Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 MMIS First Semester 2010/2011

Lesson 9 5 January 2011

Face Image Recognition with Principal Components

Outline:

Recognizing Normalised Face Images.

Principal Components Analysis was first proposed as to determine features for face image recognition by Mathew Turk and Sandy Pentland at MIT In 1991. The method requires that the face image be normalized in position, orientation and size. Typically this can be provided by detecting the face position and size with a cascade classifier (lecture 8), and then texture mapping the face region into a standard sized window. We will refer to such a normalized image window as an "imagette".

A typical useful sized window is size 32 x 32 pixels. The method can be successful used with imagettes as small as 24 x 24 and has been demonstrated with smaller imagettes, although discrimination falls off rapidly with sizes below 16 x 16 pixels.

To use the method we require a training set of images for each person. Suppose that we have M_k normalized face imagettes $\{W_m^k\}$ for each of K persons. Our training data is composed of

$$\{W_m\} = \bigcup_k \{W_m^k\}, \; M = \sum_k M_k$$

We will use the set $\{W_m\}$ to learn a set of D feature bases images φ_D.
We will then project the training set onto the feature space to obtain a feature vector for each face imagette:

$$\vec{X}_m^k = \left\langle \vec{\Phi}, W_m^k \right\rangle \text{ where each component is } x_{dm}^k = \left\langle \varphi_d, W_m^k \right\rangle$$

We then train a Bayesian classifier using the training set $\{\vec{X}_m^k\}$.
This can be done using EM to learn a Gaussian mixture model for each face class.
Many alternative techniques are possible.

Principle components analysis of imagettes.

The following derivation can be made using 2-D Imagettes, W(i,j).
However, for notation reasons, it is often convenient to map the imagettes onto a 1-D vector W(n) using the following.

Assume that the imagette is of size I columns by J rows.

For each i,j, compute n = j*I + i
Then W(n) = W(i, j). W(n) has N=I x J pixels.

Principal components analysis is a method to determine a linear subspace that is optimal for reconstructing a set of vectors.

Assume a set of M training vectors (imagettes): $\{W_m\}$
We are going to use the training data to determine an orthogonal basis set

$$\vec{\varphi}(n) = \begin{pmatrix} \varphi_1(n) \\ \varphi_2(n) \\ \vdots \\ \varphi_D(n) \end{pmatrix}$$

such that D < K < M to represent $W_m(n)$.
To do this we compute

Average imagette: $$\mu(n) = \frac{1}{M}\sum_{m=1}^{M} W_m(n)$$

Zero Mean Imagettes: $$V_m(n) = W_m(n) - \mu(n)$$

Projection of $W(n)$ onto the orthogonal basis

$$x_d = \langle W(n), \varphi_d(n) \rangle = \sum_{n=1}^{N} W(n) \cdot \varphi_d(n)$$

or : $$\vec{X} = \langle W(n), \vec{\varphi}(n) \rangle = \sum_{n=1}^{N} W(n) \cdot \vec{\varphi}(n) \text{ for d=1,…,D}$$

Reconstruction: $$\hat{W}(n) = \mu(n) + \sum_{d=1}^{D} x_d \varphi_d(n)$$

Residue image: $$R(n) = W(n) - \hat{W}(n)$$

Residue Energy: $\varepsilon_R^2 = \sum_{n=1}^{N} R(n)^2$

The orthogonal bases are constructed as the Principal Components of the covariance of $\{V_n(m)\}$.

This covariance matrix has N x N = N^2 coefficients.

$$\mathbf{C} = E\{V_m(n)V_m(n)^T\}$$

Assume the matrix **V**

$$\boldsymbol{V} = \left[V_1(n)V_1(n)...V_m(N)\right] = \begin{bmatrix} V_1(1) & V_2(1) & ... & V_M(1) \\ V_1(2) & V_2(2) & ... & V_M(2) \\ \vdots & \vdots & \ddots & \vdots \\ V_1(n) & V_2(n) & ... & V_M(n) \end{bmatrix}$$

V has N lignes and M columns.
Each column is an image, m. Each row is a pixel (i,j).

$$C_n = \boldsymbol{V}\boldsymbol{V}^T = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

For an N=IxJ imagette there are N^2 coefficients in $\mathbf{C}_n$.
For example, in a 32 x 32 imagette, $\mathbf{C}_n$ is of size 1024x1024.

The coefficients of $\mathbf{C}_n$ are the covariances for the pixels, n.

$$\sigma_{ij}^2 = \frac{1}{M}\sum_{m=1}^{M} V_m(i)V_m(j)$$

The principal components of $\mathbf{C}_n$ are the direction vectors for a rotation that diagonalizes $\mathbf{C}_n$.

$$\varphi^T \boldsymbol{C}_n \varphi = \varphi^T \boldsymbol{V}\boldsymbol{V}^T \varphi = \Lambda_n$$

such that the Λ is a diagonal matrix composed of N principal values : λ_n.
In English these are called the Eigen-vectors $\varphi_n(n)$ and the Eigen-values, λ_n

We can obtain these vectors from an algorithm for diagonalising large matrices, such as Householder's method or SVM. (See numerical recipes in C).

$(\varphi_n(n),\ \Lambda_n) \leftarrow \mathrm{PCA}(\mathbf{C_n})$.

For an imagette of size 32 x 32, the matrix $C_n = V V^T$ is of size 2^{10} x $2^{10} = 2^{20}$ coefs.
For an image of size 512 x 512, the matrix $C_n = V V^T$ is of size 2^{18} x $2^{18} = 2^{36}$ coefs.

Most diagonalisation methods are capable of handling matrices of up to N = 32.

For larger imagettes, there is a clever trick that can be used provided that M < N.

Instead of computing $C_n = VV^T$ we can computer $C_m = V^T V$

C_m will be the MxM covariance between image pairs.
Each term is the covariance of two images, *k, l.*

$$\sigma_{kl}^2 = \frac{1}{N}\sum_{n=1}^{N} V_k(n) V_l(n)$$

We can look for a rotation matrix R such that

$$R^T (V^T V)\ R = \Lambda_m$$

Because the same information is used, the first M principal components C_m are the same as those of C_n

$$\Lambda_n = \left(\begin{array}{c|ccc} \Lambda_m & 0 & 0 & 0 \\ \hline 0 & \lambda_{m+1} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{array}\right)$$

and we can note that

$$R\,R^T\,V^T\,V\,R = R\Lambda_m$$

R is a square matrix such that $R^TR = RR^T = I$.

Thus

$$V^TV\;R = R\,\Lambda_m$$

Now multiple by V

$$V\,(V^TV)\,R = (V\,V^T)\,V\,R = V\,R\;\Lambda_m$$

$$= (VV^T)(V\,R) = (V\,R)\;\Lambda_m$$

and Λ_m is an MxM submatrix of Λ_N

As a result, if we substitute $\varphi_m = (V\,R)$
Where φ_m is the first m terms of φ

We see that for the first m terms,

$$= (VV^T)\,\varphi_m = \varphi_m\;\Lambda_m$$

or

$$\varphi_m^T\;(VV^T)\;\varphi_m = \Lambda_m$$

Thus we can use $\varphi_m = (V\,R)$
as the principal components of C_n

Thus

$$(VV^T)\;\varphi = \varphi\,\Lambda_n = (V\,R)\;\Lambda_m$$

$$\varphi_m(n) = \sum_{l=1}^{M} V_l(n)R(l,m)$$

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

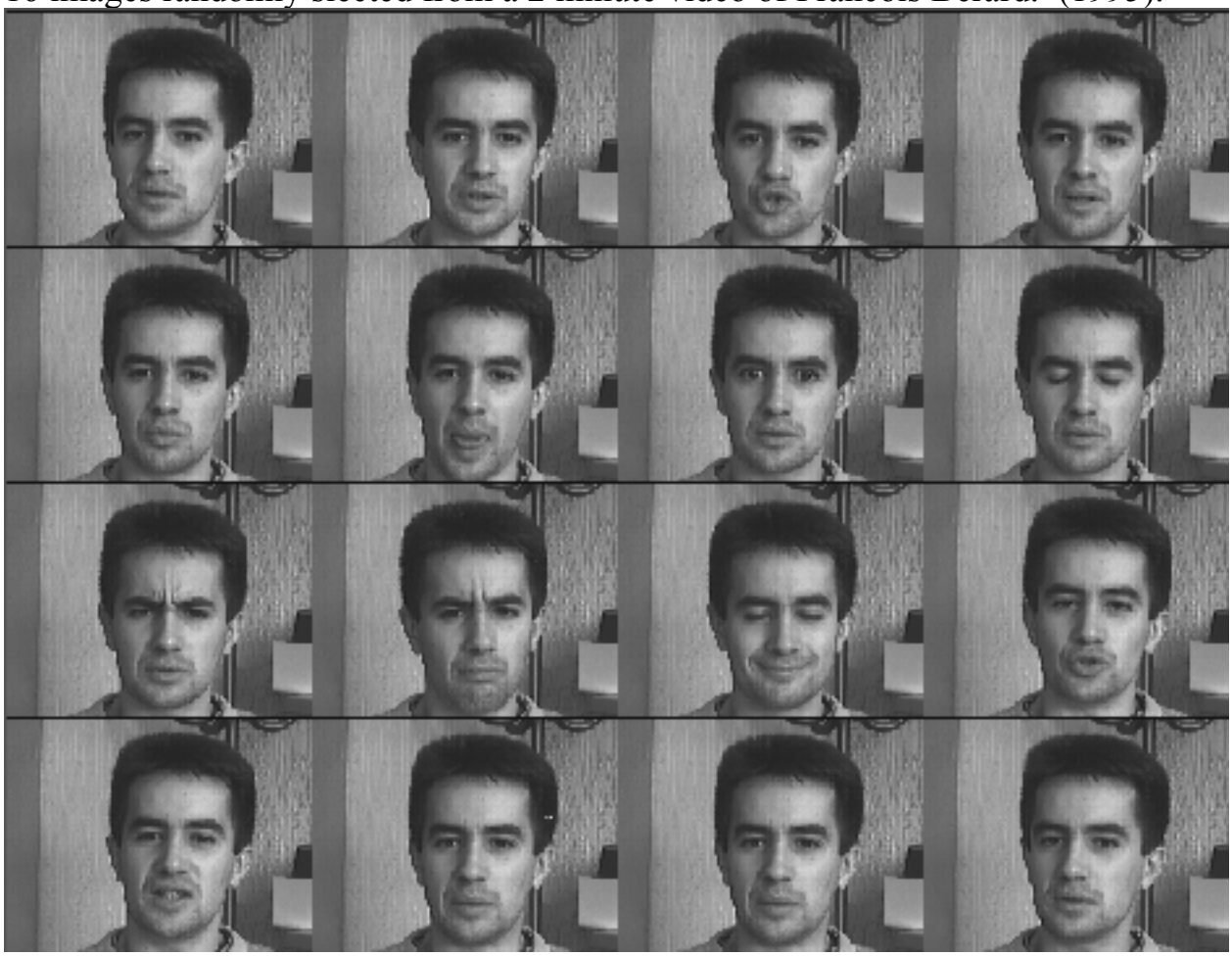
Reason: The same information was used to construct C and C_m.

For $x_d = \langle W(n), \varphi_d(n) \rangle = \sum_{n=1}^{N} W(n) \cdot \varphi_d(n)$

The coeficients x_d are a code that represents V(n).

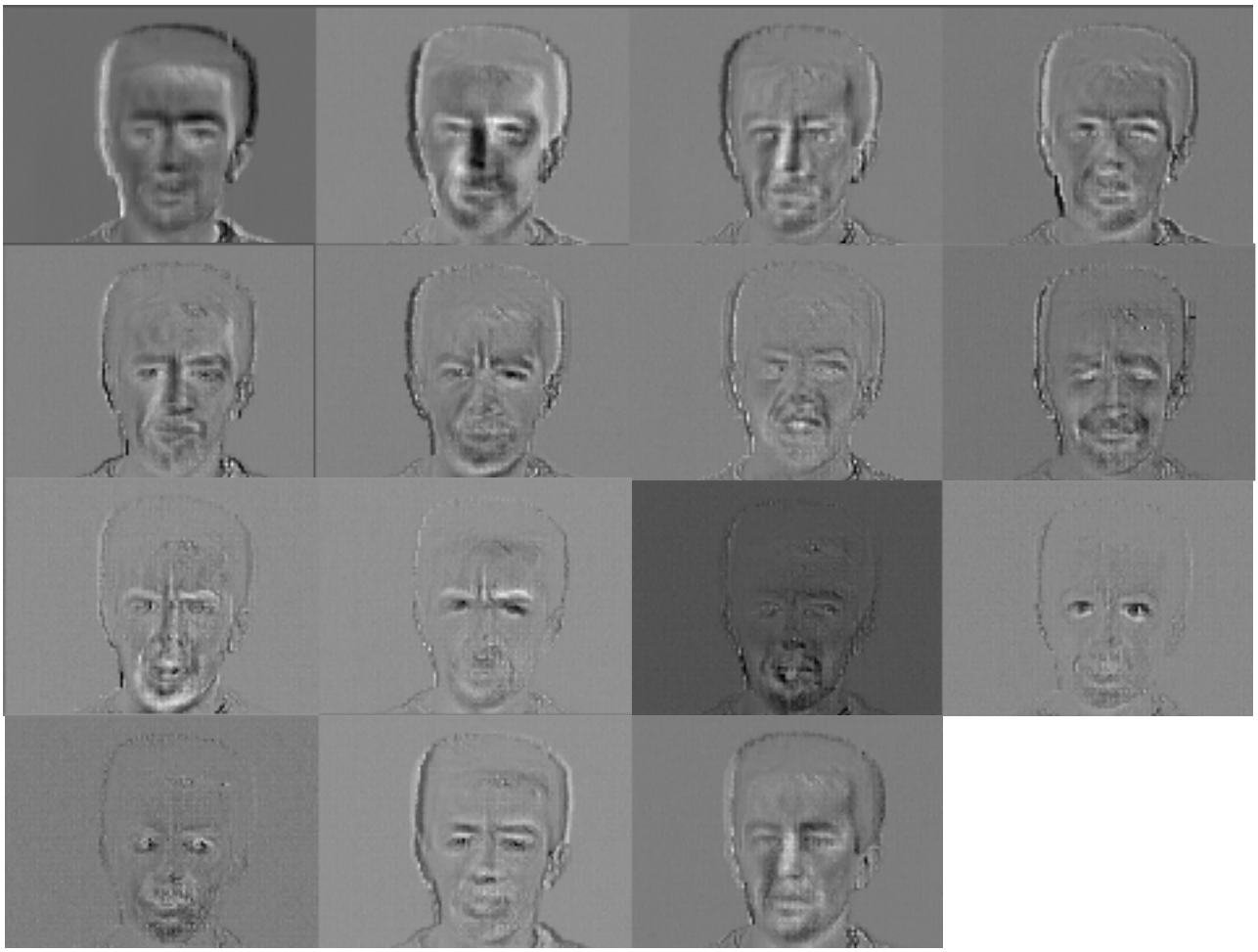
Example :

16 images randomly slected from a 2 minute video of Francois Berard. (1995).

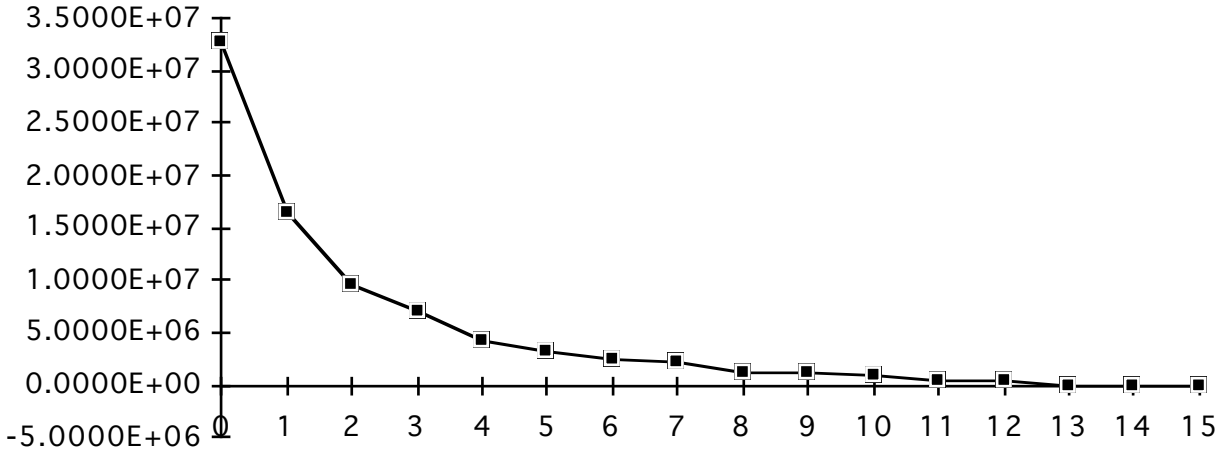


Average Image

Principals Components images



Eigen Values.



Reconstruction

Reconstruction: $\hat{W}(n) = \mu(n) + \sum_{d=1}^{D} x_d \varphi_d(n)$

Image Reconstructed image (120 bytes) Error Image.

Reconstruction (120 bytes)

Image Error

Eigen Faces as a basis for recognition.

The set Eigen basis images $\varphi_n(n)$ or $\varphi_m(m)$ are called "eigenfaces".

The Eigen faces form a linear subspace of $\{W_m\}$ that is optimal for reconstruction. If all N bases re used, then any imagette from $\{W_m\}$ can be perfectly reconstructed (except for round off error). For any subset D < N of $\varphi_n(n)$, the residue error will be smaller than with any other basis of D vectors.

Reconstruction: $\hat{W}(n) = \mu(n) + \sum_{d=1}^{D} x_d \varphi_d(n)$

Residue image: $R(n) = W(n) - \hat{W}(n)$

Residue Energy: $\varepsilon_R^2 = \sum_{n=1}^{N} R(n)^2$

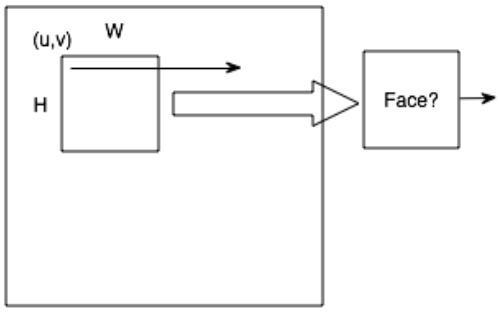
In practice, if there are variations in illumination, these will dominate the first eigen values. In this case the corresponding eigen vectors are not useful for recognition and can be omitted.

Distance from Face Space

The residue image can be used to determine if a new face imagette, W(m), is "similar" to the eigenspace (linear subspace). In this case, the residue is called the "Distance from Face Space" (DFS)

The disctance from Face Space can be used as a face detector!

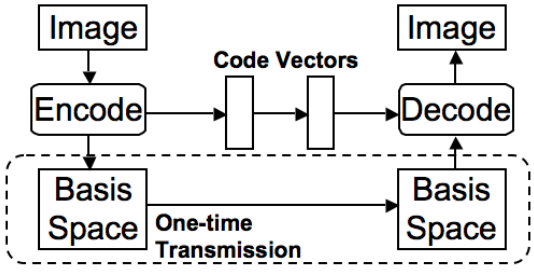
We scan the image different size windows, texture map each window to a standard size, then computer the residue (DFS). If the DFS is small, the window contains a face similar to the Face space.



In practice, this method is less effective and more expensive than the cascade classifiers seen last lecture.

Eigenspace coding for transmission.

Eigen space coding is a very effective method for signal compression!.



In 1996 we were able to demonstrate video telephony in real time (video rates) over a 9600 baud serial line! We ran a video conf with MIT over the internet of the period. In this demo, we used 32 coefficients per image! (32 x 4 - 128 bytes/image).

Eigenspace Coding for Face Recognition.

In 1991, Eigen space coding was a revolutionary technique for face recognition, because it was the first technique to actually work!

However, testing soon revealed that it could only work with:
1) Controlled lighting
2) Pre-defined face orientation (i.e. Cooperating Subject)
3) Normalised image position and size.
4) No occlusions
5) Limited size population
In other conditions the results are unreliable.

In particular, recognition with unconstrained face orientation remains an unsolved problem.