Principal component analysis

PCA

- Images are high dimensional correlated data.
- Goal of PCA is to reduce the dimensionality of the data by retaining as much as variation possible in our original data set.
- The simplet way is to keep one variable and discard all others: not reasonable! Or we can reduce dimensionality by combining features.
- In PCA, we can see intermediate stage as visualization.
- It is based on Eigen value and Eigen Vector.

Image Representation

- Training set of m images of size $N*N$ are represented by vectors of size N^2

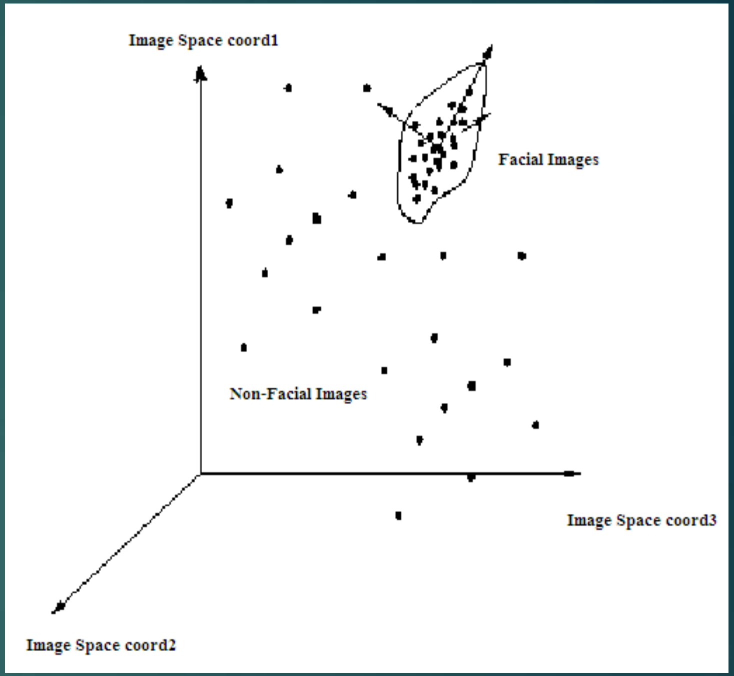
$$x_1, x_2, x_3, \ldots, x_M$$

Example

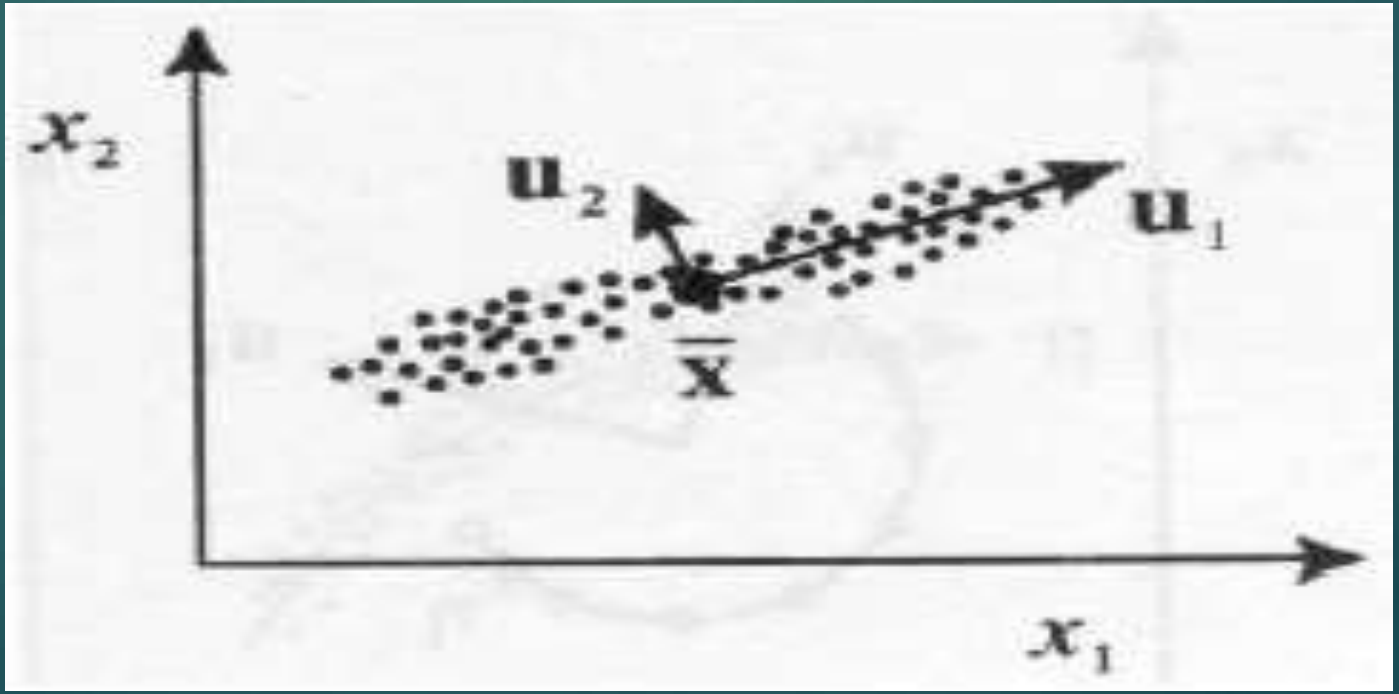
$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & -1 & 2 \\ 4 & 5 & 1 \end{bmatrix}_{3\times 3} \longrightarrow \begin{bmatrix} 1 \\ 2 \\ 3 \\ 3 \\ -1 \\ 2 \\ 4 \\ 5 \\ 1 \end{bmatrix}_{9\times 1}$$

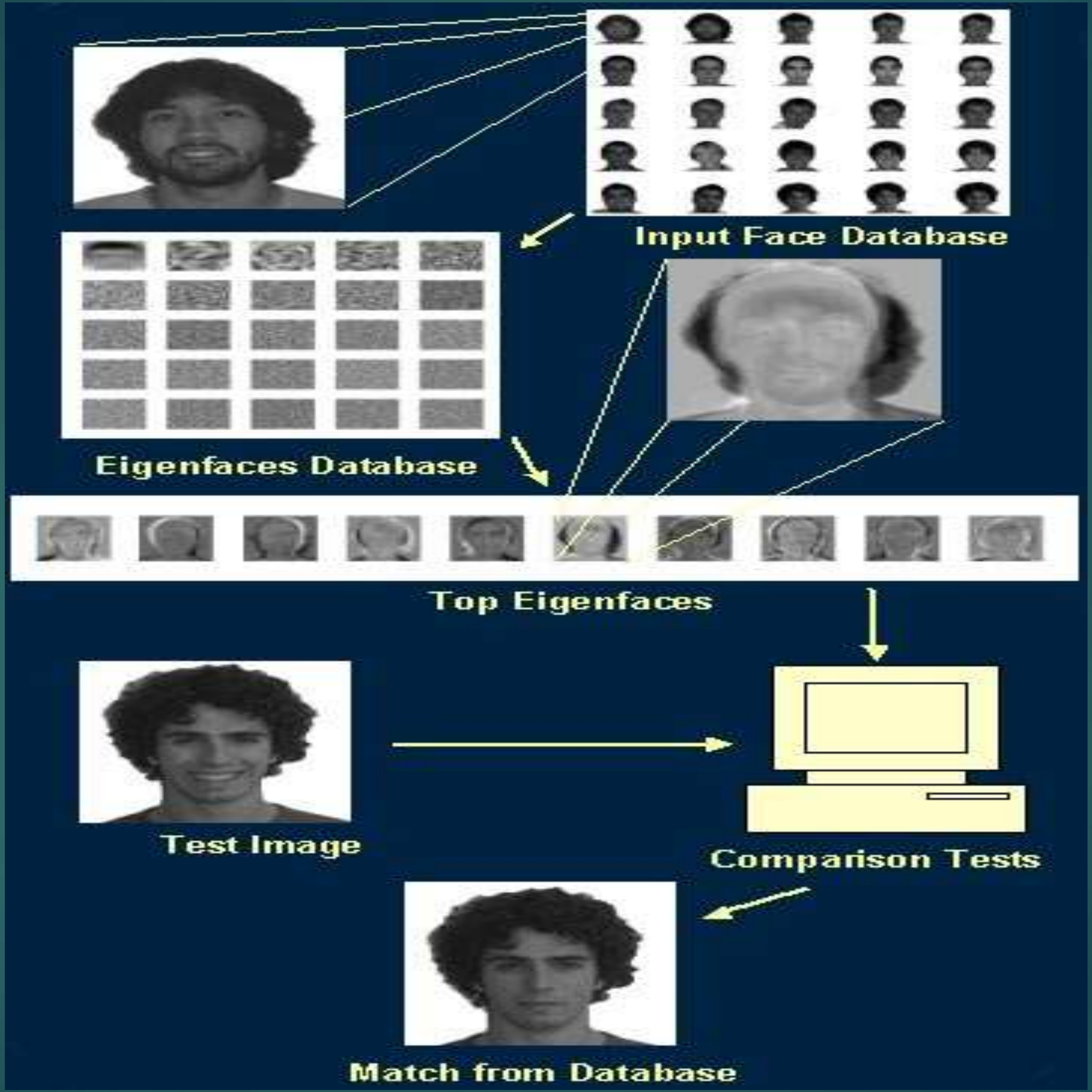
Principal Component Analysis

- A N *x N* pixel image of a face, represented as a vector occupies a single point in N^2-dimensional image space.
- Images of faces being similar in overall configuration, will not be randomly distributed in this huge image space.
- Therefore, they can be described by a low dimensional subspace.
- Main idea of PCA for faces:
  - To find vectors that best account for variation of face images in entire image space.
  - These vectors are called eigen vectors.
  - Construct a face space and project the images into this face space (eigenfaces).



- Geometric interpretation
  - PCA projects the data along the directions where the data varies the most.
  - These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
  - The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.




Input Face Database
Eigenfaces Database
Top Eigenfaces
Test Image
Comparison Tests
Match from Database

Eigenfaces, the algorithm

- Assumptions
  - Square images with Width = Height = N
  - M is the number of images in the database
  - P is the number of persons in the database

Eigenfaces, the algorithm

- The database

$$= \begin{pmatrix} a_1 \\ a_2 \\ \mathrm{M} \\ a_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} b_1 \\ b_2 \\ \mathrm{M} \\ b_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} c_1 \\ c_2 \\ \mathrm{M} \\ c_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} d_1 \\ d_2 \\ \mathrm{M} \\ d_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} e_1 \\ e_2 \\ \mathrm{M} \\ e_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} f_1 \\ f_2 \\ \mathrm{M} \\ f_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} g_1 \\ g_2 \\ \mathrm{M} \\ g_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} h_1 \\ h_2 \\ \mathrm{M} \\ h_{N^2} \end{pmatrix}$$

Eigenfaces, the algorithm

- We compute the average face

$$\vec{m} = \frac{1}{M}\begin{pmatrix} a_1 + b_1 + \cdots + h_1 \\ a_2 + b_2 + \cdots + h_2 \\ \vdots \quad \vdots \quad \quad \vdots \\ a_{N^2} + b_{N^2} + \cdots + h_{N^2} \end{pmatrix}, \quad \textit{where } M = 8$$

Eigenfaces, the algorithm

- Then subtract it from the training faces
- This reduce the common things to each face.

$$\vec{a}_m = \begin{pmatrix} a_1 & - & m_1 \\ a_2 & - & m_2 \\ \vdots & & \vdots \\ a_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{b}_m = \begin{pmatrix} b_1 & - & m_1 \\ b_2 & - & m_2 \\ \vdots & & \vdots \\ b_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{c}_m = \begin{pmatrix} c_1 & - & m_1 \\ c_2 & - & m_2 \\ \vdots & & \vdots \\ c_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{d}_m = \begin{pmatrix} d_1 & - & m_1 \\ d_2 & - & m_2 \\ \vdots & & \vdots \\ d_{N^2} & - & m_{N^2} \end{pmatrix},$$

$$\vec{e}_m = \begin{pmatrix} e_1 & - & m_1 \\ e_2 & - & m_2 \\ \vdots & & \vdots \\ e_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{f}_m = \begin{pmatrix} f_1 & - & m_1 \\ f_2 & - & m_2 \\ \vdots & & \vdots \\ f_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{g}_m = \begin{pmatrix} g_1 & - & m_1 \\ g_2 & - & m_2 \\ \vdots & & \vdots \\ g_{N^2} & - & m_{N^2} \end{pmatrix}, \quad \vec{h}_m = \begin{pmatrix} h_1 & - & m_1 \\ h_2 & - & m_2 \\ \vdots & & \vdots \\ h_{N^2} & - & m_{N^2} \end{pmatrix}$$

Eigenfaces, the algorithm

- Now we build the matrix which is N^2 by M

$$A = \begin{bmatrix} \vec{a}_m & \vec{b}_m & \vec{c}_m & \vec{d}_m & \vec{e}_m & \vec{f}_m & \vec{g}_m & \vec{h}_m \end{bmatrix}$$

- The covariance matrix which is N^2 by N^2

$$Cov = AA^{\mathrm{T}}$$

Eigenfaces, the algorithm

- PCA assumes that the information is carried in the variance of the features
- Find eigenvalues of the covariance matrix
  - The matrix is very large
  - The computational effort is very big
- We are interested in at most M eigenvalues
  - We can reduce the dimension of the matrix

Eigenfaces, the algorithm

- Compute another matrix which is M by M

$$L = A^{\mathrm{T}} A$$

- Find the M eigenvalues and eigenvectors
  - Eigenvectors of Cov and L are equivalent
- Build matrix V from the eigenvectors of L

Eigenfaces, the algorithm

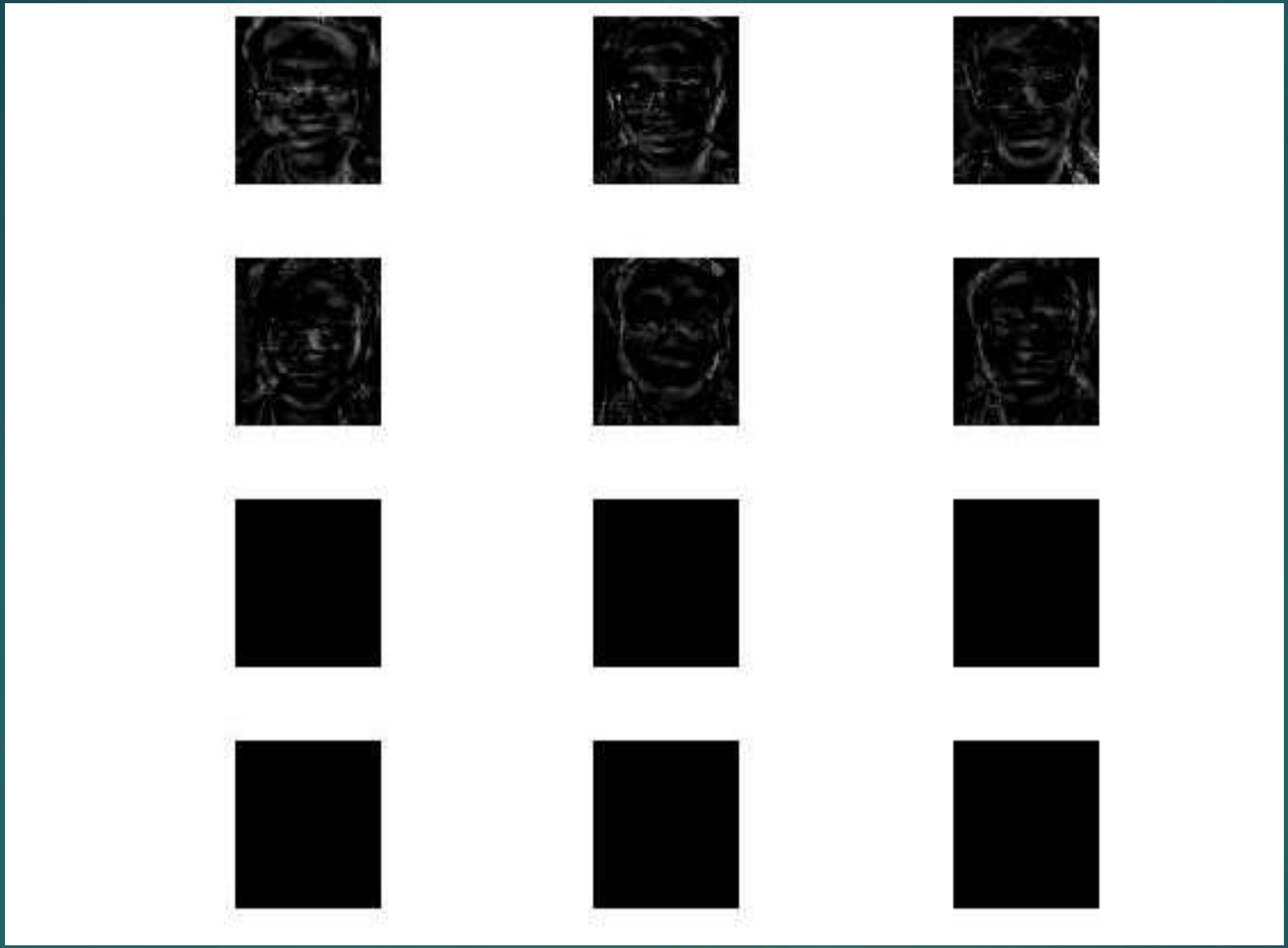
- Eigenvectors of Cov are linear combination of image space with the eigenvectors of L

$$U = AV$$

V is Matrix of eigenvectors

$$A = \left[\vec{a}_m \ \vec{b}_m \ \vec{c}_m \ \vec{d}_m \ \vec{e}_m \ \vec{f}_m \ \vec{g}_m \ \vec{h}_m \right]$$

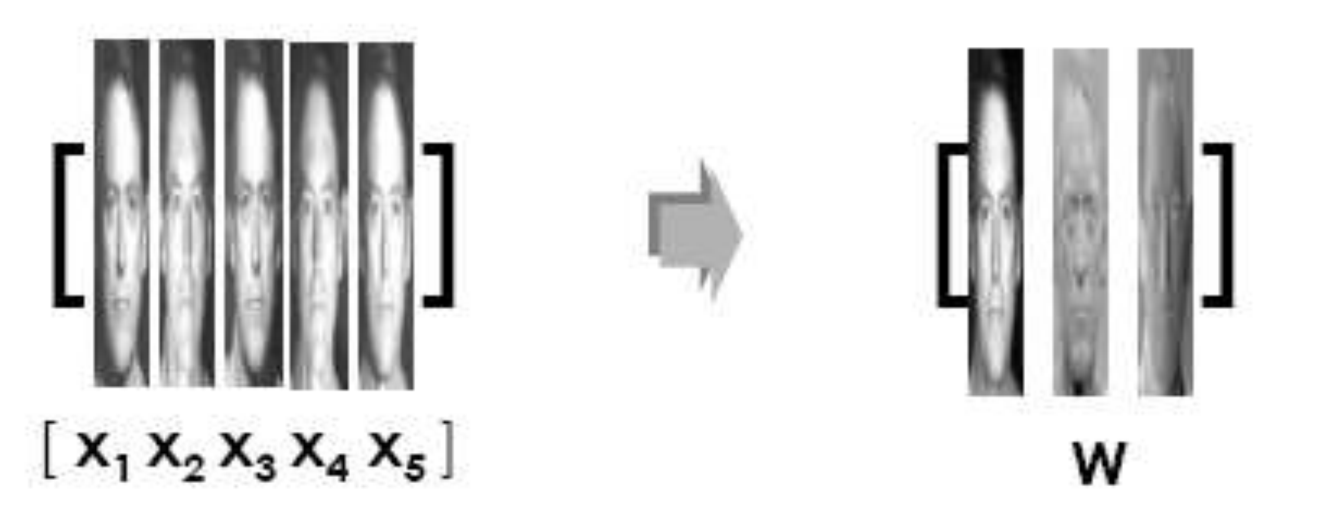
- Eigenvectors represent the variation in the faces



Eigen Face

- Eigen vectors resembles facial images which look like ghostly and are called Eigen faces.
- Eigen faces correspond to each face in the free space and discard the faces for which Eigen value is zero, thus reducing the Eigen face to an extent.
- The Eigen faces are ranked according to their usefulness in characterizing the variation among the images.
- After it we can remove the last less significant Eigen faces.

Eigenfaces, the algorithm



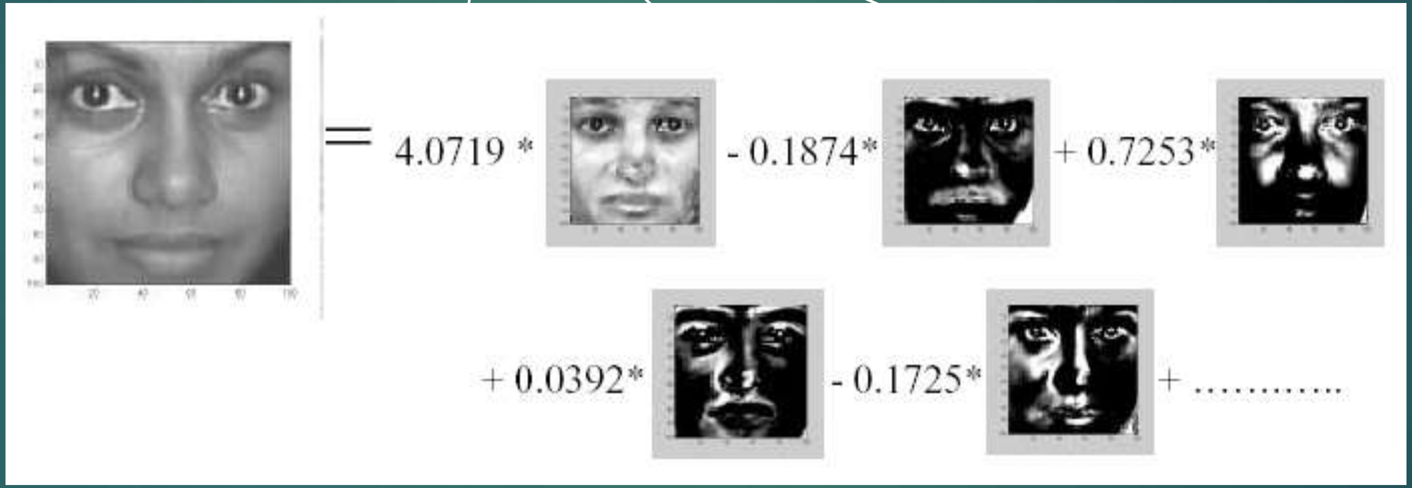
A: collection of the training faces

U: Face Space / Eigen Space

Transform into ‘Face Space’

- Projection

$$f = U * (I - A)$$



Face space

Transform known faces to face space

Eigenfaces: **Recognition Procedure**

- To recognize a face

$$= \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{N^2} \end{pmatrix}$$

- Subtract the average face from it

$$\vec{r}_m = \begin{pmatrix} r_1 - m_1 \\ r_2 - m_2 \\ \vdots \quad \vdots \\ r_{N^2} - m_{N^2} \end{pmatrix}$$

Eigenfaces, the algorithm

- Compute its projection onto the face space U

$$\Omega = U^{T}(\vec{r}_m)$$

- Compute the distance in the face space between the face and all known faces
- Minimum is answer.

$$\varepsilon_i^2 = \left\| \Omega - \Omega_i \right\|^2 \quad for\ i = 1..M$$

- Problems
  - Background (de-emphasize the outside of the face – e.g., by multiplying the input image by a 2D Gaussian window centered on the face)
  - Lighting conditions (performance degrades with light changes)
  - Scale (performance decreases quickly with changes to head size)
  - Orientation (performance decreases but not as fast as with scale changes)
    - plane rotations can be handled
    - out-of-plane rotations are more difficult to handle