Revue des algorithmes PCA, LDA et EBGM utilisés en reconnaissance 2D du visage pour la biométrie

Nicolas MORIZET*, **Thomas EA, Florence ROSSANT, Frédéric AMIEL, Amara AMARA**
Institut Supérieur d'Électronique de Paris (ISEP), Département d'Électronique
21, rue d'Assas 75270 Paris Cedex 06
{prenom}.{nom}@isep.fr

Résumé : Ce tutoriel expose trois algorithmes classiques utilisés en reconnaissance 2D du visage, travaillant à partir d'une unique photo de visage. Les deux premiers algorithmes, PCA et LDA, utilisent des méthodes statistiques classiques, tandis que le troisième algorithme, l'EBGM, utilise une approche différente en s'appuyant sur des points caractéristiques du visage et en utilisant des ondelettes 2D de Gabor. Cette étude s'inscrit dans le cadre de recherches en biométrie multimodale, consistant à fusionner les signatures biométriques de l'iris et du visage, en vue d'améliorer le taux de reconnaissance et de contrer les techniques de fraude des systèmes biométriques unimodaux.
Mots-clés : Biométrie, reconnaissance 2D du visage, PCA, Eigenfaces, LDA, Fisherfaces, EBGM, ondelettes, CMC, imagerie.

1 INTRODUCTION

Depuis plusieurs années, des efforts importants sont fournis dans le domaine de la recherche en biométrie. Ce phénomène s'explique en partie par la présence d'un contexte international où les besoins en sécurité deviennent de plus en plus importants et où les enjeux économiques sont colossaux.
Qu'est-ce que la biométrie ? La biométrie est la science qui étudie à l'aide de mathématiques les variations biologiques à l'intérieur d'un groupe déterminé. Le terme de biométrie regroupe en fait ce que l'on appelle des *modalités biométriques*[1]. La reconnaissance par le visage est une méthode *non intrusive*[2] et constitue la voie la plus naturelle pour reconnaître un individu. Il existe des méthodes de reconnaissance 2D s'appuyant sur une photo du visage, des méthodes 3D exploitant l'information de profondeur ainsi que des méthodes mixtes, joignant la 2D à la 3D. On discerne les méthodes holistiques (agissant sur la globalité de l'image), des méthodes basées sur des modèles (nécessitant certains points caractéristiques du visage). On notera qu'il existe également d'autres méthodes d'apprentissage et de classification comme les réseaux de neurones (NN), les modèles cachés de Markov (HMM), etc. mais ces algorithmes ne feront pas l'objet de cette revue.

Nous commencerons par faire un bref rappel de mathématiques statistiques et d'algèbre linéaire [Smith, 2004]. Puis nous étudierons deux algorithmes globaux. Enfin, nous verrons un algorithme local utilisant des ondelettes 2D de Gabor.

2 PRÉREQUIS MATHÉMATIQUES

2.1 Statistiques

2.1.1 Moyenne

On considère un vecteur X représentant une distribution de n données : $X = [X_1, X_2, \ldots, X_i, \ldots, X_n]$.
La moyenne $\overline{X}$ de cette distribution s'écrit :

$$\overline{X} = \sum_{i=1}^{n} \frac{X_i}{n} \tag{1}$$

2.1.2 Écart-type

L'écart-type est une *mesure de la dispersion* d'un ensemble de données. D'un point de vue qualitatif, l'écart-type caractérise la largeur d'une distribution de données en mesurant la dispersion autour de la moyenne. La formule de l'écart-type σ est :

$$\sigma = \sqrt{\sum_{i=1}^{n} \frac{(X_i - \overline{X})^2}{n-1}} \tag{2}$$

Remarque : On se pose généralement la question « Pourquoi $n-1$? ». La raison pour laquelle on divise par $n-1$ au lieu de n est issue de l'interaction permanente entre les statistiques et les probabilités[3].

2.1.3 Variance

La variance est une autre mesure de la dispersion d'un ensemble de données. Il s'agit tout simplement du carré de l'écart-type.

*Auteur. Tél.: +33 1 49 54 52 92; fax: +33 1 49 54 52 51. Adresse e-mail: nicolas.morizet@isep.fr.

[1] elles servent à prouver notre identité : les empreintes digitales, la main, la voix, l'iris, la rétine, le visage, etc. sont autant de modalités biométriques.

[2] méthode qui n'atteint pas l'intimité de l'utilisateur, ce dernier est donc relativement coopératif lors du procédé.

[3] On trouvera une démonstration sur le Wikipédia, à l'adresse suivante `http://fr.wikipedia.org/wiki/` en entrant "écart-type" dans le moteur de recherche interne.

La formule de la variance σ^2 est donc :

$$\sigma^2 = \sum_{i=1}^{n} \frac{(X_i - \overline{X})^2}{n-1} \tag{3}$$

2.1.4 *Covariance*

La covariance est une extension de la notion de variance. Elle mesure la *variation simultanée* de deux variables et donne une matrice carrée.
La formule de covariance entre deux variables X et Y s'écrit donc :

$$cov(X,Y) = \sum_{i=1}^{n} \frac{(X_i - \overline{X})(Y_i - \overline{Y})}{n-1} \tag{4}$$

Si les quantités $(X_i - \overline{X})$ et $(Y_i - \overline{Y})$ varient dans le même sens (on dit alors que les variables X et Y *covarient*), la covariance sera positive. À l'inverse, si $(X_i - \overline{X})$ et $(Y_i - \overline{Y})$ varient dans des sens opposés, la covariance sera négative. Enfin, on voit bien d'après (4) que si la covariance est calculée entre une variable et elle-même (donc si $X = Y$), nous obtenons (3), donc la variance.

2.1.5 *Matrice de Covariance*

La matrice de covariance permet de mesurer la *variation simultanée* de n variables. Elle nous sera très utile puisque nous serons amenés à manipuler des vecteurs avec de très nombreuses composantes, lors des algorithmes 2D de reconnaissance du visage.
Pourquoi utiliser une matrice ? Si nous devons travailler dans un espace vectoriel de dimension[4] supérieure à 2, plusieurs mesures de covariance peuvent alors être calculées. Plus précisément, pour n dimensions, nous pouvons calculer $\frac{n!}{2\times(n-2)!}$ valeurs différentes de covariance (en excluant les termes de la diagonale principale). Une manière pratique d'obtenir toutes ces valeurs est de les calculer et de les mettre dans une matrice. Ainsi, pour n dimensions, la définition d'une matrice de covariance est :

$$\mathbf{C}^{n\times n} = (c_{i,j},\ c_{i,j} = cov(Dim_i, Dimj))$$

où $\begin{cases} \mathbf{C}^{n\times n} & \text{Matrice avec } n \text{ lignes et } n \text{ colonnes,} \\ Dim_x & \text{la } x^{\text{e}} \text{ dimension.} \end{cases}$

Par exemple, en dimension 3, utilisant les dimensions usuelles x,y,z, nous obtenons une matrice de covariance de taille (3×3) et dont les valeurs sont les suivantes :

$$\mathbf{C} = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix} \tag{5}$$

Remarque : le long de la diagonale, on voit que la valeur de la covariance est selon une dimension et elle-même. Ce sont les variances pour ces dimensions. L'autre chose à noter est que comme $cov(a,b) = cov(b,a)$, la matrice est symétrique par rapport à sa diagonale principale.

[4] Il convient de bien faire la différence entre la dimension n d'un vecteur V (qui correspond aux nombre d'éléments de V) et la dimension $Dim(\mathcal{E})$ d'un espace vectoriel $\mathcal{E}$ (qui correspond au nombre total de vecteurs constituant une base de $\mathcal{E}$). Chaque vecteur V_i d'une base de $\mathcal{E}$ peut alors être vu comme "support" d'une dimension $Dim_i(\mathcal{E})$.

2.2 Algèbre linéaire

Dans cette section, nous allons revoir les notions de vecteurs propres et de valeurs propres qui jouent un rôle essentiel dans les algorithmes globaux.

2.2.1 *Vecteurs propres et valeurs propres*

Comme nous le savons, nous pouvons multiplier deux matrices entre-elles, sous réserve de compatibilité de leurs dimensions respectives. Les vecteurs propres sont un cas particulier de cette opération matricielle.
Soit E un *vecteur propre* d'une *matrice carrée* M, et λ un réel, alors :

$$M \cdot E = \lambda \times E, \quad \lambda \in \mathrm{R} \tag{6}$$

où λ est la *valeur propre* associée au vecteur propre E. Plus la valeur propre est grande, plus l'information contenue dans le vecteur propre correspondant est importante.

D'après l'équation (6), on voit bien que les vecteurs propres d'une matrice carrée M sont les vecteurs qui ne subissent pas de changement de direction lorsqu'ils sont multipliés, *à gauche*, par M (figure 1). C'est pour cela que ces vecteurs propres sont aussi appelés les *directions principales* de la matrice M.

$$\underbrace{\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}}_{M} \times \underbrace{\begin{pmatrix} 1 \\ 3 \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} 11 \\ 5 \end{pmatrix}}_{X'}$$

$$\underbrace{\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix}}_{M} \times \underbrace{\begin{pmatrix} 3 \\ 2 \end{pmatrix}}_{E} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = \underbrace{4}_{\lambda} \times \underbrace{\begin{pmatrix} 3 \\ 2 \end{pmatrix}}_{E}$$

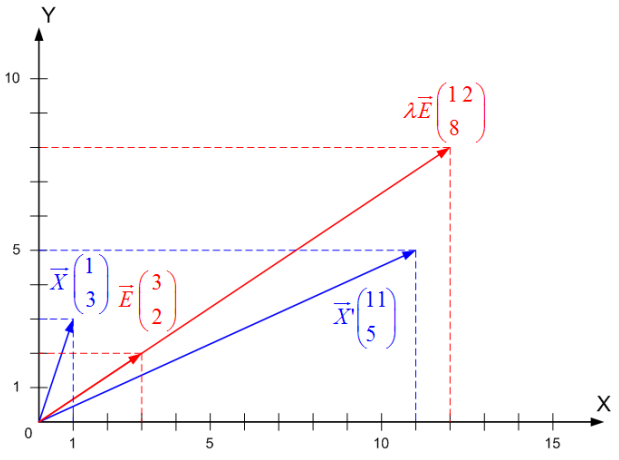


FIG. 1 – Caractérisation d'un vecteur propre E de M. Le vecteur quelconque X devient X' et la direction est changée. Le vecteur propre E devient λE et la direction reste la même.

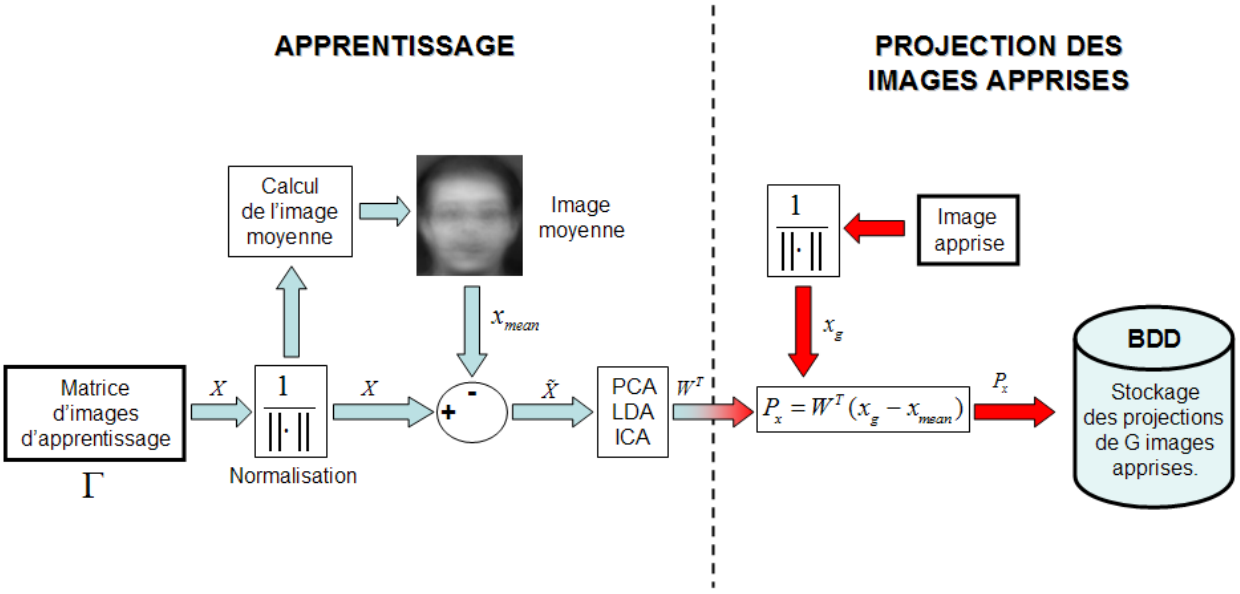


FIG. 2 – Phase d'apprentissage d'un système de reconnaissance faciale utilisant une méthode globale [Delac, 2004]

2.2.2 Propriétés des vecteurs propres

1. Les vecteurs propres n'ont de sens que pour des matrices carrées mais toute matrice carrée ne possède pas forcément des vecteurs propres.
2. Étant donné une matrice de taille $(n \times n)$ qui possède des vecteurs propres, il existe n vecteurs propres.
3. Même si un vecteur propre subit une homothétie quelconque avant d'être multiplié par une matrice de transformation, nous obtenons toujours comme résultat un multiple de celui-ci. Ceci s'explique par le fait qu'une homothétie change la longueur d'un vecteur mais pas sa direction.
4. Tous les vecteurs propres d'une matrice sont *orthogonaux* entre-eux, qu'importe le nombre de dimensions de l'espace vectoriel dans lequel nous travaillons. Cela signifie que l'on est capable d'exprimer les données de départ selon ces vecteurs orthogonaux, au lieu de les exprimer selon les axes x et y.

3 LES MÉTHODES DE RECONNAISSANCE 2D DU VISAGE

3.1 Généralités

Ces méthodes travaillent à partir d'une photo 2D du visage. On peut les diviser en deux catégories : les méthodes globales d'une part, les méthodes locales d'autre part. Dans les deux cas, le processus se déroule en deux étapes bien distinctes : La *phase d'apprentissage* (figure 2) qui s'effectue en amont et dont les calculs sont parfois très longs et la *phase de reconnaissance* (figure 3) qui a lieu lorsqu'un individu se présente physiquement devant le système.

3.2 Processus d'une méthode globale

Par analogie, la phase d'apprentissage correspondrait à un enrollement réel de personnes qui seraient enregistrées dans une base de données. Cette phase consiste donc à récolter une grande quantité d'images de visage afin de se constituer une base de données de départ. Dans un premier temps, on construit une matrice Γ contenant M images de la base d'apprentissage ("Training Set" en anglais). les images sont ensuite normalisées (normalisation géométrique, normalisation de l'histogramme, etc.) puis l'image moyenne est calculée. On réajuste ensuite les données par rapport à la moyenne, pour pouvoir suivre, de manière simple, le comportement des valeurs d'écart-type, de variance et de covariance (rappelons-nous que toutes ces formules font intervenir une soustraction par rapport à la moyenne). On applique alors un algorithme de reconnaissance globale à cette matrice réajustée. La chose à retenir est que ces algorithmes fournissent en sortie ce que l'on appelle une *matrice de projection* W qui va nous être très utile dans la seconde partie de la phase d'apprentissage.
Elle consiste en la projection des images apprises (normalisées et réajustées par rapport à la moyenne) sur un espace vectoriel dont les vecteurs sont les éléments de notre matrice de projection W. Toutes ces projections sont finalement stockées dans une grande base de données.
Passons maintenant à la phase de reconnaissance. Lorsqu'une nouvelle image de la base de Test ("Probe Set" en anglais) est mise devant le système, elle est normalisée et l'image moyenne de la base d'apprentissage lui est retirée. On la projette ensuite sur l'espace vectoriel relatif à la matrice de projection W afin de comparer avec toutes les projections issues de la phase d'apprentissage et qui étaient stockées dans la base de données. Par le terme comparer, il faut comprendre

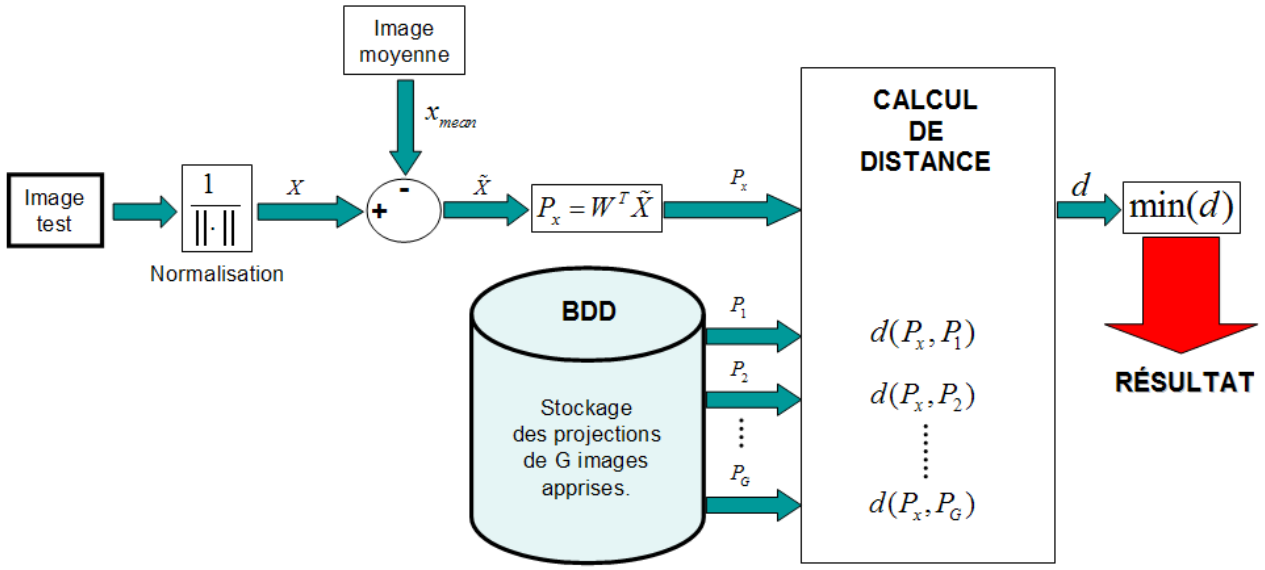


FIG. 3 – Phase de reconnaissance d'un système de reconnaissance faciale utilisant une méthode globale [Delac, 2004]

effectuer un calcul de distance entre les projections vectorielles. Il semble logique que plus la distance entre deux projections est petite, plus ces deux projections se ressemblent. Ainsi, le résultat de la reconnaissance est l'image de la base d'apprentissage qui ressemble le plus à la nouvelle image présentée au système.

Nous venons de voir qu'il est nécessaire de calculer des distances vectorielles entre différentes projections. Autant il est facile de voir ce que peut donner une *mesure de distance vectorielle* en deux voire trois dimensions, autant cela devient beaucoup plus difficile lorsqu'on essaie de se représenter une distance entre des vecteurs à 76800 dimensions ! Pourtant, cela est parfaitement possible d'un point de vue mathématique.

Nous allons voir deux types de mesures de distance vectorielle [Beveridge, 2003]. La première catégorie est constituée de distances Euclidiennes et sont définies à partir de la *distance de Minkowski d'ordre* p dans un espace Euclidien $\mathbf{R}^N$ (N déterminant la dimension de l'espace Euclidien). Considérons deux vecteurs $X = (x_1, x_2, ..., x_N)$ et $Y = (y_1, y_2, ..., y_N)$, la *distance de Minkowski d'ordre* p notée $\mathbf{L}_p$ est définie par :

$$\mathbf{L}_p = \left(\sum_{i=1}^{N} |x_i - y_i|^p \right)^{1/p}$$

C'est à partir de cette formule générique que vont être définies des distances couramment utilisées dans les algorithmes de reconnaissance du visage.

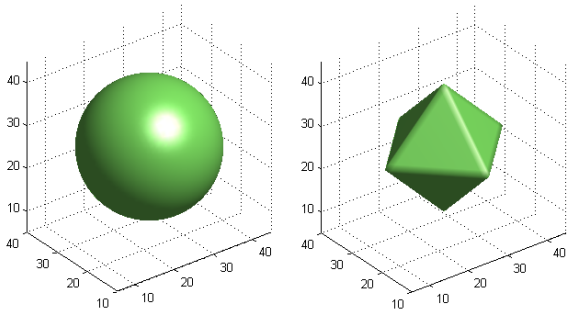
pour $p = 1$, on obtient la distance *City-Block* (ou distance de *Manhattan*) :

$$\mathbf{L}_1 = \sum_{i=1}^{N} |x_i - y_i| \quad (7)$$

Pour $p = 2$, on obtient la distance *Euclidienne* :

$$\mathbf{L}_2 = \sqrt{\sum_{i=1}^{N} |x_i - y_i|^2} \quad (8)$$

Les objets peuvent alors apparaître de façon très différentes selon le type de distance choisie. L'illustration qui suit montre le cas d'une sphère.



(a) (b)

FIG. 4 – Représentation d'une sphère. (a) avec la distance Euclidienne. (b) avec la distance City-Block

La seconde catégorie regroupe des distances non-Euclidiennes. Nous détaillerons une de ces mesures de distances dans le chapitre suivant, lors de la présentation du premier algorithme global.

3.3 Présentation des algorithmes globaux

Ces algorithmes s'appuient sur des propriétés statistiques bien connues et utilisent l'algèbre linéaire. Ils sont relativement rapides à mettre en oeuvre mais sont

sensibles aux problèmes d'éclairement, de pose et d'expression faciale. L'algorithme global le plus connu est sans aucun doute l'*Analyse en Composantes Principales* (**PCA** pour "*Principal Component Analysis*" en anglais). Il est à la base de nombreux algorithmes globaux actuels qui ont, depuis, apporté des améliorations ou quelques variantes. Nous étudierons aussi un algorithme appelée *Analyse Discriminante Linéaire* (**LDA** pour "*Linear Discriminant Analysis*" en anglais) qui permet de trouver des caractéristiques qui séparent relativement bien plusieurs classes.

3.3.1 L'Analyse en Composantes Principales (PCA)

L'algorithme PCA est né des travaux de MA. Turk et AP. Pentland au MIT Media Lab, en 1991. Il est aussi connu sous le nom de *Eigenfaces* car il utilise des vecteurs propres et des valeurs propres (respectivement *Eigenvectors* et *Eigenvalues* en anglais).

L'*idée principale* consiste à exprimer nos M images de départ selon une base de vecteurs orthogonaux particuliers - nos fameux vecteurs propres - contenant des informations indépendantes d'un vecteur à l'autre. Ces nouvelles données sont donc exprimées d'une manière plus appropriée à la reconnaissance du visage.

Dans le langage de la *théorie de l'information*, nous voulons extraire l'information caractéristique d'une image de visage, pour l'encoder aussi efficacement que possible afin de la comparer à une base de données de modèles encodés de manière similaire.

En *termes mathématiques*, cela revient à trouver les vecteurs propres de la matrice de covariance formée par les différentes images de notre base d'apprentissage.

Une image $\mathbf{I}_{i_{(m,n)}}$ est traitée comme un vecteur $\mathbf{\Gamma}_{i_{(m\times n,1)}}$ dans un espace vectoriel de grande dimension ($N = m \times n$), par concaténation des colonnes (figure 5).

Après avoir rassemblé nos M images dans une unique matrice, nous obtenons une *matrice d'images* Γ, où chaque colonne représente une image Γ_i :

$$\mathbf{\Gamma} = \begin{pmatrix} a_{1,1} & b_{1,1} & \dots & z_{1,1} \\ \vdots & \vdots & \dots & \vdots \\ a_{n,1} & b_{n,1} & \dots & z_{n,1} \\ \vdots & \vdots & \dots & \vdots \\ a_{1,m} & b_{1,m} & \dots & z_{1,m} \\ \vdots & \vdots & \dots & \vdots \\ a_{n,m} & b_{n,m} & \dots & z_{n,m} \end{pmatrix}$$

On calcule ensuite l'image moyenne Ψ de toutes les images collectées. Cette image peut être vu comme le *centre de gravité* du jeu d'images :

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{9}$$

$$\Downarrow$$

$$\mathbf{I_1} = \begin{pmatrix} a_{1,1} & \dots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,m} \end{pmatrix} \quad \Rightarrow \quad \mathbf{\Gamma_1} = \begin{pmatrix} a_{1,1} \\ \vdots \\ a_{n,1} \\ \vdots \\ a_{1,m} \\ \vdots \\ a_{n,m} \end{pmatrix}$$

FIG. 5 – Passage d'une image vers un vecteur dans un espace vectoriel de grande dimension. les coefficients $a_{i,j}$ représentent les valeurs des pixels en niveau de gris, codés de 0 à 255

On ajuste ensuite les données par rapport à la moyenne. L'image moyenne est alors soustraite de chaque image avec la formule suivante :

$$\Phi_i = \Gamma_i - \Psi, i = 1...M \tag{10}$$

On calcule ensuite la matrice de covariance du jeu de données. Cette matrice peut être vue comme une *matrice de moments d'ordre 2* :

$$\mathbf{C} = \sum_{i=1}^{M}\Phi_i\Phi_i^T = AA^T, A = [\Phi_1\Phi_2...\Phi_M] \tag{11}$$

La prochaine étape consiste à calculer les vecteurs propres et les valeurs de cette matrice de covariance $\mathbf{C}$ de taille $(N \times N)$, c'est-à-dire *de l'ordre de la résolution d'une image*. Le problème est que cela peut parfois être très difficile et très long ! En effet, si $N > M$ (si la résolution est supérieure au nombre d'images), il y aura seulement $M - 1$ vecteurs propres qui contiendront de l'information (les vecteurs propres restants auront des valeurs propres associées *nulles*). Par exemple, pour 100 images de résolution 320×240, nous pourrions résoudre une matrice $\mathbf{L}$ de 100×100 au lieu d'une matrice de 76800×76800 pour ensuite prendre les combinaisons linéaires appropriées des images Φ_i. Le gain de temps de calcul serait considérable ! Typiquement, nous passerions d'une complexité de l'ordre du nombre de pixels dans une image à une complexité de l'ordre du nombre d'images.

Voici comment procéder pour accélérer les calculs :

Considérons les vecteurs propres e_i de $\mathbf{C} = AA^T$, associés aux valeurs propres λ_i. On a :

$$\mathbf{C}e_i = \lambda_i e_i \tag{12}$$

Les vecteurs propres v_i de $\mathbf{L} = A^T A$, associés aux valeurs propres μ_i sont tels que :

$$\mathbf{L}v_i = \mu_i v_i$$

soit :

$$A^T A v_i = \mu_i v_i$$

En multipliant *à gauche* par A des deux côtés de l'égalité, nous obtenons :

$$AA^T A v_i = A\mu_i v_i$$

Puisque $\mathbf{C} = AA^T$, nous pouvons simplifier :

$$\mathbf{C}(Av_i) = \mu_i(Av_i) \tag{13}$$

De (12) et (13), nous voyons que Av_i et μ_i sont respectivement les vecteurs propres et les valeurs propres de $\mathbf{C}$:

$$\begin{cases} e_i = Av_i \\ \lambda_i = \mu_i \end{cases} \tag{14}$$

Nous pouvons donc trouver les valeurs propres de cette énorme matrice $\mathbf{C}$ en trouvant les valeurs propres d'une matrice $\mathbf{L}$ beaucoup plus petite. Pour trouver les vecteurs propres de $\mathbf{C}$, il suffit juste de pré-multiplier les vecteurs propres de $\mathbf{L}$ par la matrice A.
Les vecteurs propres trouvés sont ensuite ordonnés selon leurs valeurs propres correspondantes, de manière décroissante. Plus une valeur propre est grande, plus la variance capturée par le vecteur propre est importante. Cela implique que la majeure partie des informations est contenue dans les premiers vecteurs propres.

Une part de la grande efficacité de l'algorithme PCA vient de l'étape suivante qui consiste à ne sélectionner que les k meilleurs vecteurs propres (ceux avec les k plus grandes valeurs propres). A partir de là, on définit un espace vectoriel engendré par ces k vecteurs propres, que l'on appelle *l'espace des visages* E_v ("*Face Space*" en anglais).
Les images originales peuvent être reconstituées par combinaison linéaire de ces vecteurs propres. Les représentations graphiques de ces vecteurs rappellent un peu des images fantômes, chacune mettant en avant une partie du visage, on les appelle *Eigenfaces* (figure 6).

Nous allons maintenant projeter nos images de départ sur E_v. Une image Γ_i est alors transformée en ses composantes Eigenfaces par une simple opération de projection vectorielle :

$$\omega_k = {e_k}^T(\Gamma_i - \Psi), k = 1, ..., M' \tag{15}$$

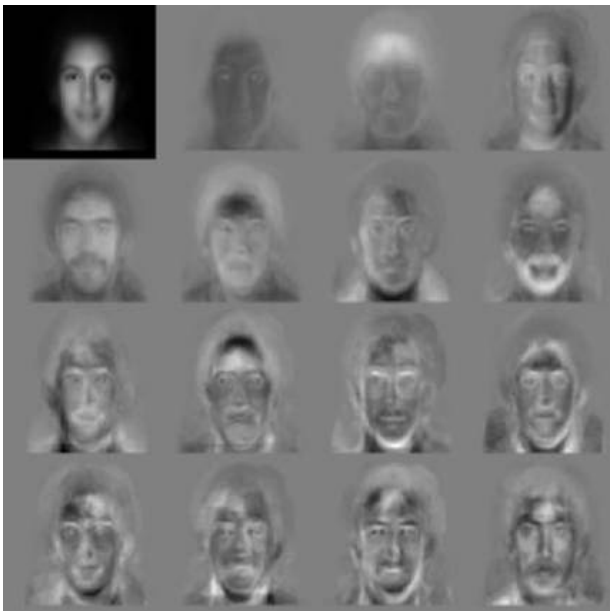


FIG. 6 – Image moyenne et les 15 [res] Eigenfaces

Les vecteurs ω_k sont appelés *poids* et forment une matrice $\Omega^T = [\omega_1, \omega_2, ..., \omega_{M'}]$ qui décrit la contribution de chaque eigenface dans la représentation de l'image d'entrée. La matrice Ω^T est alors utilisée pour trouver quelle est, parmi un nombre pré-défini de classes, celle qui décrit le mieux une image d'entrée.

La méthode la plus simple pour déterminer quelle classe de visage fournit la meilleure description d'une image d'entrée est de trouver la classe de visage k qui minimise la distance Euclidienne.

$$\varepsilon_k^2 = \|\Omega - \Omega_k\|^2 \tag{16}$$

où Ω_k est un vecteur qui décrit la k[e] classe de visage.

Un visage appartient à une classe k quand le minimum ε_k est en dessous d'un certain seuil θ_ε. Dans le cas contraire, le visage est classé comme étant *inconnu* et peut éventuellement être utilisé pour créer une nouvelle classe de visage. La création de la matrice de poids Ω^T est équivalent à la projection du visage original sur E_v. Puisque la distance ε entre l'image de visage et E_v est simplement le carré de la distance entre l'image d'entrée réajustée par rapport à la moyenne $\Phi = \Gamma - \Psi$ et $\Phi_f = \sum_{i=1}^{M'} \omega_i e_i$, sa projection sur E_v est : $\varepsilon^2 = \|\Phi - \Phi_f\|^2$.

Dans le chapitre précédent, nous avions parlé d'un type de distance non-Ecludienne, nous allons la détailler ici :
Nous allons passer de l'espace usuel des images ($\Im_m$) à un espace que l'on appelle l'*espace de Mahalanobis* ($\mathcal{E}_{Mah}$). Avant de continuer, il convient de bien définir ce nouvel espace métrique.

Nous venons de voir, qu'en sortie de l'algorithme PCA, nous obtenons des vecteurs propres associés à des valeurs propres (représentant la variance selon chaque

dimension). Ces vecteurs propres définissent une rotation vers un espace dont la covariance entre les différentes dimensions est nulle. L'espace de Mahalanobis est un espace où la variance selon chaque dimension est égale à 1. On l'obtient à partir de l'espace des images $\Im_m$ en divisant chaque vecteur propre par son écart-type correspondant.
Soit u et v deux vecteurs propres de $\Im_m$, issus de l'algorithme PCA, et m et n deux vecteurs de $\mathcal{E}_{Mah}$. Soit λ_i les valeurs propres associées aux vecteurs u et v, et σ_i l'écart-type, alors on définit $\lambda_i = \sigma_i^2$. Les vecteurs u et v sont reliés aux vecteurs m et n à partir des relations suivantes :

$$m_i = \frac{u_i}{\sigma_i} = \frac{u_i}{\sqrt{\lambda_i}} \quad et \quad n_i = \frac{v_i}{\sigma_i} = \frac{v_i}{\sqrt{\lambda_i}} \tag{17}$$

Nous pouvons maintenant définir la distance *MahCosine* (pour Cosinus dans l'espace de Mahalanobis). Il s'agit tout simplement du cosinus de l'angle entre les vecteurs u et v, une fois qu'ils ont été projetés sur E_v et normalisées par la variance.

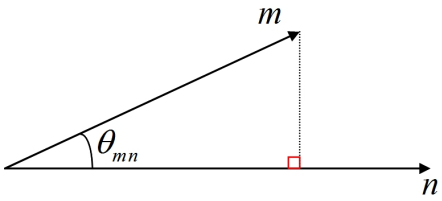


FIG. 7 – Les deux vecteurs m et n dans l'espace de Mahalanobis

Nous avons donc par définition :

$$d_{MahCosine}(u, v) = \cos(\theta_{mn})$$

De plus, on peut écrire :

$$\cos(\theta_{mn}) = \frac{|m||n| \times \cos(\theta_{mn})}{|m||n|}$$

D'où la formule finale de la distance *MahCosine* :

$$d_{MahCosine}(u, v) = \frac{m \cdot n}{|m||n|} \tag{18}$$

Cette mesure de distance associée à l'algorithme PCA donne, en général, de meilleurs résultats, que lors de l'utilisation de la distance Euclidienne (figure 23).

Revenons à nos projections sur E_v, il y a alors, *quatre possibilités* (tableau 1 et figure 8) pour une image d'entrée d'être reconnue ou non :
Dans le cas 1, un individu est reconnu et identifié. Dans le cas 2, un individu inconnu est présent. Les deux derniers cas (3 et 4) indiquent que l'image n'est pas une image de visage. Pour le cas 3 , l'image est éloigné de E_v mais la projection est proche d'une classe connue, on parle alors de *fausse acceptation*.

	espace des Visages (E_v)	Classe de Visage
Cas 1	proche	proche
Cas 2	proche	éloigné
Cas 3	éloigné	proche
Cas 4	éloigné	éloigné

TAB. 1 – Les quatres possibilités qui apparaissent lors de la phase de reconnaissance

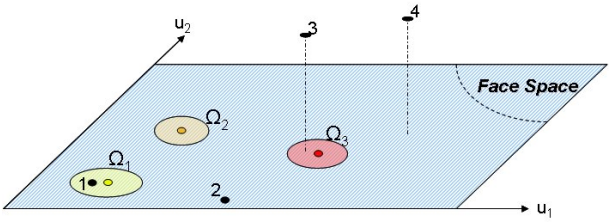


FIG. 8 – Une version simplifiée de E_v illustrant les quatre résultats de la projection d'une image sur E_v. Dans ce cas, il y a deux vecteurs propres ($u1$ et $u2$) et trois classes d'individus connus ($\Omega1$, $\Omega2$, $\Omega3$).

Ce que l'on peut retenir de l'algorithme PCA

Tout d'abord, l'algorithme PCA est une méthode globale utilisant en premier lieu les niveaux de gris des pixels d'une image. Sa simplicité à mettre en oeuvre contraste avec une forte sensibilité aux changements d'éclairement, de pose et d'expression faciale.
Néanmoins, le PCA ne nécessite aucune connaissance à priori sur l'image et se révèle plus efficace lorsqu'il est couplé à la mesure de distance MahCosine.
Le principe selon lequel on peut construire un sous-espace vectoriel en ne retenant que les « meilleurs »vecteurs propres, tout en conservant beaucoup d'information utile, fait du PCA un algorithme efficace et couramment utilisé en réduction de dimensionnalité où il peut alors être utilisé en amont d'autres algorithmes (comme le LDA par exemple).
Enfin, l'étude théorique de l'algorithme PCA est très pédagogique et permet d'acquérir de solides bases pour la reconnaissance 2D du visage. C'est un algorithme incontournable !

3.3.2 L'Analyse Discriminante Linéaire (LDA)

L'algorithme LDA est né des travaux de Belhumeur et al. de la Yale University (USA), en 1997. Il est aussi connu sous le nom de *Fisherfaces*.
Contrairement à l'algorithme PCA, l'algorithme LDA effectue une véritable *séparation de classes* (figure 9).
Pour pouvoir l'utiliser, il faut donc au préalable organiser la base d'apprentissage d'images en plusieurs classes : une classe par personne et plusieurs images par classe.
Le LDA analyse les vecteurs propres de la matrice de dispersion des données, avec pour objectif de maximiser les variations inter-classes tout en minimisant les variations intra-classes.

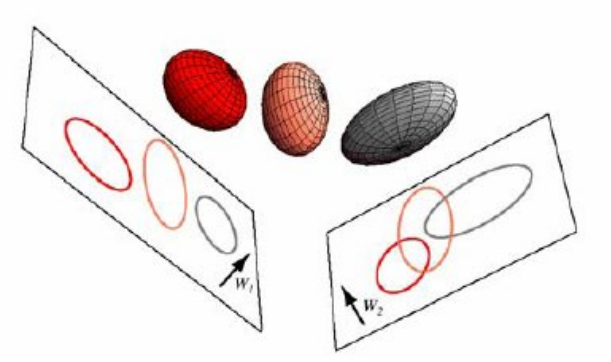


FIG. 9 – Illustration du principe de séparation optimale des classes par le LDA. Trois distributions 3D sont projetées sur deux sous-espaces 2D décris par les vecteurs $\mathbf{W_1}$ et $\mathbf{W_2}$. Puisque le LDA essaye de trouver la plus grande séparation parmi les classes, on voit bien que $\mathbf{W_1}$ est ici le vecteur optimal [Gul, 2003]

Tout comme dans le PCA, on rassemble les images de la base d'apprentissage dans une grande matrice d'images Γ où chaque colonne représente une image Γ_i, puis on calcule l'image moyenne Ψ.
Ensuite, pour chaque classe C_i, on calcule l'image moyenne Ψ_{C_i} :

$$\Psi_{C_i} = \frac{1}{q_i} \sum_{k=1}^{q_i} \Gamma_k \tag{19}$$

avec q_i, le nombre d'images dans la classe C_i.

Chaque image Γ_i de chaque classe C_i est ensuite recentrée par rapport à la moyenne. On obtient alors une nouvelle image Φ_i :

$$\Phi_i = \Gamma_i - \Psi_{C_i} \tag{20}$$

Vient ensuite le calcul de nos différentes matrices de dispersion. On notera c le nombre total de classes (i.e. le nombre d'individus), q_i le nombre d'images dans la classe C_i et M le nombre total d'images.

1. La Matrice de Dispersion Intra-Classe ($\mathbf{S}_w$)

$$\mathbf{S}_w = \sum_{i=1}^{c} \sum_{\Gamma_k \in C_i} (\Gamma_k - \Psi_{C_i})(\Gamma_k - \Psi_{C_i})^T \tag{21}$$

2. La Matrice de Dispersion Inter-Classe ($\mathbf{S}_b$)

$$\mathbf{S}_b = \sum_{i=1}^{c} q_i \, (\Psi_{C_i} - \Psi)(\Psi_{C_i} - \Psi)^T \tag{22}$$

3. La Matrice de Dispersion Totale ($\mathbf{S}_T$)

$$\mathbf{S}_T = \sum_{i=1}^{M} (\Gamma_i - \Psi)(\Gamma_i - \Psi)^T \tag{23}$$

Une fois ces matrices calculées, nous devons trouver une projection optimale W qui maximise la dispersion intra-classe, relative à la matrice $\mathbf{S}_w$, tout en minimisant la dispersion inter-classe, relative à la matrice $\mathbf{S}_b$.
En d'autres termes, nous devons trouver W qui maximise le *critère d'optimisation de Fisher* $J(T)$:

$$W = \arg\max_T (J(T))$$

$$\Rightarrow \quad \max(J(T)) = \frac{|T^T \mathbf{S}_b T|}{|T^T \mathbf{S}_w T|} | T = W \tag{24}$$

W peut alors être trouvée en résolvant le problème généralisé aux valeurs propres [Golub, 2004] :

$$\mathbf{S}_b W = \lambda_w \mathbf{S}_w W \tag{25}$$

Une fois W trouvée, le même schéma que le PCA concernant la projection des images apprises ainsi que la projection d'une image test est appliqué.
Ainsi, la projection vectorielle d'une image apprise réajustée par rapport à la moyenne Φ_i est définie par :

$$g(\Phi_i) = W^T \Phi_i$$

La phase de reconnaissance d'une image test Φ_t s'effectue en projetant Φ_t sur W^T :

$$g(\Phi_t) = W^T \Phi_t$$

Enfin, on effectue une mesure de distance entre l'image test et l'image projetée sur l'espace vectoriel engendré par W^T. Par exemple, pour la distance Euclidienne, on calcule la distance d_{ti} :

$$d_{ti} = \|g(\Phi_t) - g(\Phi_i)\|$$

d'où :

$$d_{ti} = \sqrt{\sum_{k=1}^{c} (g(\Phi_t) - g(\Phi_i))^2} \tag{26}$$

Finalement, une image test est dans la classe dont la distance est minimale par rapport à toutes les autres distances de classe.

Ce que l'on peut retenir de l'algorithme LDA

Tout d'abord, l'algorithme LDA permet d'effectuer une véritable séparation de classes, selon un critère mathématique qui minimise les variations entre les images d'un même individu (variations intra-classe) tout en maximisant les variations entre les images d'individus différents (variations inter-classes). Cependant, pour des problèmes « sous-échantillonnés »en reconnaissance du visage, c'est-à-dire lorsque le nombre d'individus à traiter est plus faible que la résolution de l'image, il est difficile d'appliquer le LDA qui peut alors faire apparaître des matrices de dispersions singulières (non inversibles). Afin de contourner ce problèmes, certains algorithmes basés sur le LDA ont récemment été mis au point (par exemple, les algorithmes ULDA, OLDA, NLDA).

3.4 Les méthodes locales

Les méthodes locales consistent à appliquer des transformations en des endroits spécifiques de l'image, le plus souvent autour de *points caractéristiques* (coins des yeux, de la bouche, le nez, ...). Elles nécessitent donc une connaissance *a priori* sur les images. Ces méthodes sont plus lourdes à mettre en place mais sont plus robustes aux problèmes posés par les variations d'éclairement, de pose et d'expression faciale. Nous allons étudier un de ces algorithmes qui donne de très bons résultats, et dont les fondements ont un lien étroit avec les neurosciences.

3.4.1 L'Elastic Bunch Graph Matching (EBGM)

L'algorithme EBGM est né des travaux de Wiskott et al. de la Southern California University (USC - USA) et de la Rurh University (Allemagne), en 1997.

À partir d'une image de visage, on localise des points caractéristiques (coins des yeux, de la bouche, nez, etc.). Cette localisation peut se faire manuellement ou automatiquement à l'aide d'un algorithme [Arca, 2005]. Un *treillis élastique virtuel* est ensuite appliqué sur l'image de visage à partir de ces points. Chaque point représente un *nœud labélisé* auquel on associe un jeu de coefficients d'ondelettes complexes de Gabor, appelés *Jet*. Pour effectuer une reconnaissance avec une image test, on fait une mesure de similarité entre les différents Jets et les longueurs des segments du treillis de deux images.

Pour comprendre comment les ondelettes de Gabor parviennent à jouer un rôle fondamental dans cet algorithme, nous allons faire un peu de neurosciences [Meyer, 2001]. Considérons la partie du cortex visuel humain à la base de la reconnaissance des objets et des visages : le *cortex visuel primaire* **V1** (figure 10). Cette partie du cerveau contient de nombreux neurones, chacun étant spécialisé dans une tâche de prétraitement (plus ou moins complexe) de l'information visuelle issue de la rétine. Dans le V1, Hubel et Wiesel [Hubel, 1968] ont identifié un sous-groupe de ces neurones appelé « *cellules simples* ». Ces cellules sont spécialisées dans la *reconnaissance des contours* et sont activées lors de l'apparition ou de la disparition d'un *stimulus visuel* dans une zone discrète que l'on appelle la *fovéa* (figure 11), qui se situe au centre de la rétine. Un modèle simple de réponses de ces cellules spécialisées est appelé *champ réceptif linéaire* (**RF**).

En **1981**, les expériences de **Pollen** et **Ronner** [Pollen, 1981] montrent que la plupart de ces cellules simples peuvent être combinées par paire. La première avec une symétrie paire que l'on modélise mathématiquement par une fonction cosinus, la seconde possédant une symétrie impaire que l'on modélise par une fonction sinus.

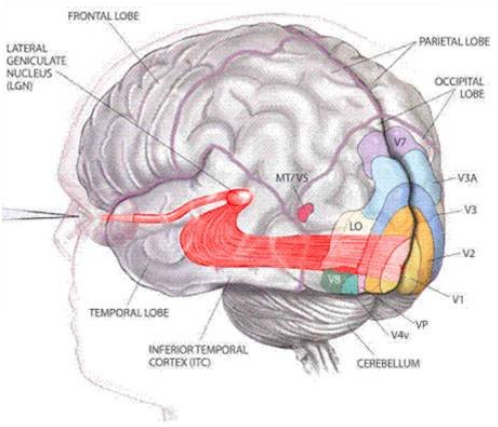


FIG. 10 – Localisation du cortex visuel primaire dans le cerveau humain [Albert, 2005]

FIG. 11 – Localisation de la fovéa par rapport à la rétine

En **1987**, les résultats de **Jones** et **Palmer** [Jones, 1987] suggèrent de modéliser la forme des champs réceptifs linéaires par des *filtres de Gabor* 2D (figure 12).

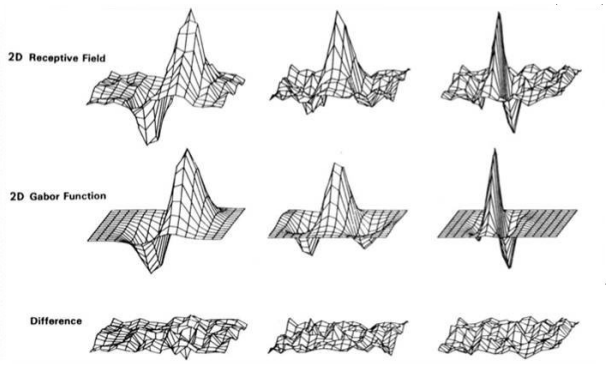


FIG. 12 – Structure spectrale 2D de RFs simples dans un cortex strié de chat. On peut apprécier la grande ressemblance entre le modèle mathématique des ondelettes 2D de Gabor et les réponses en fréquence des cellules simples de $\mathbf{V1}$

Ainsi, la réponse $a_j(x_0)$ d'une telle cellule peut être exprimée comme la corrélation de la donnée en entrée (i.e. une image $\mathbf{I}(x)$) et du modèle simple **RF** $\Psi_j(x - x_0)$:

$$a_j(x_0) = \int \mathbf{I}(x)\Psi_j(x - x_0)dx \tag{27}$$

Quittons maintenant les neurosciences pour revenir à notre algorithme. Dans l'introduction, nous avions utilisé le terme de *Jet*. Par ailleurs, nous venons de voir que

nous pouvons modéliser le modèle simple de réponses **RF** par des ondelettes 2D de Gabor. Il est alors plus facile de comprendre que, dans notre algorithme, le Jet correspond à la *réponse électrophysiologique* $a_j(x_0)$.
Ainsi, un *Jet* est basé sur une *transformée en ondelettes* [Hubbard, 1995], défini comme la convolution d'une image avec une famille de *noyaux de Gabor*. Ces noyaux de Gabor peuvent être assimilés à des ondes localisées dans le temps, modulées par une Gaussienne (figure 13). On peut parler de transformée en ondelettes car la famille des noyaux de Gabor est générée à partir d'une *ondelette mère* par dilatation et rotation.

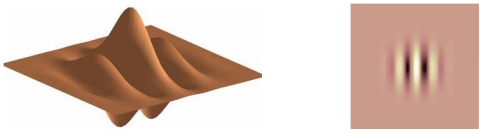


FIG. 13 – Représentation 3D (à gauche) et 2D (à droite) de la partie réelle d'un filtre de Gabor.

Dans l'algorithme EBGM, les ondelettes de Gabor sont des fonctions de type $f(\theta, \lambda, \phi, \sigma, \gamma)$. Voici une explication des 5 paramètres :

1. L'orientation de l'ondelette (θ)
   Ce paramètre fait pivoter l'ondelette autour de son centre. L'orientation de l'ondelette détermine l'angle des contours ou des lignes de l'image auxquelles l'ondelette va être sensible.

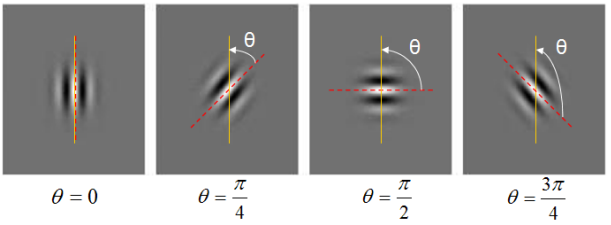


FIG. 14 – Orientation de l'ondelette

2. La fréquence centrale de l'ondelette (λ)
   Ce paramètre spécifie la longueur d'onde du cosinus ou inversement la fréquence centrale de l'ondelette. Les ondelettes avec une grande longueur d'onde seront sensibles à des changements progressifs d'intensité dans une image. Les ondelettes avec une petite longueur d'onde seront sensibles à des contours et des lignes abruptes.

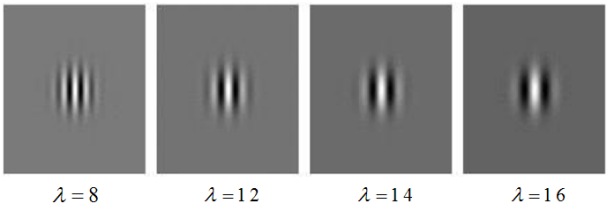


FIG. 15 – Fréquence centrale de l'ondelette

3. La phase de la sinusoïde (ϕ)
   On utilise à la fois la partie réelle et la partie imaginaire de l'ondelette complexe de Gabor. Ce qui nous donne, en quelque sorte, deux ondelettes : une ondelette paire et une ondelette impaire. La convolution (incluant alors deux phases) donne un coefficient complexe basé sur deux ondelettes qui sont déphasées de $\frac{\pi}{2}$.

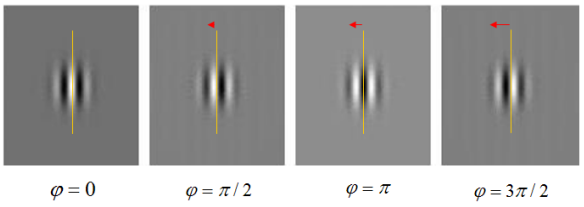


FIG. 16 – Phase de l'ondelette

4. Le support temporel de l'ondelette (σ)
   Ce paramètre spécifie le rayon de la Gaussienne. La taille de la Gaussienne détermine la quantité de pixels de l'image qui vont être pris en compte dans la convolution.

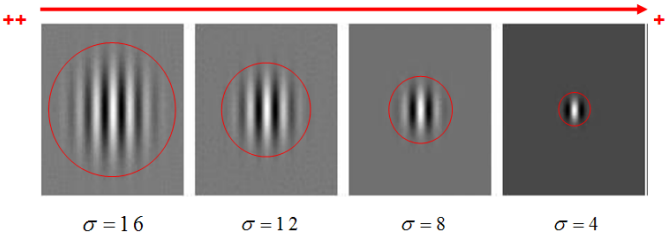


FIG. 17 – Support temporel de l'ondelette

5. L'enveloppe de la Gaussienne (γ)
   Ce paramètre agit sur la forme de l'enveloppe Gaussienne, en l'étirant spatialement. Ce paramètre a été inclus de manière à ce que les ondelettes puissent approximer certains modèles biologiques. La plupart des ondelettes testées avec l'algorithme EBGM du CSU System 5.0 [CSU, 2003] utilisent une enveloppe Gaussienne avec $\gamma = 1$.

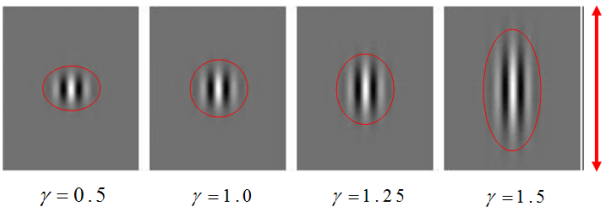


FIG. 18 – Forme de l'enveloppe Gaussienne

En utilisant 5 fréquences différentes, 8 orientations différentes, et 2 phases différentes. On obtient un total de 80 masques d'ondelettes de Gabor différents (figure 19).

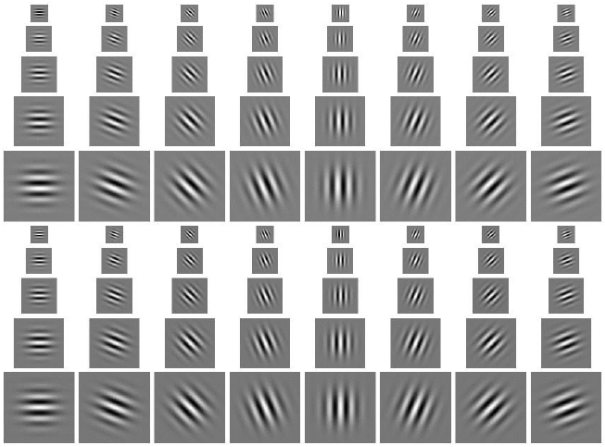


FIG. 19 – Les 80 masques d'ondelettes de Gabor

Maintenant que nous avons vu comment paramétrer l'ondelette de Gabor, nous allons résumer les différentes étapes de l'algorithme EBGM avec l'aide de quelques illustrations.

1. Sélection de points caractéristiques
   Des points caractéristiques d'une image de visage sont sélectionnés (manuellement ou avec un algorithme).

FIG. 20 – La sélection de points caractéristiques

2. Création du treillis
   Un treillis est construit en reliant les points caractéristiques précédemment trouvés.

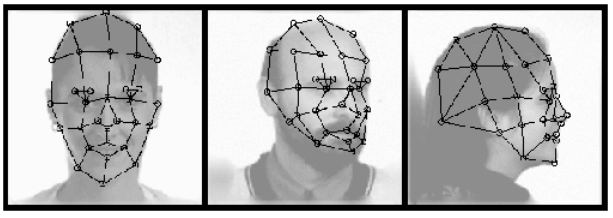


FIG. 21 – Création du treillis

3. Calcul des Jets
   À chaque nœud du treillis correspond un point caractéristique et contient un jeu de coefficients complexes d'ondelettes de Gabor : le Jet. Les différents Jets sont calculés en convoluant l'image autour des points caractéristiques avec plusieurs ondelettes de Gabor paramétrées. Un Jet peut être écrit comme un ensemble de coefficients complexes $\mathcal{J}_j = a_j \exp(i\phi_j)$ avec une amplitude a_j qui varie peu avec la position, et une phase ϕ_j, dont la variation en rotation est plus importante. La représentation obtenue est appelée "Face Bunch Graph"(FBG) (figure 22).

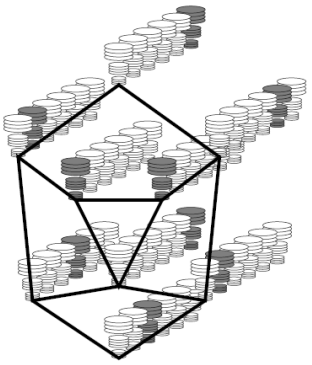


FIG. 22 – Le "Face Bunch Graph" sert de représentation générale pour les visages. Chaque empilement de disques représente un Jet. À partir d'un jeu de Jets relié à un nœud du treillis, seulement celui qui correspond le mieux est sélectionné pour la reconnaissance (indiqué en ombre grisée à titre d'exemple).

4. Calcul de similarité de deux images
   Une fois la structure du FBG trouvée, l'algorithme va effectuer un calcul de similarité entre une image de la base d'apprentissage et une image test. Pour cela, les points caractéristiques de l'image test sont trouvés, le treillis est mis en place et les nouveaux Jets calculés. Précisons une dernière fois que l'on associe au treillis non seulement la localisation des points caractéristiques mais aussi les différents Jets. La similarité de deux images est alors une fonction de la correspondance des treillis. La reconnaissance finale se fait en maximisant cette fonction.

Ce que l'on peut retenir de l'algorithme EBGM

Tout d'abord, l'algorithme EBGM trouve ses fondements dans les neurosciences, en imitant le fonctionnement de certaines cellules spécialisées localisées dans le cortex visuel primaire. C'est un algorithme local (il ne traite pas directement les valeurs de niveaux de gris des pixels d'une image de visage), ce qui lui confère une plus grande robustesse aux changements d'éclairement, de pose et d'expression faciale. Cependant il est plus difficile à implémenter que les méthodes globales PCA et LDA précédemment exposées, et le temps de preprocessing s'en retrouve augmenté. Enfin, une partie de son originalité provient du fait que l'EBGM utilise des ondelettes entièrement paramétrables pour générer des coefficients complexes qui vont être utilisés lors de la phase de reconnaissance.

4 MESURE DE LA PERFORMANCE D'UN ALGORITHME DE RECONNAISSANCE DU VISAGE

Selon la nature du système de reconnaissance, il existe deux façons de mesurer la performance d'un algorithme de reconnaissance du visage.

La courbe **CMC** (pour "Cumulative Match Characteristic" en anglais) est utilisée pour mesurer la performance d'un système d'*identification* (on utilise aussi le terme de reconnaissance $1 : n$). Cette courbe (figure 23) donne le pourcentage de personnes reconnues en fonction d'une variable que l'on appelle le *rang*. On dit qu'un système reconnaît au rang 1 lorsqu'il choisit la plus proche image comme résultat de la reconnaissance. On dit qu'un système reconnaît au rang 2, lorsqu'il choisit, parmi deux images, celle qui correspond le mieux à l'image d'entrée, etc. On peut donc dire que plus le rang augmente, plus le taux de reconnaissance correspondant est lié à un niveau de sécurité plus faible.

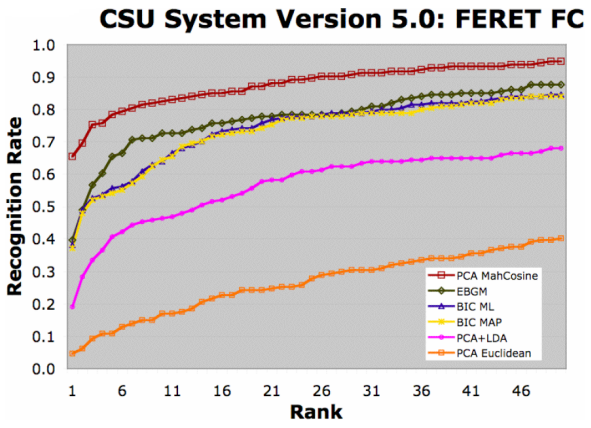


FIG. 23 – CMC du CSU System 5.0 obtenue pour le Probe Set FERET FC (tests de changements en éclairement) avec différents algorithmes, dont le PCA, le LDA et l'EBGM [Beveridge, 2005].

En revanche, la courbe **ROC** (pour "Receiver Operating Characteristic" en anglais) permet de mesurer la performance d'un système d'*authentification* (on utilise aussi le terme de reconnaissance $1 : 1$). La courbe ROC (figure 24) trace le *Taux de Faux Rejet* (FRR)[5] en fonction du *Taux de Fausse Acceptation* (FAR)[6].

La courbe CMC n'est qu'une autre manière d'afficher les données et peut être calculée à partir du FAR et du FRR. Une étude comparative précisant le lien entre les courbes CMC et ROC peut être trouvée dans [Bolle, 2006].

[5]Le FRR représente le pourcentage de personnes censées être reconnues par le système mais qui sont « rejetées ».

[6]Le FAR représente le pourcentage de personnes censées ne pas être reconnues par le système mais qui sont « acceptées ».

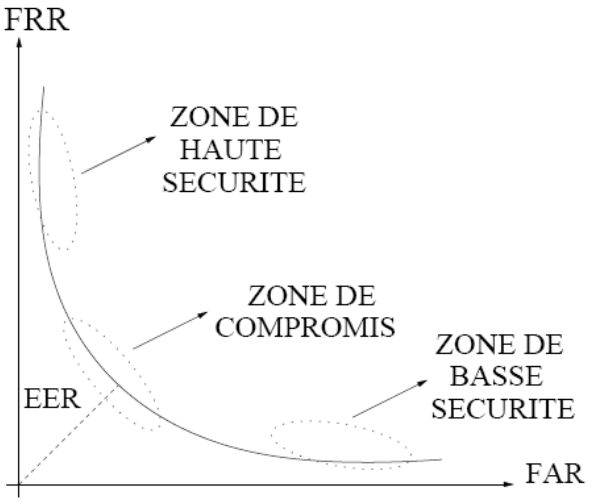


FIG. 24 – Courbe ROC [Perronnin, 2002]

5 CONCLUSION

Nous avons tout d'abord présenté les bases mathématiques nécessaires à la compréhension des algorithmes classiques en reconnaissance 2D du visage. Nous avons ensuite étudié deux algorithmes globaux, le premier (PCA) qui dérive d'une méthode statistique classique pour mettre en avant des similitudes et des différences entre des données issues d'un espace vectoriel de grande dimension, le deuxième (LDA) qui permet d'effectuer une véritable séparation de classes. Le choix a ensuite été fait de présenter un algorithme local (EBGM) qui cherche à modéliser le fonctionnement du cortex visuel primaire humain et qui utilise des ondelettes 2D de Gabor. Enfin, une explication de deux mesures possibles de la performance d'un algorithme de reconnaissance faciale a été exposé.

6 REMERCIEMENTS

Une partie des travaux présentés dans ce papier utilise certaines images de visage issues de la "Color FERET Database" [FERET, 2003], ainsi que certaines images de visage issues de la "AT&T Database" [ATT, 2002].

7 RÉFÉRENCES ET CITATIONS

BIBLIOGRAPHIE

[Albert, 2005] Albert M.V., "Nonlinearity in Primary Visual Cortex", Informal Talk, Nonlinear systems IGERT group, 2005.

[ATT, 2002] The AT&T Database, Adresse Internet :`http://www.cl.cam.ac.uk/Research/DTG/attarchive/facedatabase.html`, 2002.

[Arca, 2005] Arca S., Campadelli P., Lanzarotti R., "A Face Recognition System Based On Automatically Determined Facial Fiducial Points", 2005.

[Belhumeur, 1996] Belhumeur P., Hespanha J., Kriegman D., "Eigenfaces vs. Fisherfaces : Recognition Using Class Specific Linear Projection", Proc. of the Fourth European Conference on Computer Vision, Vol. 1, Cambridge, UK, 1996.

[Beveridge, 2003] Beveridge R., Bolme D., Teixeira M., Draper B., "The CSU Face Identification Evaluation System User's Guide : Version 5.0", Computer Science Department Colorado State University, 2003.

[Beveridge, 2005] Beveridge R., Kirby M., "Biometrics and Face Recognition", IS&T Colloquium, p.25, 2005.

[Bolle, 2006] Bolle R.M., Connell J.H., Pankanti S., Ratha N.K., Senior A.W., "The Relation between the ROC Curve and the CMC", IBM Research Report, 2006.

[Bolme, 2003] Bolme D.S., "Elastic Bunch Graph Matching", M.Sc. Thesis, Colorado State University (CSU), 2003.

[CSU, 2003] The CSU Face Identification Evaluation System (Version 5.0), Adresse Internet :`http://www.cs.colostate.edu/evalfacerec/index.html`,2003.

[Delac, 2004] Delac K., Grgic M., Grgic S., "Independent Comparative Study of PCA, ICA, and LDA on the FERET Data Set", Technical Report, University of Zagreb, FER, 2004.

[FERET, 2003] The Color FERET Database, Adresse Internet :`http://www.itl.nist.gov/iad/humanid/colorferet/home.html`, 2003.

[Gul, 2003] Gul A.B., "Holostic Face Recognition by Dimension Reduction", M.Sc., School of Natural and Applied Sciences of the Middle East Technical University, Department of Electrical and Electronics Engineering, 2003.

[Golub, 2004] Golub G.H., "The Generalized Eigenvalue Problem", Lectures on Matrix Computation, Ph.D program of the Dipartimento di Matematica "Istituto Guido Castelnuovo", Lecture No. 11, Roma, 2004.

[Hubbard, 1995] Hubbard Burke B., "Ondes et ondelettes, La Saga d'un Outil Mathématique", Belin pour la science, 1995.

[Hubel, 1968] Hubel, D. H. and Wiesel, T. N., "Receptive fields and functional architecture of monkey striate cortex", J. Physiol., 195, 215-243, London, 1968.

[Jones, 1987] Jones J. P. Palmer L. A., "The two-dimensional spectral structure of simple receptive fields in cat striate cortex", Journal of Neurophysiology, Vol. 58, No. 6, pp. 1187-1211, December 1987.

[Meyer, 2001] Meyer Y., "Images et vision", 2001.

[Perronnin, 2002] Perronnin F., Dugelay J-L., "Introduction à la Biométrie - Authentification des Individus par Traitement Audio-Vidéo", Revue Traitement du Signal, Vol. 19, No. 4, Sophia Antipolis, 2002.

[Pollen, 1981] Pollen D.A., Ronner S.F., "Phase relationship between adjacent simple cells in the visual cortex", Science, 212 :1409-1411, 1981.

[Smith, 2004] Smith, Lindsay I., "A Tutorial on Principal Component Analysis", `http://csnet.otago.ac.nz/cosc453`, 2002.

[Turk, 1991] Turk M., Pentland A., "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, Vol. 3, No. 1, 1991.

[Wiskott, 1997] Wiskott L., Fellous J.M., Krüger N., Von Der Malsburg C., "Face Recognition by Elastic Bunch Graph Matching", IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7) :775-779, 1997.