*h. Chaque pixel est codé par un élément du vecteur. La construction du vecteur à partir de l'image est obtenue par concaténation des rangées de l'image comme le montre la figure ci-dessous:



Le vecteur décrit précédemment appartient à un espace. Cet espace est l'espace des images, l'espace de toutes les images dont les dimensions sont w*h pixels. La base sous-tendant cet espace est composée des éléments suivants:



Pratiquement tous les visages se ressemblent. Ils ont deux yeux, une bouche, un nez, etc. qui sont situés à la même place dans le visage. C'est pourquoi les vecteurs caractérisant les visages sont groupés dans l'espace des images.



L'espace entier des images n'est dès lors pas un espace optimal pour la description des visages. L'idée est donc de trouver l'espace qui décrit le mieux les visages. Les vecteurs de base de ce nouvel espace sont appelés composantes principales.

Le but de méthode présentée, en l'occurrence l'analyse en composantes principales (ACP), est de réduire la dimension d'un ensemble ou d'un espace de sorte que la nouvelle base décrive au mieux les composants typiques de l'ensemble. L'analyse en composantes principales tend à capter les variations dans un ensemble et à exprimer ces variations à l'aide d'un nombre réduit de variables.

L'analyse en composantes principales a d'abord été développé par des statisticiens. Elle a également été reformulée par une approche s'appuyant sur les réseaux neuronaux et les mémoires auto-associatives. Nous analyserons les deux approches qui sont fort complémentaires pour la compréhension de l'ACP. En effet, l'approche réseaux neuronaux donne une idée plus intuitive de ce qu'est l'ACP alors que l'approche statistique est plus rigoureuse mathématiquement.

2.2 Composantes principales et réseaux neuronaux ([1] et [2])

Cette approche est basée sur les travaux d'Andersen, Silverstein, Ritz, Jones et Kohonen sur les mémoires auto-associatives. Les mémoires auto-associatives sont un cas particulier des mémoires associatives où les stimuli présentés sont associés à eux-mêmes. Les mémoires auto-associatives linéaires sont composées de neurones linéaires (c-à-d des neurones dont la réponse est une fonction linéaire de l'activation) et fonctionnent comme des mémoires auto-adressables. Chaque neurone est relié à tous les autres neurones de la mémoire par des connexions modifiables par apprentissage. Kohonen utilise des visages comme stimuli d'entrée.

2.2.1 Description du modèle

L'image digitale I(x,y) du visage k est représenté sous forme d'un vecteur $\boldsymbol{x_k}$ par concaténation des rangées de pixels de l'image. Chaque élément du vecteur code l'intensité lumineuse d'un pixel qui correspond à un neurone. Le vecteur $\boldsymbol{x_k}$ est normalisé de sorte que $x_k^T.x_k = 1$. L'ensemble des M visages qui constituent l'ensemble d'apprentissage est représenté par une matrice $\boldsymbol{X}$ où les éléments de la kième colonne correspondent aux éléments du vecteur $\boldsymbol{x_k}$. Les visages sont stockés dans une mémoire

auto-associative où le poids de la connexion entre chaque neurone est proportionnel à la corrélation entre les pixels.

La valeur des connexions est représentée par une matrice W telle que

$$W = \sum_{k=1}^{M} x_k x_k^T = XX^T$$

La reconstruction du kième visage est alors obtenue par

$$\tilde{x}_k = Wx_k$$

Où $\tilde{x}_k$ représente l'estimation du kième visage dans la mémoire. La qualité de cette estimation peut être mesurée en calculant le cosinus de l'angle entre $\tilde{x}_k$ et x_k

$$\cos(x_k, \tilde{x}_k) = \frac{\tilde{x}_k^T x_k}{\|\tilde{x}_k\| \|x_k\|}$$

Un cosinus égal à 1 indique une reconstruction parfaite du stimulus. Plus le cosinus est proche de 1, plus il est probable que l'image a été apprise. Certains vecteurs possèdent la propriété de donner effectivement un cosinus de 1. Ils restent inchangés lorsqu'ils sont rappelés par la matrice à l'exception d'une multiplication par une constante qui correspond à une amplification par la mémoire. Ils s'appellent les vecteurs propres de la matrice de connexion. Le coefficient d'amplification s'appelle valeur propre.

Les vecteurs propres: des macrocaractéristiques

Les vecteurs propres s'interprètent également comme des macro-caractéristiques: chaque stimulus appris peut se reconstituer comme une somme pondérée des vecteurs propres. Chaque visage peut donc se représenter par l'ensemble des coefficients de sa reconstruction. D'un point de vue statistique, cela revient à effectuer l'analyse en composantes principales (ACP) où les composantes principales sont les vecteurs propres. Les vecteurs propres s'ordonnent en fonction de la valeur propre (le premier vecteur propre est celui qui possède la plus grande valeur propre). Pour l'analyse en composantes principales, la valeur propre donne la part de la variance expliquée par la composante principale.

La matrice W définie ci-dessus est une matrice définie semi-positive (toutes ses valeurs propres sont réelles positives ou nulles).

Par conséquent W peut être exprimée par une somme pondérée de ses vecteurs propres.

$$W = \sum_{l=1}^{L} \lambda_l u_l u_l^T = U \Lambda U^T$$

où
- $U^T U = I$
- u_l est le lième vecteur propre de W
- λ_l est la lième valeur propre
- I est la matrice unité
- Λ représente la matrice diagonale L x L des valeurs propres
- U est la matrice des vecteurs propres
- L est le rang de la matrice W

La matrice des visages X peut être exprimée en terme de vecteurs propres et de valeurs propres en utilisant la technique de décomposition en valeurs singulières:

$$X = U \Delta V^T$$

où

- U est la matrice des vecteurs propres de XX^T
- V est le vecteur des valeurs propres de XX^T
- Δ est la matrice diagonale des valeurs singulières qui est égale à la racine carrée des valeurs propres de XX^T ou de $X^T X$ (qui ont les mêmes valeurs propres).
-

Compte tenu de ce qui précède, l'estimation d'un visage peut désormais être représentée par une somme pondérée de vecteurs propres

$$\tilde{x}_k = W x_k = \sum_{l=1}^{L} \lambda_l u_l u_l^T x_k = \sum_{l=1}^{L} \lambda_l \gamma_l u_l \text{ avec } \gamma_l = u_l^T x_k$$

Les poids γ_l sont les projections de chaque visage sur les vecteurs propres. Ces poids peuvent être interprétés comme étant une indication de l'importance d'un vecteur propre donné (ou macrocaractéristique) à caractériser un visage particulier.

Erreur de Widrow-Hoff

Les performances d'un auto-associateur peuvent être améliorées en utilisant la règle d'apprentissage de Widrow-Hoff ou règle Delta. Cette règle d'apprentissage corrige la différence entre la réponse donnée du système et la réponse désirée en changeant de manière itérative les poids de la matrice $\boldsymbol{W}$ comme suit:

$$W_{[t+1]} = W_{[t]} + \eta(X - W_{[t]}X)X^T$$

Où η est un coefficient positif (typiquement plus petit que 1).

Lorsque la procédure d'apprentissage converge, les visages stockés sont parfaitement reconstitués. Les visages inconnus sont déformés par la mémoire en fonction inverse de la ressemblance d'ensemble avec les visages appris.

Avantages des vecteurs propres

- Un premier avantage de l'utilisation des vecteurs propres pour représenter des visages est qu'ils sont déterminés à posteriori.
- Un second avantage est qu'ils reflètent la structure statistique de l'ensemble des visages dans lequel ils sont extraits.

2.2.2 Analyse de l'information perceptible dans les visages

Un aspect intéressant de l'approche ACP est qu'elle fournit un outil d'analyse de l'information perceptible dans les visages. Ainsi O'Toole et al. suggèrent une dissociation du type d'information fourni par différents ensembles de vecteurs propres. Les vecteurs propres avec des grandes valeurs propres véhiculent une information sur la forme de base et la structure des visages. Ils permettent d'effectuer des classifications des visages suivant des critères généraux comme le sexe ou la race. Par contre, les vecteurs propres avec des petites valeurs propres tendent à capter l'information

spécifique des visages. Ces vecteurs sont utiles pour distinguer un visage particulier d'un autre.

2.3 L'analyse en composante principale et l'approche statistique

2.3.1 Composantes principales

L'idée de base de l'analyse en composantes principales est de décrire les variations d'un ensemble de données à variables multiples, corrélées entres elles, en termes de variables non corrélées. Soient $x_1, \ldots, x_p$ les variables corrélées et $y_1, \ldots, y_k$ les variables non corrélées :

$$y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1p}x_p$$
$$\ldots$$
$$y_k = a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kp}x_p$$

Les variables $y_1, \ldots, y_k$ sont appelées **composantes principales** et sont des combinaisons linéaires des variables $x_1, \ldots, x_p$. Ces composantes principales permettent de mieux quantifier l'information contenue dans les données ou plus exactement les variations subies par ces dernières. Il existe au maximum p composantes principales qui expriment 100 % de la variance des données. En général, un nombre $k < p$ de ces composantes suffit pour capter l'essentiel de la variance. L'ACP permet donc de réduire le nombre de variables permettant de caractériser les données étudiées. Chaque composantes principales y_i exprime un certain pourcentage de la variance totale.

Notation matricielle :

Soit X le vecteur des variables originales, A la matrice des poids a_{ij} et Y le vecteur des composantes principales.

$$Y = A * X$$

On a également:

$$y_i = A_i^T * X$$

Dans le cas des images, X_i est le vecteur correspondant à l'image i:

$$X_i = \begin{bmatrix} x1 \\ x2 \\ \dots \\ x(h*w) \end{bmatrix}$$

et X la matrice formée à partir des vecteurs des différentes images :

$$X = [X_1 \quad X_2 \quad \dots \quad X_M]$$

2.3.2 Variance des composantes principales

La variance de la composante principale y_i, σ^2_{yiyi}, est donnée par l'expression suivante:

$$\sigma^2_{yiyi} = \sigma^2_{(A_i^T * X)} = A_i^T * C_x * A_i$$

Où C_x est la matrice (symétrique) des variances et covariances pour l' ensemble des visages.

C_x est définie par l'expression suivante :

$$C_x = E[(X - E(X)).(X - E(X))^T]$$

ou encore :

$$C_x = X * X^T = \begin{bmatrix} \sigma^2_{x1x1} & \sigma^2_{x1x2} & \dots & \sigma^2_{x1x(w*h)} \\ \sigma^2_{x2x1} & \sigma^2_{x2x2} & \dots & \sigma^2_{x2x(w*h)} \\ \dots & \dots & \dots & \dots \\ \sigma^2_{x(w*h)x1} & \sigma^2_{x(w*h)x2} & \dots & \sigma^2_{x(w*h)x(w*h)} \end{bmatrix}$$

où σ^2_{xixj} représente la covariance entre le pixel x_i et le pixel x_j.

La matrice C_x n'est en général pas diagonale. Le but de la méthode présentée est alors de construire un espace des visages où les composantes ne sont pas corrélées entre elles. Ce qui signifie que la matrice C_y des variances et covariances de ces nouvelles composantes serait diagonale.

$$C_y = Y * Y^T = \begin{bmatrix} \sigma^2_{y1y1} & 0 & \dots & 0 \\ 0 & \sigma^2_{y2y2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \sigma^2_{y(w*h)y(w*h)} \end{bmatrix}$$

où

- y_i est le vecteur colonne décrivant le visage X_i dans la base de l'espace des visages.
- Y est la matrice contenant les vecteurs y_i.

Considérons P la matrice de transformation telle que :

$$Y = P^T * X \quad \text{et} \quad X = P * Y$$

La question est de savoir comment doit être P pour que C_y soit une matrice diagonale

$$C_y = Y * Y^T = P^T * X * X^T * P = P^T * C_x * P$$

Ainsi C_y est la rotation de C_x par P. Si P est choisi comme étant la matrice des vecteurs propres de C_x, on a:

$$C_x * P = \Lambda * P$$

Où Λ est la matrice diagonale contenant les valeurs propres de C_x. Dès lors :

$$C_y = P^T * \Lambda * P = \Lambda * P^T * P = \Lambda$$

car $P^T * P = I$ avec I est la matrice unité (En effet, une matrice M définissant un changement de base orthonormal est une matrice orthogonale et est telle que $M^T = M^{-1}$). Il en résulte que C_y est une matrice diagonale dont les éléments de la diagonale sont les valeurs propres de C_x. Les valeurs propres de C_x sont dès lors les variances des composantes de l'espace des visages.

2.3.3 Réduction des dimensions

Le but de l'ACP est de réduire les dimensions de l'espace de travail. Le nombre maximal de composantes principales est égal au nombre de variables de l'espace d'origine. Pour pouvoir réduire les dimensions, certaines composantes principales ne

doivent pas être prises en compte. La question se pose alors de savoir quels vecteurs de base seront maintenus et quels sont ceux qui ne seront pas considérés.

La reconstruction d'un visage par les composantes y_i est donnée par la formule suivante :

$$x_i = P * y_i$$

Si p_i sont les vecteurs colonnes de P alors la reconstruction d'un visage utilisant certains composantes y_{ij} est :

$\tilde{x}_i = \sum_{j=1}^{M1} y_{ij}.p_j + \sum_{j=M1+1}^{M} y_{ij}.p_j = \sum_{j=1}^{M1} y_{ij}.p_j + \sum_{j=M1}^{M} a_j.p_j$ où a_j est une constante dont la valeur sera déterminée.

Cela entraîne une erreur . Le vecteur correspondant à l'erreur est :

$$e = x_i - \tilde{x}_i = \sum_{M1+1}^{M} (y_{ij} - a_j).p_j$$

Le vecteur d'erreur est un vecteur aléatoire et sa norme est calculée par la moyenne du carré du vecteur :

$$< e^2 >= E(e^T * e) = E\left(\sum_{j=M1+1}^{M} \sum_{l=M1+1}^{M} (y_{ij} - a_j).(y_{ij} - a_l) \right).p_i^T.p_i = \sum_{i=M1+1}^{M} E(y_i - a_i)^2$$

Le but est de minimiser l'erreur. On obtient cela en résolvant :

$$\frac{\partial}{\partial a_i} < e^2 >= 0$$

ou $\quad -2 * E((y_i - a_i)) = 0$

ou encore $\quad a_i = E(y_i) \quad$ pour $\quad < e^2 >= \sum_{i=M1+1}^{M} E(y_i - E(y_i))^2 = \sum_{i=M1+1}^{M} \lambda_i$

c'est-à-dire la somme des variances des composantes omises.

Si les valeurs propres sont classées par ordre décroissant, les dernières valeurs propres (et les vecteurs propres correspondants) peuvent être omises. Cela signifie que certaines composantes principales peuvent être omises car elles expriment qu'une faible

quantité 'd'information', la plus grande partie de cette information étant contenue dans les autres composants principaux. La quantité d'information que la ième composante véhicule est déterminé par sa valeur propre λ_i. Généralement les composantes principales sont ordonnées en ordre décroissant de leur valeur propre et les dernières ne sont pas tenues en compte. Ce fait est particulièrement si on tient compte du fait que pour les visages la décroissance des valeurs propres est exponentielle.

2.3.4 Dimensions requises

Il n'est pas aisé de déterminer à priori quelles seront les dimensions de l'espace des visages. Cependant, un simple test peut donner des indications sur le nombre de dimensions à considérer. En sachant que les valeurs propres décroissent de manière exponentielle, nous pouvons éliminer les dernières valeurs propres qui sont pratiquement identiques :

$$\lambda_N = \lambda_{N+1} = \lambda_{N+2} = ... = \lambda_M \quad \text{avec } 1 < N < M$$

CHAPITRE 3

Chapitre 3: Reconnaissance de visages par la méthode des visages propres

3.1 Introduction:

La méthode des visages propre est basée sur une approche qui décompose les images de visages dans un ensemble d'images de traits caractéristiques appelés visages propres (eigenfaces). Ces visages peuvent être considérés comme étant les composantes principales de l'ensemble initial des visages. La reconnaissance est alors effectuée en projetant une nouveau visage (visage à tester) dans le sous-espace sous-tendu par les visages propres ('espace des visages') et en classant ce visage par comparaison de sa position dans l'espace des visages avec la position de visages de personnes connues.

3.2 L'approche visage propre:

Une approche simple pour extraire l'information contenue dans l'image d'un visage est de capter d'une manière ou d'une autre les variations dans un ensemble d'images de visages et d'utiliser cette information pour encoder et comparer les images de visages. Mathématiquement, cela revient à rechercher les composantes principales de la distribution de visages ou les vecteurs propres de la matrice de covariance de l'ensemble des images de visages, considérant une image comme un point (vecteur) dans un espace multidimensionnel.

Les vecteurs propres peuvent être considérés comme un ensemble de traits généralisés qui caractérisent les variations dans les images de visages. Chaque image contribue plus ou moins à chaque vecteur propre de telle sorte que chaque vecteur propre formé à partir d'un ensemble d'images de visages apparaît comme une sorte de visage fantomatique appelé visage propre.

Chaque visage peut être exprimé exactement par une combinaison linéaire des visages propre. Chaque visage peut également être approximé en n'utilisant que les 'meilleurs' visages propres, c'est-à-dire ceux qui possèdent les plus grande valeurs

propres et donc qui prennent en compte les plus fortes variations dans l'ensemble des visages. Ces vecteurs propres sous-tendent un sous-espace multidimensionnel de l'espace des images, appelé espace des visages.

La reconnaissance de visage par l'approche visages propres comporte les opérations d'initialisation suivantes:

1. Acquérir un ensemble d'images de visage (ensemble d'apprentissage).
2. Calculer les visages propres à partir de l'ensemble de d'apprentissage et ne garder que les M visages propres correspondant aux plus grandes valeurs propres. Ces M visages définissent l'espace des visages. Lorsque de nouveaux visages sont expérimentés, les visages propres sont mis à jour ou recalculé.
3. Calculer dans l'espace M-dimensionnel la distribution de poids correspondant pour chaque individu connu en projetant leur visage dans l'espace des visages.

Ayant initialisé le système, les étapes suivantes sont utilisées pour reconnaître un 'nouveau' visage:

1. Calculer un ensemble de poids, à partir d'une image d'entrée et des M visages propres en projetant l'image d'entrée sur chaque visage propre.
2. Déterminer si l'image est un visage propre en observant si l'image est suffisamment proche dans l'espace des visages.
3. S'il s'agit d'un visage, déterminer si oui ou non c'est une personne connue.
4. (Optionnel) Adapter les visages propres et/ou poids.
5. (Optionnel) Si un visage inconnu est vu plusieurs fois, l'incorporer dans les visages connus.

3.3 Calcul des visages propres

Considérons l'image d'intensité d'un visage I(x,y) comme un tableau à deux dimensions N x N de valeurs d'intensité (codés sur 8 bits). Cette image peut également être représentée par un vecteur de dimension N^2. Ainsi une image de taille 256 x 256 peut être représentée par un vecteur de dimension 65.536 ou de façon équivalente par un point dans un espace 65.536-dimensionnel. Un ensemble d'images correspond à une collection de points dans cet espace gigantesque.

Les images de visages, étant globalement semblables, ne seront pas distribués de manière aléatoire dans cet espace d'images gigantesque et peuvent donc être définies dans un sous-espace de faibles dimensions. L'idée de base de l'analyse en composantes principales est de trouver les vecteurs qui prennent le mieux en compte la distribution des images de visages dans l'espace entier des images. Ces vecteurs définissent le sous-espace des images de visages appelé espace des visages. Ces vecteurs qui sont les vecteurs propres de la matrice de covariance des images de visages de l'ensemble d'apprentissage sont les visages propres.

Considérons l'ensemble des M images de visages de la base de données $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$. Le visage moyen de cet ensemble de visages est défini par

$$\Psi = \frac{1}{M}\sum_{i=1}^{i=M}\Gamma_i$$

Chaque visage diffère de la moyenne par le vecteur $\phi_i = \Gamma_i - \psi$. Cet ensemble de vecteurs très larges est soumis à l'analyse des composantes principales qui recherche un ensemble de M vecteurs orthonormaux $\mathbf{u_i}$ qui décrivent le mieux la distribution des données. Le kième vecteur $\mathbf{u_k}$ est choisi de manière telle que :

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(u_k^t.\Phi_n)^2$$

soit maximal, considérant également que

$$u_l^t.u_k = \delta_{lk} = 1, \text{ si l =k.}$$

$$= 0, \text{ autrement.}$$

Les vecteurs $\mathbf{u_k}$ et scalaires λ_k sont respectivement les M vecteurs propres et les valeurs propres significatifs de la matrice de covariance C où

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n . \Phi_n^t$$

$$= A.A^t$$

avec la matrice $A = [\Phi_1 \Phi_2 ... \Phi_M]$

La matrice C étant une matrice N^2 x N^2, déterminer les vecteurs propres et valeurs propres de cette matrice est une tâche pratiquement insoluble pour des images de taille courante. Nous devons donc trouver une méthode qui soit faisable d'un point de vue calcul pour déterminer les vecteurs et valeurs propres.

Considérons les vecteurs propres $\mathbf{v_i}$ de $A^t.A$ tels que:

$$A^t A \mathbf{v_i} = \mu_i \mathbf{v_i}$$

En multipliant chaque membre de l'équation par A, on obtient:

$$A A^t A \mathbf{v_i} = A \mu_i \mathbf{v_i}$$

Ou encore:

$$A A^t (A \mathbf{v_i}) = \mu_i (A \mathbf{v_i})$$

On constate que $A \mathbf{v_i}$ sont les vecteurs propres de $C = A A^t$. Poursuivant cette analyse, construisons la matrice $L = A^t A$ de dimension M x M dont les éléments sont:

$L_{mn} = \Phi_m^t \Phi_n$ et cherchons les M vecteurs propres $\mathbf{v_i}$. Par des combinaisons linéaires des visages de l'ensemble de départ, ces vecteurs vont former les visages propres $\mathbf{u_l}$:

$$\mathbf{u_l} = A \mathbf{v_i} \qquad l = 1, 2, .., M$$

Par cette analyse, les calculs sont considérablement réduits du nombre de pixels dans les images (N^2), à l'ordre du nombre d'images dans l'ensemble de départ. En général, l'ensemble de départ est relativement petit ($M << N^2$) et les calculs deviennent plus gérables. Les valeurs propres nous permettent d'ordonner les vecteurs propres en fonction de leur importance à caractériser les variations dans les visages et dès lors de pouvoir se limiter à un sous-ensemble de visages propres significatifs. La question de savoir combien de visages propres faut-il retenir trouve sa réponse dans un compromis entre la précision de reconnaissance et le temps de calcul. Chaque visage propre additionnel augmente ce temps de calcul mais ajoute de la précision. Pour des petites

bases de données, ce problème n'est pas vital mais il devient fort important pour des bases de données très larges.

3.4 Utilisation des visages propres pour la reconnaissance

Lors de l'opération de reconnaissance, un nouveau visage est projeté dans l'espace des visages. Il est représenté par ses composantes ou poids ω_i où

$$\omega_i = u_i^t(\phi - \Psi) \qquad i = 1, 2, \ldots, M'$$

M' étant le nombre de visages propres retenus. Les différents poids forment un vecteur $\Omega^t = [\omega_1\omega_2 \ldots \omega_{M'}]$ qui décrit la contribution de chaque visage propre dans le visage à tester. Ce vecteur peut alors être utilisé dans un algorithme standard de reconnaissance afin de déterminer qui, parmi un nombre prédéfini de classe de visages, décrit le mieux le visage en question. La méthode la plus simple pour déterminer quelle classe de visage décrit le mieux le visage à tester est de minimiser la distance euclidienne

$$\varepsilon_k = \left\|(\Omega - \Omega_k)\right\|^2$$

où Ω_k est un vecteur qui décrit la kième classe. Les classes Ω_i sont calculées en effectuant la moyenne des résultats des poids de plusieurs images d'un même individu. Un visage est classé comme appartenant à la classe k lorsque le minimum ε_k est en deçà d'un certain seuil θ_k. Autrement, le visage est classé comme inconnu et optionnellement est utilisé pour créer une nouvelle classe de visage.

3.5 Utilisation de l'espace des visages pour la localisation des visages

L'algorithme de reconnaissance décrit précédemment suppose qu'on a une image de visage centrée, que cette image a la même taille que les images de l'ensemble d'apprentissage et que ceux des visages propres. On a donc besoin d'une méthode fiable pour localiser un visage dans une scène afin de procéder à la reconnaissance. On peut utiliser la connaissance de l'espace des visages pour localiser et reconnaître des visages dans des images .

Créer le vecteur des poids revient à projeter une image dans l'espace des visages de faibles dimensions. C'est donc un "mapping many-to-one". Les images de visages ne changent pas de manière radicale lorsqu'elles sont projetées dans l'espace des visages, tandis que la projection d'images qui ne sont pas des visages est assez différente de l'image originale. Soit $\Phi = \Gamma - \psi$ l'image d'entrée ajustée par la moyenne des images de l'ensemble de départ et $\Phi_f = \sum_{k=1}^{M} \omega_k u_k$, sa projection dans l'espace des visages. Soit

$$\varepsilon = \left\| \Phi - \Phi_f \right\|$$

L'idée est alors d'utiliser cette métrique ε pour détecter la présence de visages dans une scène. A chaque endroit de l'image, on calcule ε entre la sous-image locale et l'espace de visage. Cette distance est utilisée comme mesure de la présence de visages. Le résultat du calcul de la distance de l'espace des visages en chaque point de l'image est une carte de visage ε(x,y). L'inconvénient de cette méthode est qu'elle est assez chère d'un point de vue calcul.

3.6 Problèmes

3.6.1 L'arrière-plan

Dans l'analyse précédente, on a ignoré le problème de l'arrière-plan. En pratique, l'arrière-plan peut affecter de manière significative les performances de la reconnaissance étant donné que l'analyse des visages propres ne permet pas de distinguer le visage du reste de l'image. Comment éliminer cet arrière-plan?
Deux méthodes sont proposées par Turk ([14]) :

- La première consiste à multiplier l'image du visage par une fenêtre gaussienne bi-dimensionnelle, centrée sur le visage. Cette méthode a comme désavantage de compliquer la mesure de la distance par rapport à l'espace des visages.
- La deuxième méthode consiste à éliminer l'arrière-plan dès la saisie des images de l'ensemble de départ. Ceci peut se faire par un marquage des

contours des visages de l'ensemble de départ. A partir de ce contour, un masque binaire est réalisé pour définir la région du visage pour chaque étape ultérieure.

3.6.2 Echelle (taille du visage) et orientation

Pour que le système puisse bien fonctionner, la taille du visage dans l'image d'entrée doit être proche de celles des visages propres. On peut soit changer l'échelle du visage d'entrée, soit utiliser des visages propres multi-échelles. De manière analogue on peut utiliser différentes échelles du visage d'entrée. Pour ce qui est du problème de l'orientation, une méthode est d'utiliser des opérateurs de symétrie qui estiment l'orientation de la tête et une fois cette orientation déterminé, il est possible d'effectuer une rotation de l'image pour l'aligner avec les visages propres.

CHAPITRE 4

CHAPITRE 4 EXPERIMENTATION

4.1 Matériel et outils de travail

4.1.1 Système et langage

Le programme réalisé a été écrit en langage C, sur station de travail HP fonctionnant avec le système d'exploitation UNIX.

4.1.2 Base de données

Les images qui ont servi à l'expérimentation ont été extraites de la base de données du SIC (Signal and Image Center) de l'ERM (Ecole Royale Militaire sise à Bruxelles). Cette base de données contient des images en vue de face et de profils d'élèves, d'assistants et de professeurs de l'ERM. Les images en vue frontale ont une taille 768*576, sont en couleur (8 bits soient 256 couleurs) et ont une résolution spatiale de 81 dpi. Afin de réduire le volume et le temps de calcul pour la méthode utilisée, la taille des images a été réduite à 100*130, la résolution a été maintenue et les images ont été transformées en images en niveau de gris (8 bits soient 256 niveaux de gris).

a) Conservation des données:

Chaque image est sauvegardée dans un fichier sous un format spécifique au SIC désigné par le suffixe RMA (Royal Military Academy). Ce fichier contient:

- Un header de 512 bytes. Il contient différentes informations telles que: Un identificateur, le type des données (données entières, données en virgule flottante simple précision, double précision, etc.) utilisées pour les valeurs d'intensité de l'image, l'offset de l'image (sa position) caractérisé par deux coordonnées x et y, la taille de l'image (largeur et hauteur) et le nom de l'image.

- Les données représentant la valeur de l'intensité des pixels de l'image.

b) Traitement des données:

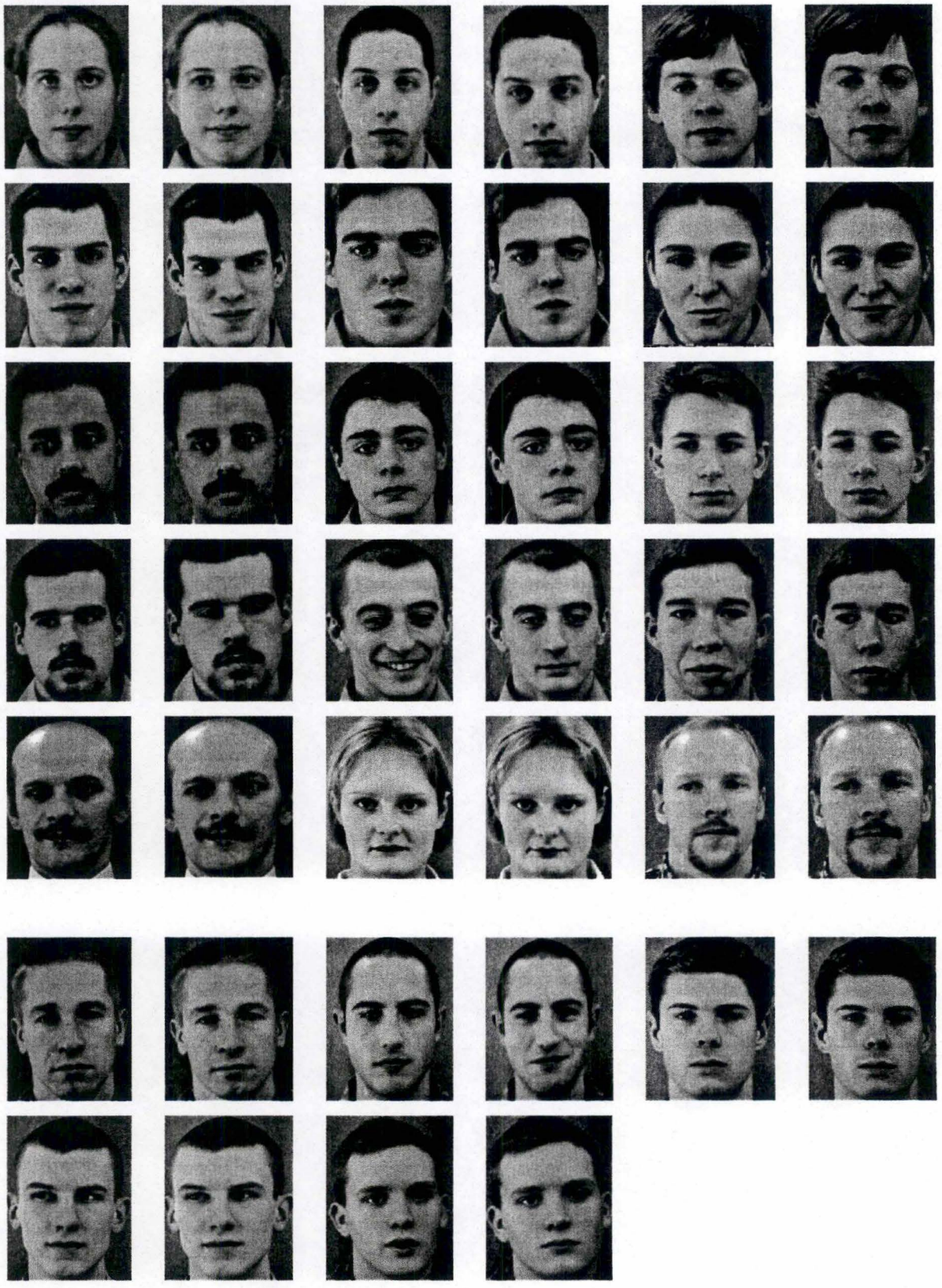
Un type particulier a été défini pour manipuler les données. Ce type appelé IP_image est une structure contenant les données du header et les valeurs d'intensité des pixels reprises sous forme d'un tableau 2D. Lors du traitement, la représentation mathématique utilisée pour les images est la matrice et/ou le vecteur.

c) Base de données pour l'expérimentation :

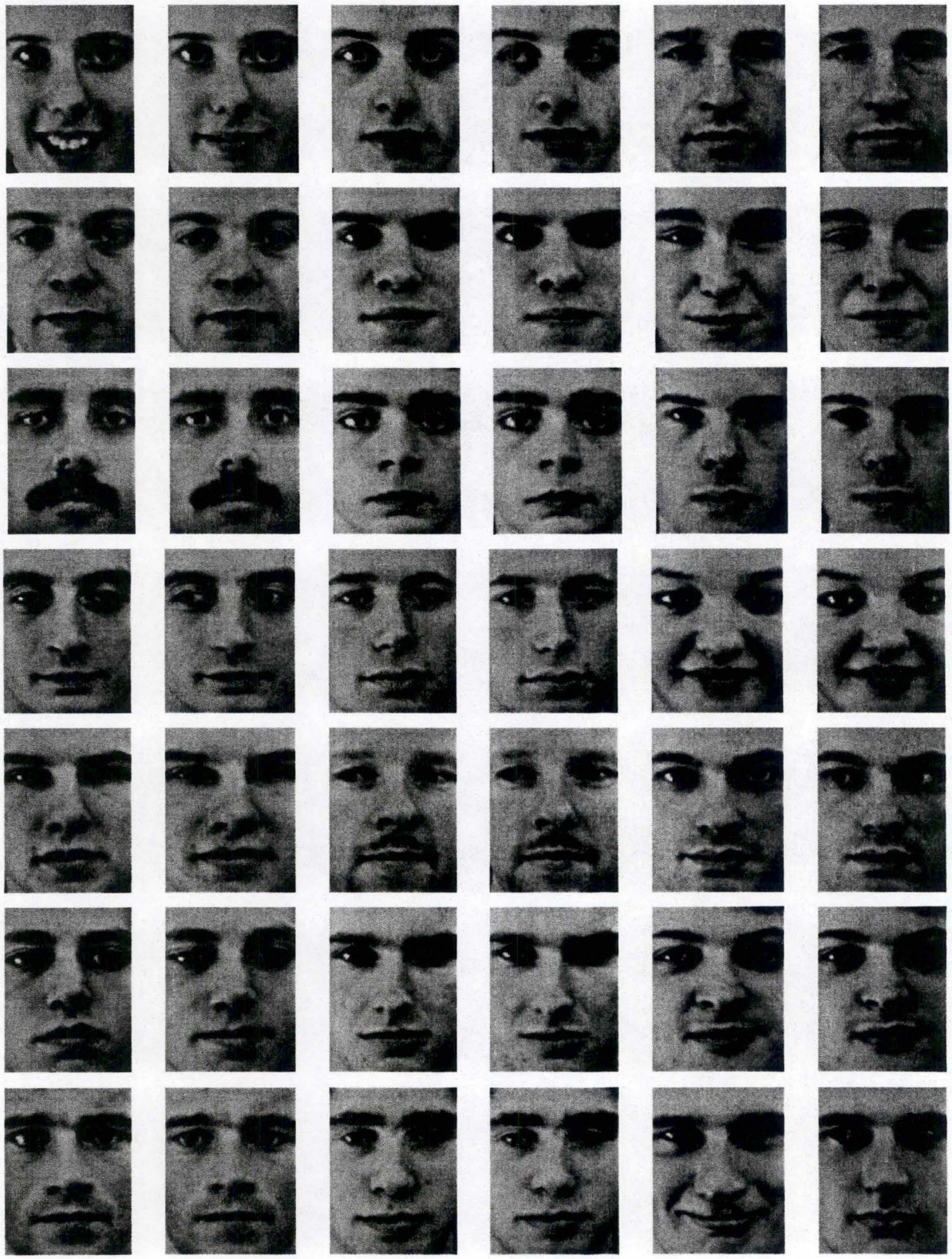
Deux 'types' d'images ont été utilisées pour la génération des visages propres et pour l'exécution des tests. En effet dans un premier temps, ont été pris en considération des images que l'on pourrait qualifier d'images non normalisées. Ces images sont des images provenant de la base de données et résultent de deux transformations qui sont une réduction de la taille (de 768*576 vers 100*130) et une transformation des images en niveaux de gris (initialement les images sont en couleur). Dans la mesure ou les résultats obtenus étaient moyennement bons et assez prometteurs, un second type d'images a été généré. Ce sont des images qualifiées ici d'images "normalisées". On entend par normalisées, des images qui ont subi des transformations successives (transformation en niveaux de gris, réduction de taille, rotation, étirement,...) de manière à se départir des problèmes de position, d'échelle ou de rotation du visage de l'image et de manière aussi a éliminer l'arrière-plan qui est également source d'erreurs. La normalisation des images a pour effet d'obtenir un alignement horizontal des yeux dans les images, un écartement et une position quasi identique.

Pour chacun de ces deux types d'images, deux ensembles ont été formés. Un premier ensemble, appelé ensemble d'apprentissage, est constitué d'images de la nouvelle base de données et a pour but de générer les visages propres. Un second ensemble regroupe les images servant à tester des candidats potentiels. Ce second ensemble sera appelé ensemble test. Ci-après est représenté un extrait de l'ensemble d'apprentissage pour les deux types d'images traites.

Ensemble d'apprentissage avec images non normalisées



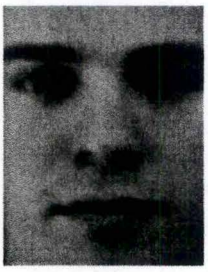
Ensemble d'apprentissage avec images 'normalisées'



4.2 Implémentation de la méthode des visages propres

La méthode des visages propres analysée plus en détail au cours des chapitres 2 et 3 sera analysée ici au niveau de son implémentation. L'algorithme suivi est celui décrit dans le chapitre 3. Pour rappel, les principales étapes de cet algorithme sont les suivantes :

1. **Acquérir un ensemble d'images de visage qui constituera l'ensemble d'apprentissage.**
   (Voir ci-avant)
2. **Calculer les visages propres à partir de l'ensemble d'apprentissage et ne garder que les M visages propres correspondant aux plus grandes valeurs propres. Ces M visages définissent l'espace des visages. Lorsque de nouveaux visages sont expérimentés, les visages propres sont mis à jour ou recalculés.**
   Cette étape est décomposée comme suit :
   a. Calcul du visage moyen : Les pixels constituant l'image du visage moyen résultent de la moyenne des pixels correspondants des différentes images de l'ensemble d'apprentissage. Ci-dessous est représenté le visage moyen dans le cas ou l'ensemble d'apprentissage contient des images normalisées.



Le visage moyen joue un rôle de filtre passe-bas dans la mesure ou il tend à 'gommer' les variations existantes entre les visages. Ces variations étant dues aux 'caractéristiques spécifiques' et donnant lieu aux hautes fréquences dans l'image.

b. Lorsque le visage moyen a été calculé. On le retranche de chaque visage de l'ensemble d'apprentissage. Cela donne des visages appelés ici visages différentiels. Un exemple de visage différentiel est repris ci-dessous :

c. Les visages différentiels sont représentés alors sous forme vectorielle et chaque vecteur forme une colonne de la matrice qu'on appelle A. La démarche consiste ensuite à déterminer les vecteurs propres de la matrice L, résultant du produit de la matrice transposée de A et de la matrice A (Cfr. chap 3). La méthode utilisée pour calculer ces vecteurs propres est décrite dans [7] pp470 à 481. En multipliant chacun des vecteurs propres obtenus par la matrice A des visages différentiels, on obtient les visages propres (le nombre maximum des vecteurs propres et donc des visages propres est égal au nombre d'images de l'ensemble d'apprentissage). Quelques exemples de visages propres sont repris ci-dessous :

3. **Calculer dans l'espace M-dimensionnel des visages propres, la distribution de poids correspondant pour chaque individu connu en projetant leur visage dans l'espace des visages.**
   Chaque visages de l'ensemble d'apprentissage est désormais décrit par un vecteur dont les composantes sont les poids. Ci-dessous est repris un exemple de ce vecteur :

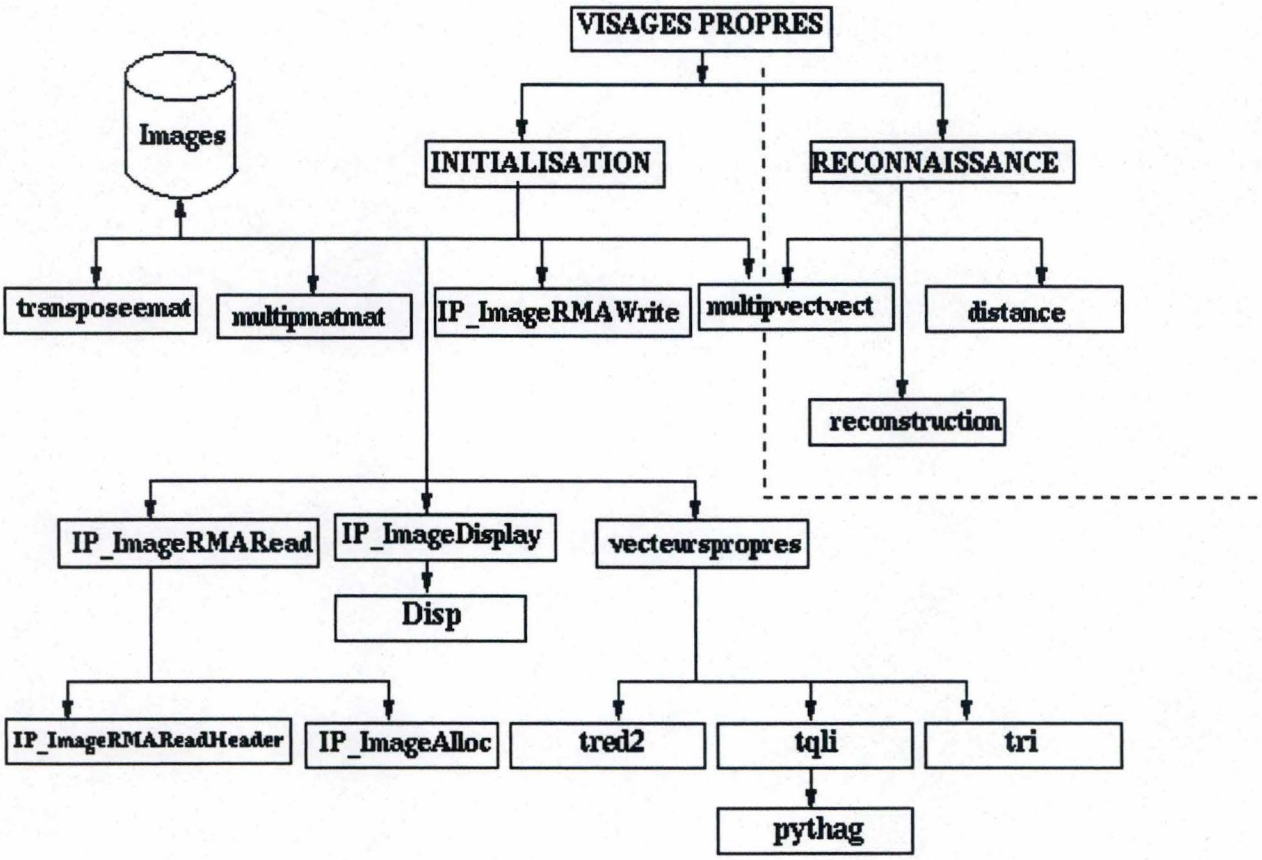
$$\Omega_{\texttt{ayhan2f2.rma}} = [-0.6272\ -0.2943\ 0.7211]$$

Le système est maintenant initialise. les étapes suivantes sont utilisées pour reconnaître un 'nouveau' visage:

4. **Calculer un ensemble de poids, à partir d'une image d'entrée et des M visages propres en projetant l'image d'entrée sur chaque visage propre.**
   Cette étape est similaire à la dernière étape de l'initialisation
5. **Déterminer si l'image est un visage propre en observant si l'image est suffisamment proche dans l'espace des visages.**
6. **S'il s'agit d'un visage, déterminer si oui ou non c'est une personne connue.**
   La méthode utilisée pour cette détermination est le calcul de la distance euclidienne. Si le visage candidat appartient à l'ensemble d'apprentissage, cette distance doit être nulle pour reconnaître le candidat. Dans le cas contraire, la distance doit être inférieure à un certain seuil qu'il faut fixer. Il n'est pas possible de déterminer a priori ce seuil. C'est donc en testant les candidats qu'il est possible de fixer un seuil approprié.
7. **(Optionnel) Adapter les visages propres et/ou poids.**

4.3 Description et commentaires du programme réalisé

Ci-dessous sont représentées les fonctions principales du programme VISAGES PROPRES.



Le programme contient deux phases qui sont:

- L'INITIALISATION
- La RECONNAISSANCE

Seront définies pour chacune des fonctions du programme, la structure, le but, les arguments, la ou les valeurs retournée(s) de chacune de ces fonctions et les fonctions qu'elles utilisent. Le code du programme et des fonctions utilisées figurent en annexe A.

Avant d'aborder les fonctions plus en détail, quelques explications sont données ci-dessous sur la définition d'un type spécifique qui a été utilisé et sur l'allocation de mémoire.

4.3.1 Variables et types

Outre les types classiques tels que les types entiers, réel, caractère, ..., un autre type, évoqué précédemment, a été utilisé. Il s'agit du type IP_image. La définition de ce type est la suivante:

```
typedef    unsigned char      u_c ;   Redéfinition de type existant
typedef    long               int4;   idem

typedef struct {
        void   (*id)() ;      Identificateur: représente les attributs de l'image
        int4    type ;        Type des données de l'image
        int4    x, y ;        Offset de l'image
        u_c     **data ;      Valeurs d'intensité de l'image
        char    *name ;       Nom de l'image
} IP_image;
```

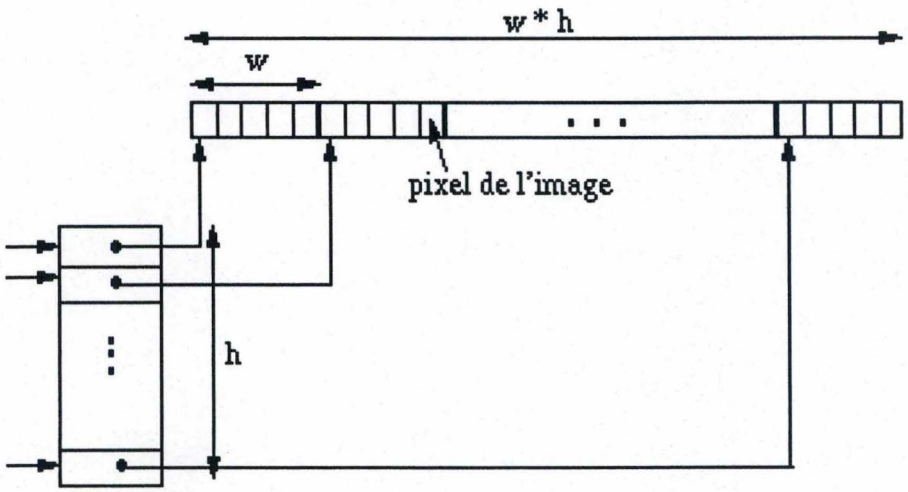
Le premier composant de la structure IP_image est un pointeur vers une fonction qui représente les attributs de l'image. La deuxième composante est le type des données utilisées pour représenter les valeurs d'intensité de l'image. L'Offset de l'image est donné par les coordonnées du pixel situé dans le coin supérieur gauche de l'image. Les valeurs d'intensité lumineuse de l'image sont représentées par un tableau bidimmensionnel.
La dernière composante de la structure est le nom de l'image.

Afin de stocker les images en mémoire, un tableau bidimensionnel dont les éléments sont de type IP_image sera utilisé. Nous avons par exemple :

****tabimages**	est utlisé pour mémoriser les images de l'ensemble d'apprentissage
****tabimgprop**	est utilisé pour mémoriser les visages propres générés

4.3.2 Allocation de mémoire

Des pointeurs sont utilisés pour représenter les tableaux bidimensionnels. Un tableau unidimensionnel de pointeurs est d'abord défini. Ces pointeurs pointent ensuite vers un autre tableau unidimensionnel dont les éléments sont les pixels d'une image par exemple.



4.3.3 INITIALISATION

But : Cette fonction a pour but de générer les visages propres à partir de l'ensemble d'apprentissage. Afin de tester la méthode, tout ou partie des visages propres sera retenu pour constituer la base de l'espace des visages. Dès que les visages propres ont été générés, les visages constituant l'ensemble d'apprentissage (notre base de donnée en quelque sorte) sont projetés dans l'espace des visages. Le résultat de cette projection est un vecteur pour chaque visage dont les éléments représentent les poids de chaque visage propres dans la représentation du visage en question

Arguments : La fonction initialisation a comme argument les données d'entrée, à savoir le nombre de visages constituant la base d'apprentissage, la localisation des fichiers d'images et les dimensions de ces images (hauteur, largeur).

Valeur retournée : Un pointeur vers le tableau de visages propres.

Fonctions utilisées : La fonction INITIALISATION utilise les fonctions suivantes :

- IP_ImageRMARead
- IP_ImageRMAWrite
- IP_ImageDisplay
- transposeemat
- multipmatmat
- vecteurspropres
- multmatvect
- multipvectvect

a) **La fonction IP_ImageRMARead :**

Structure: IP_image *IP_ImageRMARead(char *file)

But : Cette fonction lit un fichier de type RMA et retourne une structure IP_image allouée.

Arguments : le fichier RMA devant être lu et dans lequel sont extraites les donnée concernant les dimensions (largeur et hauteur) de l'image, le type et les valeurs

d'intensité des pixels de l'image.

Valeur retournée : Un pointer vers une structure IP_image 'remplie' ou 0 si la lecture et /ou l'allocation de mémoire ont échoué. Utilise 'errno' pour le débogage.

Fonctions utilisées : La fonction IP_ImageRMARead utilise les fonctions IP_ImageReadHeader et IP_ImageAlloc.

La fonction IP_ImageRMARead Header:

Structure : int IP_ImageRMAReadHeader(char *file, int4 *type, int4 *x, int4 *y, int4 *w, int4 *h)

But: Cette fonction lit le header du fichier de l'image et retourne les attributs de l'image.

Arguments: Le fichier RMA à lire.Le type utilisée pour les valeurs d'intensité des pixels, la largeur, la hauteur et l'Offset de l'image.

Valeur retournée: -1 en cas de problème. (utiliser errno pour déboguer).
0 si tout s'est bien passé.

La fonction IP_ImageAlloc:

Structure : IP_image *IP_ImageAlloc(int4 type, int4 w, int4 h, char *name)

But: Cette fonction alloue une structure IP_image.

Arguments: Le type utilisée pour les valeurs d'intensité des pixels, la largeur et la hauteur et le nom de l'image.

Valeur retournée: Un pointeur vers la structure IP_image
0 si le type, la largeur et la hauteur sont invalides.
0 s'il ya des problèmes de mémoire (vérifier errno dans ce cas-là).

b) **La fonction IP_ImageRMAWrite** :

Structure: int IP_ImageRMAWrite(IP_image *img, char *file)

But : La fonction IP_ImageRMAWrite écrit les données d'une image dans un fichier de type RMA.

Arguments : La structure IP_image devant être sauvée et le nom du fichier dnas lequel elle doit être sauvée.

Valeur retournée : Une valeur entière dupliquant la valeur 'errno' (qui indique les erreurs dans le système UNIX) pour indiquer des problèmes d'accès de fichier ou –1 si img a de mauvais attributs (type, largeur ou hauteur).

c) **La fonction IP_ImageDisplay** :

Structure : void IP_ImageDisplay(char *file) ;

But : La fonction IP_ImageDisplay affiche une image RMA à l'écran.

Arguments : Le nom du fichier contenant l'image à afficher.

Valeur retournée : Nihil.

d) **La fonction transposeemat** :

Structure: float **transposeemat(float **a, int n, int m)

But: La fonction transposeemat a pour but de transposer une matrice a de dimension m x n.

Argument s: Le pointeur vers la matrice à transposer, le nombre de rangées et de colonnes de cette matrice.

Valeur retournée : Un pointeur vers la matrice transposée.

e) **La fonction multipmatmat :**

Structure: float **multipmatmat(float **f, float **g, int m, int n, int p)

But: La fonction multipmatmat effectue la multiplication d'une matrice f de dimensions m x n avec une matrice g de dimension n x p.

Argument s: Les pointeurs vers les matrices f et g à multiplier, le nombre de rangées et de colonnes de ces matrices.

Valeur retournée : Un pointeur vers la matrice résultante.

f) **La fonction vecteurspropres :**

Structure: float **vecteurspropres(float **a, int n)

But: La fonction vecteurpropres calcule les valeurs propres et vecteurs propres associés d'une matrice a et les retourne dans l'ordre décroissant des valeurs propres.

Arguments : La matrice dont il faut calculer les vecteurs propres et le nombre de vecteurs propres souhaités.

Valeur retournée : Un pointeur vers une matrice contenant les vecteurs propres.

Fonctions utilisées : tred2, tqli, tri.

La fonction tred2 ([7])

Structure : void tred2(float **a, int n, float *d, float *e) ;

But : La fonction transforme une matrice a, symétrique, de dimensions n x n en une matrice tridiagonale par des transformations orthogonales successives. Cette transformation s'appuye sur la méthode de réduction 'Householder'.

Arguments : Un pointeur vers la matrice a dont on souhaite la forme tridiagonale, la taille n de cette matrice et deux pointeurs vers les vecteurs d et e retournant les éléments de la diagonale principale et des éléments situés sous les éléments de la diagonale principale.

Valeur retournée : Les valeurs sont retournées via les arguments par le biais des pointeurs.

NB : Cette fonction tred2 est exécutée avant la fonction tqli car cette dernière utlise les arguments de tred2.

La fonction tqli ([7])

structure : void tqli(float *d, float *e, int n, float **z) ;

But : La fonction tqli détermine les valeurs propres et vecteurs propres d'une matrice réelle, symétrique, tridiagonale ou d'une matrice réelle, symétrique précédemment traitée par la fonction tred2. Cette fonction est basée sur l'algorithme QL avec 'shifts' implicites.

Arguments : Les arguments d'entrée de la fonction tqli sont les pointeurs vers les vecteurs d et e résultant de l'exécution de la fonction tred2. En sortie, les éléments du vecteur d sont les valeurs propres recherchées. Le pointeur vers la matrice z est le même que celui vers la matrice a, précédemment traitée par la fonction tred2. Via ce pointeur les vecteurs propres sont retournés. La kième colonne de Z étant le vecteur propre normalisé correspondant à la valeur propre d[k].

Valeur retournée : Les valeurs sont ici également retournées via les arguments par le biais des pointeurs.

La fonction tri

Structure : float **tri(float *x, float **r, int n) ;

But : Cette fonction effectue un tri par ordre décroissant des valeurs d'un tableau bidimensionnel (matrice dont les colonnes sont des vecteurs).

Arguments : Un pointeur vers un tableau de données et le nombre d'éléments de ce tableau, un pointeur vers le tableau de pointeurs pointant vers les vecteurs à trier.

Valeur retournée : Un pointeur vers le tableau de pointeurs dont les éléments pointés ont été triés par ordre décroissant.

g) **La fonction multmatvect :**

Structure : float *multmatvect(float **a,float *b,int m,int n) ;

But: La fonction multmatvect effectue la multiplication d'une matrice a de dimensions m x n par un vecteur b de dimensions n x 1.

Arguments: Un pointeur vers la matrice a, un pointeur vers le vecteur b et les dimensions de la matrice et du vecteur

Valeur retournée: Un pointeur vers le vecteur résultant.

h) **La fonction multipvectvect :**

Structure : float multipvectvect(float *a,float *b,int n) ;

But: La fonction multipvectvect effectue le produit de deux vecteurs a et b de dimension n.

Arguments : Un pointeur vers les vecteurs a et b et la dimension n de ces vecteurs.

Valeur retournée : Le résultat du produit des deux vecteurs.

4.3.4 RECONNAISSANCE

But : La phase RECONNAISSANCE projète d'abord un visage candidat soumis au test de la reconnaissance dans l'espace des visages propres. Le résultat de cette projection est un

vecteur. On détermine ensuite les distances entre ce vecteur et l'ensemble des vecteurs représentant les visages de la base de données. Si une des distances est inférieure à un seuil prédéterminé, alors le candidat est reconnu sinon le candidat est déclaré inconnu. Si le candidat est reconnu, le résultat de la reconstruction du visage candidat à partir des visages propres retenus est affiché.

Fonctions utilisées : La phase de reconnaissance utilise les fonctions suivantes :

- multipvectvect
- distance
- reconstruction

a) **La fonction multipvectvect**: (Cfr ci-avant)

b) **La fonction distance**

Structure : double distance(float *a,float *b,int n) ;

But : La fonction distance calcule la distance euclidienne entre deux points d'un espace, représentés par des vecteurs.

Arguments : La fonction distance prend en argument les pointeurs vers les deux vecteurs et la dimension de ces vecteurs.

Valeur retournée : La valeur retournée est la distance entre ces points.

c) **La fonction reconstruction**

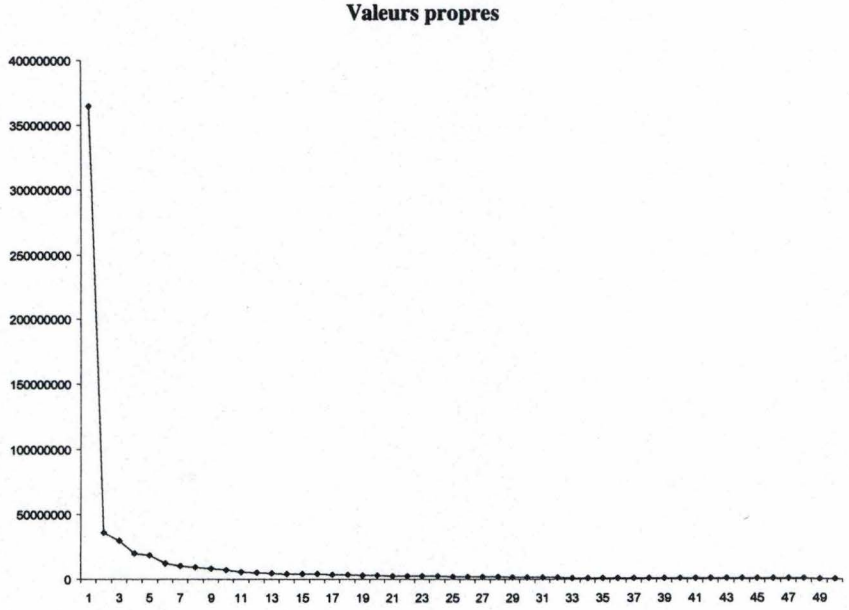
Structure : float *reconstruction(float **a,float *b,int m,int n);

But : La fonction reconstruction reconstruit et affiche le visage reconnu à partir de l'ensemble des visages propres retenus et du vecteur résultant de la projection du visage candidat dans l'espace des visages.

Arguments : La fonction prend en argument une matrice a dont les rangées sont les vecteurs représentant les visages propres, un vecteur b résultant de la projection du visage candidat dans l'espace des visages et les dimensions de la matrice et du vecteur.

4.4 Principaux résultats obtenus

Si dans la phase d'initialisation, le nombre maximum de visages propres est retenu, on observe les résultats suivants. Lorsque le visage du candidat fait partie de l'ensemble d'apprentissage, le taux de reconnaissance est de 100 %. Lorsque le visage du candidat ne fait pas partie de l'ensemble d'apprentissage mais qu'une autre image de son visage en fait partie les résultats restent encore très bons (86 %). Les performances du système diminuent fortement lorsqu' on diminue le nombre de vecteurs propres pris en compte. Ainsi le nombre de vecteurs propres à maintenir pour observer des résultats entre 85 et 90 % d'acceptation est de 12 visages propres pour un ensemble d'apprentissage de 50 personnes donc et donc pour un nombre maximal de 50 visages propres. Cela apparaît plus clairement dans le graphique ci-dessous qui représente la valeur de chaque valeur propre. On constate effectivement que la décroissance des valeurs propres est exponentielle (Cfr Chapitre 2
paragraphe 2.3.4).



Il apparaît au vu de ce graphique que presque les 40 dernières valeurs propres (et les vecteurs propres correspondants) peuvent être négligées.

CONCLUSIONS

CONCLUSIONS

L'objectif du travail visait à implémenter une méthode de reconnaissance de visages en vue frontale. La question était alors de déterminer quelle méthode il fallait choisir et sur base de quels critères. Malgré les recherches intensives dans le domaine de la reconnaissance depuis des années, aucune méthode ne semble vraiment avoir fait l'unanimité, chacune possédant bien sur ses avantages et ses inconvénients. On peut classer les méthodes en deux grandes catégories : celles qui ont une approche considérant le visage comme une entité globale et celles qui considèrent le visage comme une collection de caractéristiques particulières. Beaucoup d'entre-elles s'appuyent notamment sur une information préalable du visage relative aux caractéristiques biométriques de celui-ci. Les auteurs de ces méthodes n'arrivent cependant pas à s'accorder sur le nombre ou le type de caractéristiques à choisir.

La méthode des visages propres basée sur la décomposition en composantes principales possède l'avantage par rapport à ces méthodes de ne nécessiter aucune information préalable si ce n'est l'information présente dans les images elles-mêmes. La méthode a l'avantage également d'être simple, d'être robuste lorsque les images sont relativement normalisées (même échelle, même orientation, etc.). Le choix a donc été porte sur cette méthode pour effectuer ses premiers pas dans le domaine de la reconnaissance de visage.

Si dans l'ensemble les résultats sont encourageants, il y a lieu de tempérer son enthousiasme par le fait que cette méthode apparaît fragile lorsque les images à tester 's'écartent' des images de l'ensemble d'apprentissage (taille différente, orientation différente, position, arrière-plan, etc.). Une autre faiblesse réside également dans la détermination des seuils à fixer pour la reconnaissance qui se fait par tâtonnements successifs.

Sur le travail proprement dit, d'autres pistes sont à explorer, comme l'information de la couleur par exemple, comme la localisation d'un visage dans une

image par exploration des sous-image de l'image, comme la classification, l'analyse des expressions, etc.

En guise de conclusion, on peut dire qu'une méthode seule ne suffit pas à mettre au point un système automatique de reconnaissance sans faille. La combinaison de méthodes (de reconnaissance de visage ou autre comme la reconnaissance vocale) semble être une solution.

ANNEXE A

ANNEXE A

Code du programme VISAGES PROPRES

```
#include "IP.H"
#include <sys/types.h>
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>
#include "nrutil.h"

main()
{
        char  choix;

        void initialisation(void);

        /* Presentation */

        printf("\n\nBienvenue dans le programme VISAGES PROPRES");
        printf("\n----------------------------------------");
        printf("\n\n le système doit etre initialisé avant de procéder a la
                reconnaissance");
        printf( "\n\n Pour continuer entrez 1. Pour arreter entrez 2");
        printf("\n\n 1. INITIALISATION");
        printf("\n\n 2. SORTIE\n\n");

        switch(choix = getchar()) {
        case '1':
                printf("\n\n INITIALISATION\n\n ");
                initialisation();

                break;

        case '2':
                printf("\n\nMerci et à bientôt\n\n");
                break;

        }
```

```
/* Fonction INITIALISATION */

void initialisation(void)
{
                        /* Déclarations des variables*/

IP_image*img,*imgprop,**tabimages,**tabimgprop,**tabimagesdiff,*visagemoyen,
            **mem, *imgdiff, *visagerec,*candidat;
double    *eps, eps2;
float     **A, **Af, **At, **B, *V, **VP, **VPt, *PHI, *u, *imgmat,**matimages;
float     *phi,*phif, *phir ,**phimatrice, **phirmatrice,*img1,**vismoy, *img2,
          *vismoyvect,*G, *H, **img3, **Z, **U, ***diff, s;
int       M, M1, i, j, res, w, h, k, l, compteur, q;
int4      x, y, width, height, type;
char      *a,*b,*c,*d,*nom;
DIR       *dirp;
struct dirent   *dp;
size_t    length;

        float **transposeemat(float **a,int n, int m);
        float **multipmatmat(float **f,float **g,int m,int n,int p);
        float **vecteurspropres(float **a, int n);
        float *multmatvect(float **a,float *b,int m,int n);
        float **vecteurspropres(float **a, int n);
        float multipvectvect(float *a,float *b,int n);
        double distance(float *a,float *b,int n);
        int min(double *x, int n);
        float *reconstruction(float **a,float *b,int m,int n);
        float *norme(float *a, int n);


/*************************************************************************/

        /* Données d'entrée */

        printf("Nombres d'images en entree? ");
        scanf("%d",&M);

        printf("\n\n Nombre de vecteurs propres retenus ? ");
        scanf("%d", &M1);

        w = 100;
        h = 130;


/*************************************************************************/
```

```
/* Allocation de memoire */

    img = (IP_image *)calloc(M, sizeof(IP_image));
    if( img != NULL )
            printf( "Allocated %d  images\n", M );
    else {
            printf( "Can't allocate memory\n" );
            free(img );
     }
    tabimages = (IP_image **)malloc(M*sizeof(IP_image *));
    if( tabimages == NULL ){
            printf( "Insufficient memory available\n" );
            free(tabimages );
    }

    for(i=0;i<M;i++) tabimages[i]=&img[i * sizeof(IP_image)];

/*************************************************************************/

    imgdiff = (IP_image *)calloc(M, sizeof(IP_image));
    if( imgdiff != NULL )
    printf( "Allocated %d  imagesdiff\n", M );
    else {
            printf( "Can't allocate memory\n" );
            free(imgdiff );
    }
    tabimagesdiff = (IP_image **)malloc(M*sizeof(IP_image *));
    if( tabimagesdiff == NULL ){
            printf( "Insufficient memory available\n" );
            free(tabimagesdiff );
    }

    for(i=0;i<M;i++) tabimagesdiff[i]=&imgdiff[i * sizeof(IP_image)];

/*************************************************************************/

    imgprop = (IP_image *)calloc(M, sizeof(IP_image));
    if( imgprop != NULL )
    printf( "Allocated %d  images propres\n", M );
    else {
            printf( "Can't allocate memory\n" );
            free(imgprop);
    }
    tabimgprop = (IP_image **)malloc(M*sizeof(IP_image *));
    if( tabimgprop == NULL ){
            printf( "Insufficient memory available\n" );
            free(tabimgprop );
    }
```

```
        else
                printf( "Memory space allocated for visages propres\n\n" );

        for(i=0;i<M;i++) tabimgprop[i]=&imgprop[i * sizeof(IP_image)];

/*************************************************************************/

        visagemoyen = IP_ImageAlloc(U_CHAR, w,h , "visage_moyen");
        if (visagemoyen == 0){
         if (errno) perror("Allocate Image MOYENNE"); exit(0);
                 printf( "Can't allocate memory\n" );
                free(visagemoyen);
        }
        else
                printf( "Memory space allocated for visagemoyen\n\n" );

        visagerec = IP_ImageAlloc(U_CHAR, w,h , "visage_reconstruit");
        if (visagerec == 0){
                if (errno) perror("Allocate visage reconstruit"); exit(0);
                printf( "Can't allocate memory\n" );
                free(visagerec);
        }
        else
                printf( "Memory space allocated for visagerec\n\n" );


        candidat = IP_ImageAlloc(U_CHAR, w,h , "visage_candidat");
        if (candidat == 0){
                if (errno) perror("Allocate visage reconstruit"); exit(0);
                printf( "Can't allocate memory\n" );
                free(candidat);
        }
        else
                printf( "Memory space allocated for candidat\n\n" );

/*************************************************************************/

vismoy = matrix(1,h,1,w);              /* matrice du visage moyen */
vismoyvect = vector(1,(w*h));          /* vecteur du visage moyen */
diff = f3tensor(1,M,1,h,1,w);
A = matrix(1,M,1,(w*h));               /* matrice des visages différentiels */
Af = matrix(1,M,1,(w*h));
At = matrix(1,(w*h),1,M);              /* matrice transposée de la matrice A */
B = matrix(1,M,1,M);                   /* matrice B = At.A */
PHI = vector(1,(w*h));
VP = matrix(1,M,1,M);                  /* matrice des vecteurs propres */
VPt = matrix(1,M,1,M);                 /* matrice transposée de la matrice VP */
U = matrix(1,M,1,(w*h));               /* matrice des visages propres */
```

```
V = vector(1,M);                    /* vecteur des composantes du candidat test */
G = vector(1,(w*h));                /* vecteur auxiliaire */
H = vector(1,(w*h));                /* vecteur auxiliaire */
Z = matrix(1,M,1,M);          /* matrice des composantes des visages de l'ensemble
                              de départ */
eps = dvector(1,M);                 /* vecteurs des distances */

phimatrice = matrix(1,h,1,w);    /* matrice du visage différentiel du candidat test */
phi = vector(1,(w*h));           /* vecteur du visage différentiel du candidat test */
phirmatrice = matrix(1,h,1,w);
phir = vector(1,(w*h));
phif = vector(1,(w*h));     /* reconstruction du visage différentiel du candidat test */


/*****************************************************************************/


        /* Ouverture, lecture et affichage des fichiers images */

dirp = opendir("/dos/user/didier/face/Programme/ImagesNormalisees");
i=0;
while ((dp = readdir(dirp)) && (i != M)){
        length = strlen(dp->d_name);
        if  ((dp->d_name[length-4] == '.') && (dp->d_name[length-3]=='r') &&
                (dp->d_name[length-2] =='m') && (dp->d_name[length-1] =='a')){
                tabimages[i]=IP_ImageRMARead(dp->d_name);
                if (tabimages[i] ==0){ perror("Read"); exit(0);}
                tabimagesdiff[i]=IP_ImageRMARead(dp->d_name);
                if (tabimagesdiff[i] ==0){perror("Read");exit(0);}
                 tabimgprop[i]=IP_ImageRMARead(dp->d_name);
                if (tabimgprop[i]==0){ perror("Read ");exit(0);}

                    /* affichage des images */
                 a = (char*)strcat(dp->d_name,"&");
                IP_ImageDisplay(a);
                 i++;
                 if (i== M){break;}
          }
}
closedir(dirp);

/*****************************************************************************/
```

```
/* Calcul et sauvegarde du visage moyen */


for (y=1; y<=h; y++)
        for (x=1; x<=w; x++)
           vismoy[y][x] = 0;


for (i=0;i<M;i++)
  for (y=0; y<h; y++)
        for (x=0; x<w; x++) {
           vismoy[y+1][x+1]  = ((tabimages[i]->data[y][x])/M + (vismoy[y+1][x+1]));
           visagemoyen->data[y][x] = abs((tabimages[i]->data[y][x])/M +
                                          (visagemoyen->data[y][x]));
        }


/* visage moyen sous forme de vecteur */


for (l=1;l<=h;l++)
         for (k=1;k<=w;k++)
              vismoyvect[(k+((l-1)*w))] = vismoy[l][k];


/* Visualisation de l'image moyenne */

res = IP_ImageRMAWrite(visagemoyen,"image_moyenne");
if (res) { perror("Write image moyenne"); exit(0); }
IP_ImageDisplay("image_moyenne&");



/*************************************************************************/

/* Calcul et sauvegarde des images differentielles */


for (i=1;i<= M;i++)
  for (y=1; y<=h; y++)
        for (x=0; x<w; x++)
              diff[i][y][x] = 0;

for (i=0;i<M;i++){
      for (y=0; y<h; y++)
         for (x=0; x<w; x++){
              tabimagesdiff[i]->data[y][x] = abs(tabimages[i]->data[y][x] -
                                                 visagemoyen->data[y][x]);
```

```
                diff[i+1][y+1][x+1] =  (tabimages[i]->data[y][x] - vismoy[y+1][x+1]);
        }
    }
```

/* Visualisation d'une image differentielle */

```
b = (char*)strcat(tabimagesdiff[1]->name,"_diff");
res = IP_ImageRMAWrite(tabimagesdiff[1],b);
if (res) { perror("Write visage diff"); exit(0); }
c =(char*)strcat(b,"&");
IP_ImageDisplay(b);
```

```
/*************************************************************************/
```

/* Formation de la matrice des images */

```
for (i=1;i<=M;i++)
      for (j=1;j<=(w*h);j++)
         A[i][j] = 0;

for (i=1;i<=M;i++){
      for (l=1;l<=h;l++)
         for (k=1;k<=w;k++)
               PHI[(k+((l-1)*w))] = diff[i][l][k];
      for (j=1;j<=(w*h);j++)
         A[i][j] = PHI[j];
}
```

```
/*************************************************************************/
```

/* Calcul des vecteurs propres de la matrice B = At.A */

```
At = transposeemat(A,(w*h),M);
B  = multipmatmat(A,At,M,(w*h),M);

VP = vecteurspropres(B,M);


printf("\nVecteurs propres tries\n");
printf("******************\n\n");
for (i=1;i<=M;i++){
      printf("\n%11s %d\n\n", "vecteur propre nr ",i );
      for (j=1;j<=M;j++)
         printf("%12.6f\n ", VP[i][j]);
}
```

```
/* Calcul des visages propres */

   for (i=1;i<=M;i++){
         u = multmatvect(At,VP[i],(w*h),M);
    for (j=1;j<=(w*h);j++)
         U[i][j] = u[j];
   }


  for (i=1;i<=M;i++)
       U[i] = norme(U[i],(w*h));


/*********************************************************************/

  /* Affichage des M1 visages propres retenus */

for (i=0;i<M;i++){
       for (y=0; y<h; y++)
          for (x=0; x<w; x++)
               tabimgprop[i]->data[y][x]=(abs) U[i+1][(x+1+(w*y))]);
  }


printf("\n\nVisages propres\n");
 for (i=0;i<M1; i++){
       sprintf(a,"%s%d","image_propre",i);
       res = IP_ImageRMAWrite(tabimgprop[i],a);
       if (res) { perror("Write Visage propre"); exit(0); }
       b =  (char*)strcat(a,"&");
       IP_ImageDisplay(b);
   }

/*********************************************************************/

       /* Composantes de chaque image dans le nouvel espace */


  for (i=1;i<=M;i++)
       for (k=1;k<=M;k++){
          for (j=1;j<=(w*h);j++) {H[j] = A[i][j];G[j] = U[k][j];}
          Z[i][k] = multipvectvect(G,H,(w*h));
       }


/*********************************************************************/
```

```
/* Affichage des composantes */

printf("\nComposantes dans l'espace des visages\n\n");

for (i=1;i<=M;i++){
    printf("\n\nComposantes du visage %d\n\n",i);
    for (j=1;j<=M1;j++)
      printf("\nZ[%d][%d] = %.4f",i,j,Z[i][j]);
}

/*************************************************************************/

    /*Phase RECONNAISSANCE*/

    printf("Visage candidat\n");
    printf("*****************\n\n");


    dirp = opendir("/dos/user/didier/face/Programme/ImagesNormalisees");
    while (((dp = readdir(dirp))!=NULL)){
        length = strlen(dp->d_name);
        if  ((dp->d_name[length-4] == '.') && (dp->d_name[length-3] == 'r')
            &&  (dp->d_name[length-2] == 'm') &&
                (dp->d_name[length-1] == 'a') )
                    printf("\n%s\n",dp->d_name);
      }
    closedir(dirp);

    printf("\nChoisissez un nom dans la liste ci-dessus (nom + ext)\n\n");
    scanf("%s", &nom);
    printf("\nLe nom choisi est\n");
    printf("%s\n",&nom);
    a=&nom;

candidat = IP_ImageRMARead(a);
if (candidat == 0) { perror("Read candidat"); exit(0); }

/* Affichage du visage du candidat en  background */
res = IP_ImageRMAWrite(candidat,"visage_candidat");
b = (char *)strcat(a,"&");
IP_ImageDisplay("visage_candidat&");
```

```
/* visage candidat - visage moyen     */

for (y=0; y<h; y++)
      for (x=0; x<w; x++)
      phimatrice[y+1][x+1] = (candidat->data[y][x]- vismoy[y+1][x+1]) ;

for (y=1;y<=h;y++)
      for (x=1;x<=w;x++)
         phi[(x+((y-1)*w))] = phimatrice[y][x];


/* Composante du visage candidat dans l'espace des visages propres */

for (k=1;k<=M;k++){
      for (j=1;j<=(w*h);j++){ G[j] = U[k][j];}
      V[k] = multipvectvect(G,phi,(w*h));
  }

/*********************************************************************/

/*Distance entre le visage du candidat et les visages de l'ensemble de départ */

   for (i=1; i<=M;i++)
               eps[i] = distance(V,Z[i],M1);

   for (i=1; i<=M;i++)
               printf("\n\n%.5f\n",eps[i]);

   q = min(eps,M);

   printf("\nResultat\n");
   printf("\n****************\n\n");
   printf("le visage le mieux place est le nr %d appele %s",(q), tabimages[q-1]->name);
   b = (char*)strcat(tabimages[q-1]->name," &");
   IP_ImageDisplay(tabimages[q-1]->name);
   printf("\nsa distance par rapport au candidat est %.5f\n", eps[q]);


/*********************************************************************/

/* Reconstruction du visage*/


   phif =  reconstruction(U,V,M1,(w*h));

   eps2 = distance(phi,phif,(w*h));
```

```
printf("%f\n",eps2);



for  (j=1;j<=(w*h);j++)
   phir[j] = (phif[j] + vismoyvect[j]);

for (y=1;y<=h;y++)
      for (x=1;x<=w;x++)
       phirmatrice[y][x] =  phir[(x+((y-1)*w))];

for (y=1; y<=h; y++)
       for (x=1; x<=w; x++)
                      visagerec->data[y-1][x-1] = phirmatrice[y][x];



/* Visualisation de l'image reconstruite */

res = IP_ImageRMAWrite(visagerec,"image_rec");
if (res) { perror("Write image rec"); exit(0); }
/*d = (char*)strcat("image_rec","& ");*/
IP_ImageDisplay("image_rec");


}
```

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[1] ABDI, H., VALENTIN,D., Modeles Neuronnaux, Connexionistes et Numeriques pour la Memoire des Visages, it Psychologie Francaise, 39(4), p357-392.

[2] ABDI, H., VALENTIN,D., O'TOOLE, A.J., Principal component and neural networks analyses of faces images: Explorations into the nature of information available for classifying faces by sex, In Progress in mathematical psychology, New-York: Elsevier

[3] BRUCE, V., BURTON, M., Processing Images of Faces, Ablex Publishing Corporation, New Jersey, 1992, 255p.

[4] BRUNELLI, R., POGGIO, T., Features versus templates, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, Nr 10, pp1042-1052, Octobre 1993.

[5] CHELLAPPA, R., WILSON, C. L., SIROHEY, S., Human and machine recognition of faces: A survey, Proceedings of the IEEE, Vol. 83, Nr 5, pp705-740, Mai 1995.

[6] CRAW, E., ELLIS, H., LISHMAN, J.R., Automatic extraction of face-features, Pattern Recognition Letters 5 pp 183-187, 1987.

[7] H PRESS, W., T. VETTERLING, W.,A. TEUKOLSKY, S., P. FLANNERY, B., Numerical Recipes in C, USA, Cambridge University Press, 1992, 994 p.

[8] KAMEL,M.S., SHEN, H.C., WONG, A.K.C., CAMPEANU,R.I.,System for the recognition of human faces, IBM SYSTEMS Journal, Vol. 32, Nr 2, pp 307-320.

[9] KAYA, A., KOBAYASHI, K., A basic study on human face recognition, Frontiers of Pattern recognition, Satori Wanabe, pp 265-288, New-York, 1972.

[10] PRATT, W. K., Digital Image Processing, A Wiley-Interscience Publication, 1978, 750p.

[11] ROMDHANI, S., Face Recognition using Principal Components Analysis, 1996, 58p.

[12] ROSENFELD, A., KALK, A.C., Digital Picture Processing, Volume 1, Academic Press, Inc, 1982, 435p.

[13] SAMAL, A., IYENGAR, P.A., Automatic recognition and analysis of human faces and facial expressions, Pattern Recognition, Vol. 25, Nr 1, pp65-72,1992.

[14] TURK M. A., PENTLAND A. P., Face recognition using eigenfaces, Proceedings International Conference on Pattern Recognition, pp 586-591, 1991.