Face Recognition

CPSC 4600/5600 @ UTC/CSE

Face Recognition

- Introduction
- Face recognition algorithms
- Comparison
- Short summary

Face Recognition Algorithms

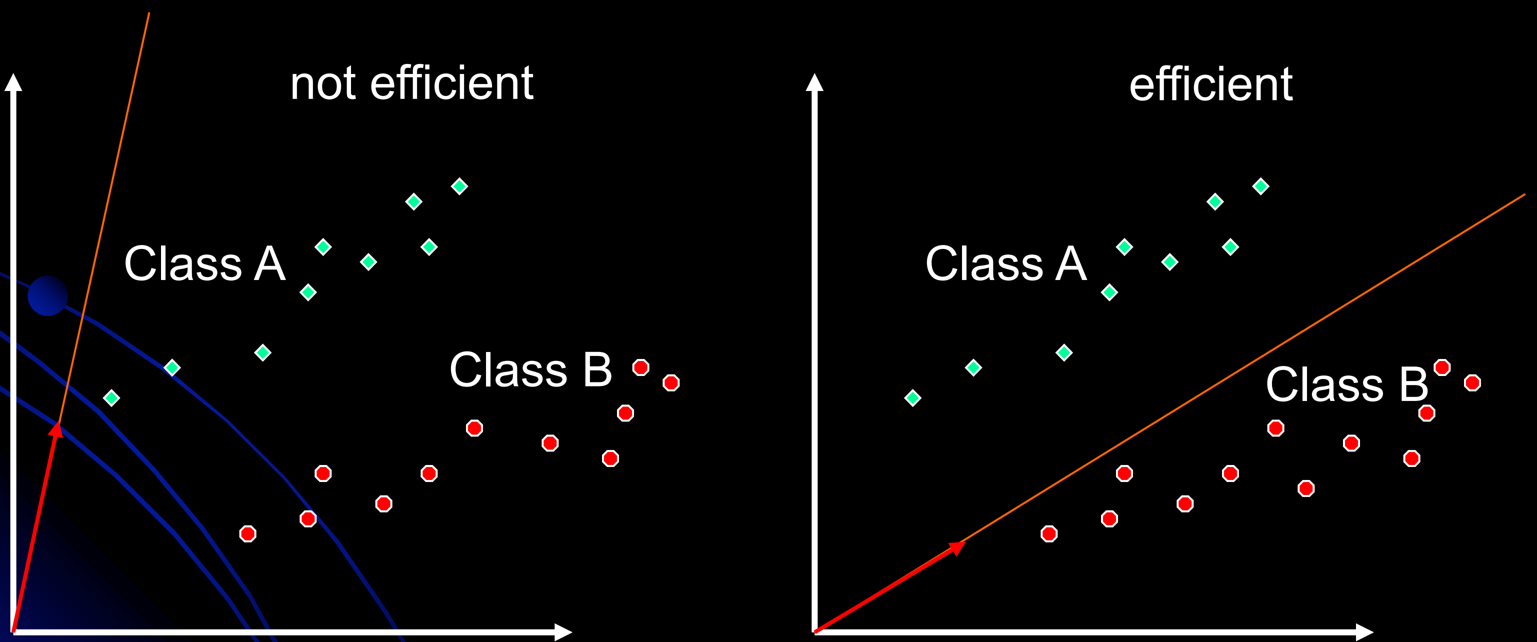
- We will introduce
  - Eigenfaces
  - Fisherfaces
  - Elastic Bunch-Graph Matching

Eigenfaces

- Developed in 1991 by M.Turk
- Based on Principal Component Analysis (PCA)
- Relatively simple
- Fast
- Robust

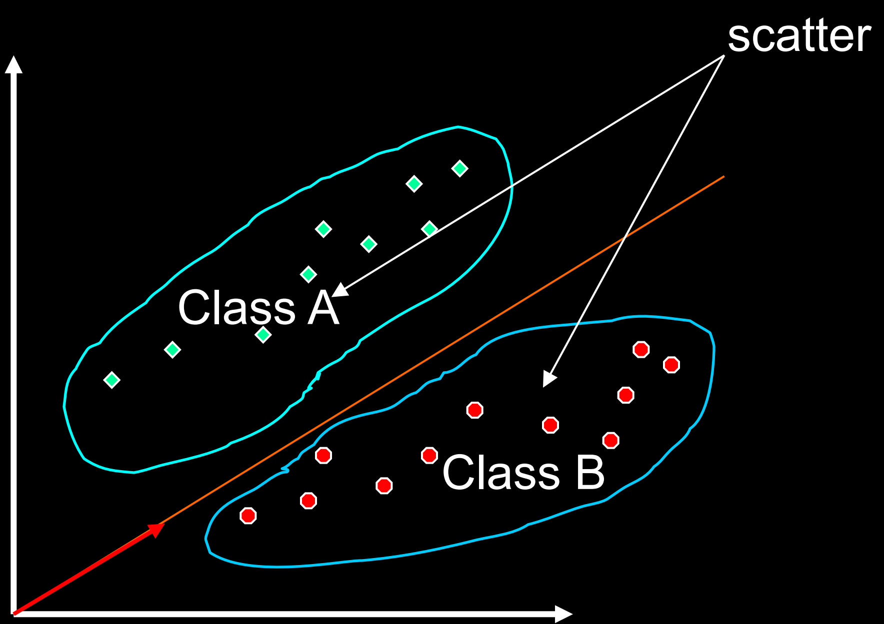
Eigenfaces

- PCA seeks directions that are efficient for representing the data



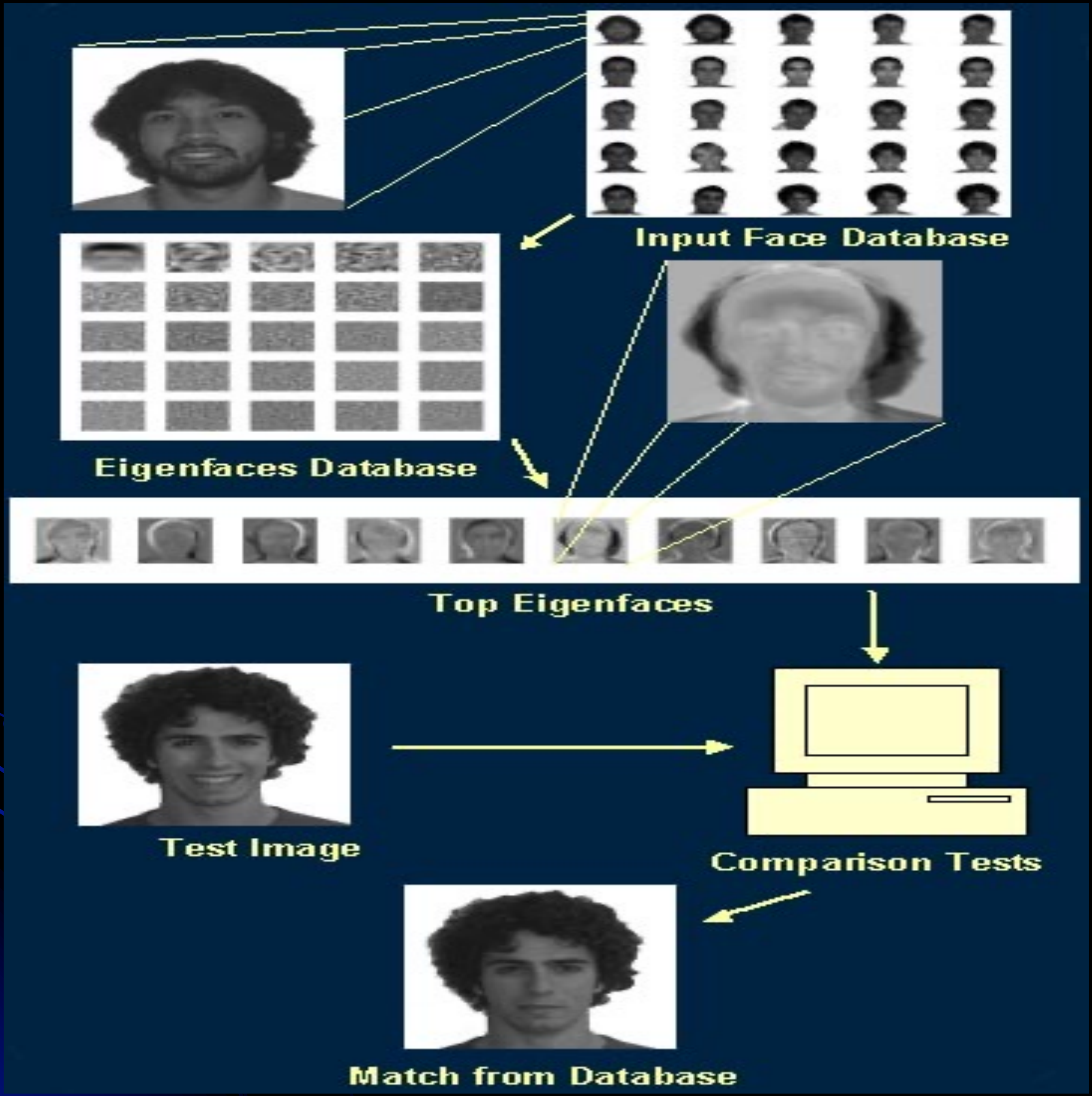
Eigenfaces

- PCA maximizes the total scatter



Eigenfaces

- PCA reduces the dimension of the data
- Speeds up the computational time


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

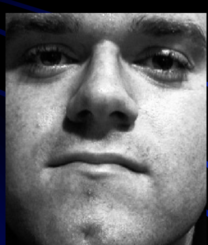
- The database

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

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

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

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



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

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

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

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

Eigenfaces, the algorithm

- We compute the average face

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

Eigenfaces, the algorithm

- Then subtract it from the training faces

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

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

Eigenfaces, the algorithm

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

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

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

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

Eigenfaces, the algorithm

- 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

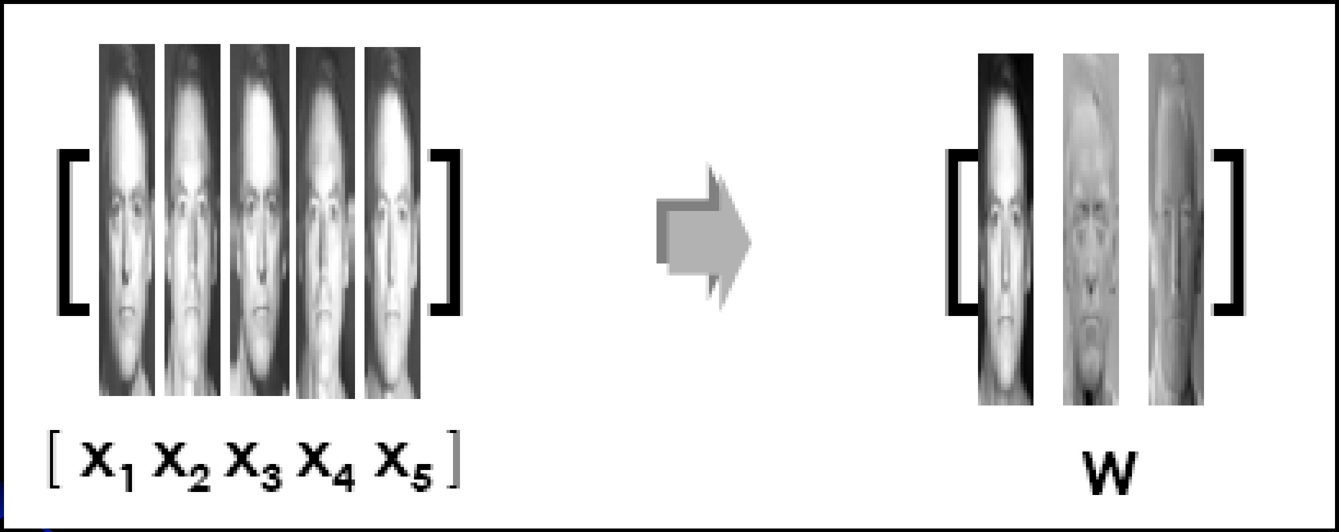
$$U = AV$$

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

Eigenfaces, the algorithm



A: collection of the training faces

U: Face Space / Eigen Space

Eigenfaces

- Eigenface of original faces

Eigenfaces, the algorithm

- Compute for each face its projection onto the face space

$$\Omega_1 = U^{\mathrm{T}}\left(\vec{a}_m\right),\quad \Omega_2 = U^{\mathrm{T}}\left(\vec{b}_m\right),\quad \Omega_3 = U^{\mathrm{T}}\left(\vec{c}_m\right),\quad \Omega_4 = U^{\mathrm{T}}\left(\vec{d}_m\right),$$

$$\Omega_5 = U^{\mathrm{T}}\left(\vec{e}_m\right),\quad \Omega_6 = U^{\mathrm{T}}\left(\vec{f}_m\right),\quad \Omega_7 = U^{\mathrm{T}}\left(\vec{g}_m\right),\quad \Omega_8 = U^{\mathrm{T}}\left(\vec{h}_m\right)$$

- Compute the threshold

$$\theta = \frac{1}{2}\max\left\{\left\|\Omega_i - \Omega_j\right\|\right\}\ for\ i, j = 1..M$$

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

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

Eigenfaces, the algorithm

- Compute its projection onto the face space U

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

- Compute the distance in the face space between the face and all known faces

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

Eigenfaces, the algorithm

- Reconstruct the face from eigenfaces

$$\vec{s} = U\Omega$$

- Compute the distance between the face and its reconstruction

$$\xi^2 = \left\| \vec{r}_m - \vec{s} \right\|^2$$

Eigenfaces, the algorithm

- Distinguish between
  - If $\xi \geq \theta$ then it's not a face; the distance between the face and its reconstruction is larger than threshold
  - If $\xi < \theta \; and \; \min\{\varepsilon_i\} < \theta$ then it's a new face
  - If $\xi < \theta \; and \; \varepsilon_i \geq \theta, (i = 1..M)$ then it's a known face because the distance in the face space between the face and all known faces is larger than threshold

Eigenfaces, the algorithm

- Problems with eigenfaces
  - Different illumination

"The variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity."

-- Moses, Adini, Ullman, ECCV '94

Eigenfaces, the algorithm

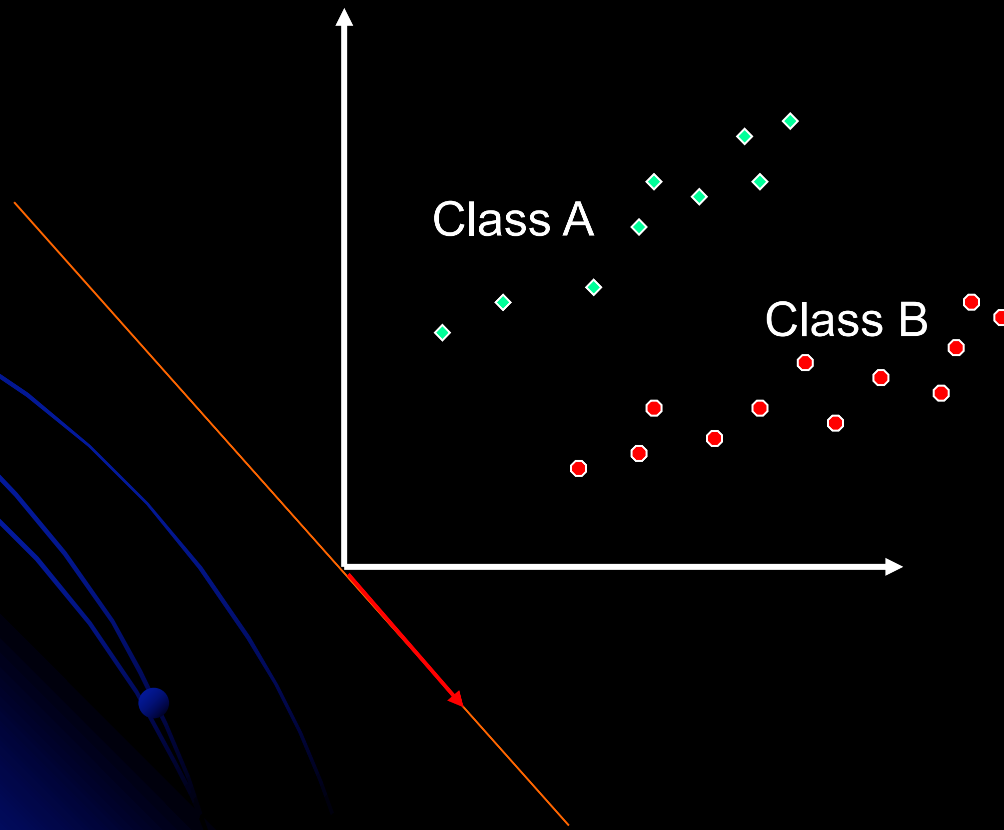
- Problems with eigenfaces
  - Different head pose
  - Different alignment
  - Different facial expression

Fisherfaces

- Developed in 1997 by P.Belhumeur et al.
- Based on Fisher's Linear Discriminant Analysis (LDA)
- Faster than eigenfaces, in some cases
- Has lower error rates
- Works well even if different illumination
- Works well even if different facial express.

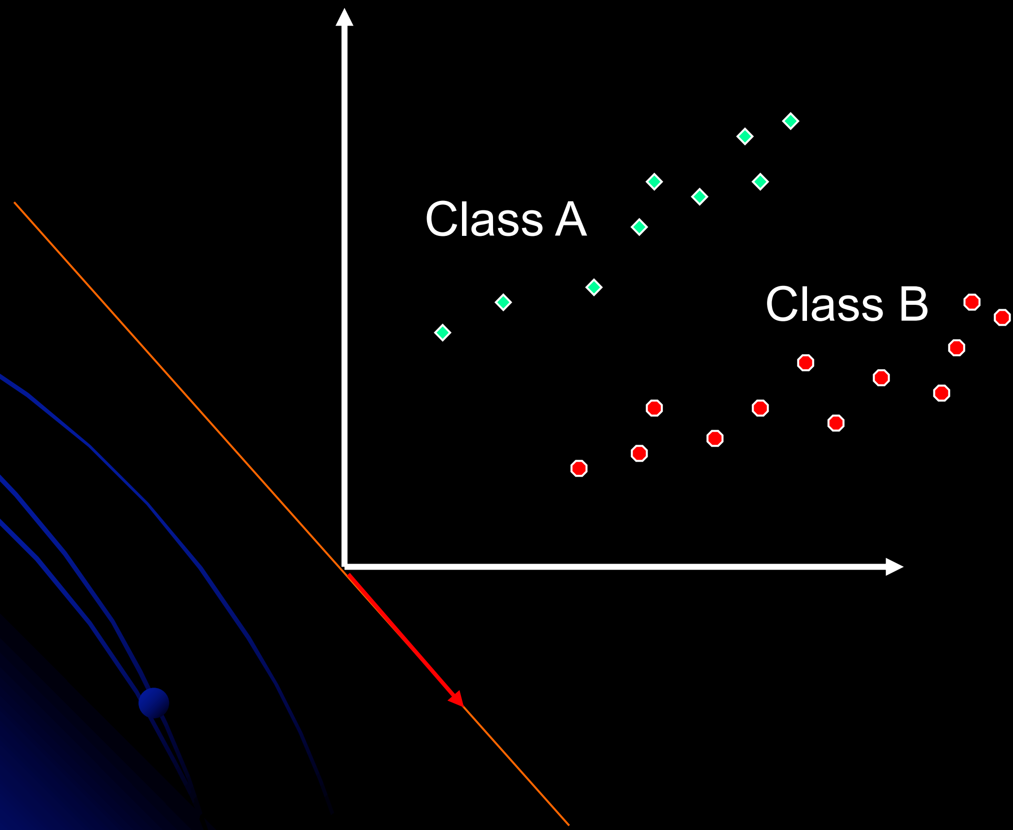
Fisherfaces

- LDA seeks directions that are efficient for discrimination between the data



Fisherfaces

- LDA maximizes the between-class scatter
- LDA minimizes the within-class scatter



Fisherfaces, 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

Fisherfaces, the algorithm

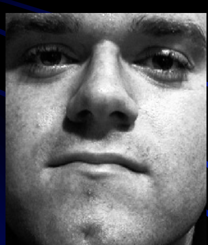
- The database


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


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


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


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


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


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


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


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

Fisherfaces, the algorithm

- We compute the average of all faces

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

Fisherfaces, the algorithm

- Compute the average face of each person

$$\vec{x} = \frac{1}{2}\begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_{N^2} + b_{N^2} \end{pmatrix}, \quad \vec{y} = \frac{1}{2}\begin{pmatrix} c_1 + d_1 \\ c_2 + d_2 \\ \vdots \\ c_{N^2} + d_{N^2} \end{pmatrix},$$

$$\vec{z} = \frac{1}{2}\begin{pmatrix} e_1 + f_1 \\ e_2 + f_2 \\ \vdots \\ e_{N^2} + f_{N^2} \end{pmatrix}, \quad \vec{w} = \frac{1}{2}\begin{pmatrix} g_1 + h_1 \\ g_2 + h_2 \\ \vdots \\ g_{N^2} + h_{N^2} \end{pmatrix}$$

Fisherfaces, the algorithm

- And subtract them from the training faces

$$\vec{a}_m = \begin{pmatrix} a_1 & - & x_1 \\ a_2 & - & x_2 \\ \vdots & & \vdots \\ a_{N^2} & - & x_{N^2} \end{pmatrix}, \quad \vec{b}_m = \begin{pmatrix} b_1 & - & x_1 \\ b_2 & - & x_2 \\ \vdots & & \vdots \\ b_{N^2} & - & x_{N^2} \end{pmatrix}, \quad \vec{c}_m = \begin{pmatrix} c_1 & - & y_1 \\ c_2 & - & y_2 \\ \vdots & & \vdots \\ c_{N^2} & - & y_{N^2} \end{pmatrix}, \quad \vec{d}_m = \begin{pmatrix} d_1 & - & y_1 \\ d_2 & - & y_2 \\ \vdots & & \vdots \\ d_{N^2} & - & y_{N^2} \end{pmatrix},$$

$$\vec{e}_m = \begin{pmatrix} e_1 & - & z_1 \\ e_2 & - & z_2 \\ \vdots & & \vdots \\ e_{N^2} & - & z_{N^2} \end{pmatrix}, \quad \vec{f}_m = \begin{pmatrix} f_1 & - & z_1 \\ f_2 & - & z_2 \\ \vdots & & \vdots \\ f_{N^2} & - & z_{N^2} \end{pmatrix}, \quad \vec{g}_m = \begin{pmatrix} g_1 & - & w_1 \\ g_2 & - & w_2 \\ \vdots & & \vdots \\ g_{N^2} & - & w_{N^2} \end{pmatrix}, \quad \vec{h}_m = \begin{pmatrix} h_1 & - & w_1 \\ h_2 & - & w_2 \\ \vdots & & \vdots \\ h_{N^2} & - & w_{N^2} \end{pmatrix}$$

Fisherfaces, the algorithm

- We build scatter matrices S_1, S_2, S_3, S_4

$$S_1 = \left(\vec{a}_m\vec{a}_m^{\mathrm{T}} + \vec{b}_m\vec{b}_m^{\mathrm{T}}\right), S_2 = \left(\vec{c}_m\vec{c}_m^{\mathrm{T}} + \vec{d}_m\vec{d}_m^{\mathrm{T}}\right),$$
$$S_3 = \left(\vec{e}_m\vec{e}_m^{\mathrm{T}} + \vec{f}_m\vec{f}_m^{\mathrm{T}}\right), S_4 = \left(\vec{g}_m\vec{g}_m^{\mathrm{T}} + \vec{h}_m\vec{h}_m^{\mathrm{T}}\right)$$

- And the within-class scatter matrix S_W

$$S_W = S_1 + S_2 + S_3 + S_4$$

Fisherfaces, the algorithm

- The between-class scatter matrix

$$S_B = 2(\vec{x}-\vec{m})(\vec{x}-\vec{m})^{\mathrm{T}} + 2(\vec{y}-\vec{m})(\vec{y}-\vec{m})^{\mathrm{T}} + 2(\vec{z}-\vec{m})(\vec{z}-\vec{m})^{\mathrm{T}} + 2(\vec{w}-\vec{m})(\vec{w}-\vec{m})^{\mathrm{T}}$$

- We are seeking the matrix W maximizing

$$J(W) = \frac{\left|W^{\mathrm{T}} S_B W\right|}{\left|W^{\mathrm{T}} S_W W\right|}$$

Fisherfaces, the algorithm

If S_W is nonsingular ($M \geq N^2$):

- Columns of W are eigenvectors of $S_W^{-1} S_B$
  - We have to compute the inverse of S_W
  - We have to multiply the matrices
  - We have to compute the eigenvectors

Fisherfaces, the algorithm

If S_W is nonsingular ($M \geq N^2$):

- Simpler:
  - Columns of W are eigenvectors satisfying

$$S_B w_i = \lambda_i S_W w_i$$

  - The eigenvalues are roots of

$$\left| S_B - \lambda_i S_W \right| = 0$$

  - Get eigenvectors by solving

$$\left(S_B - \lambda_i S_W \right) w_i = 0$$

Fisherfaces, the algorithm

If S_W is singular ($M < N^2$):

- Apply PCA first
  - Will reduce the dimension of faces from N^2 to M
  - There are M M-dimensional vectors

- Apply LDA as described

Fisherfaces, the algorithm

- Project faces onto the LDA-space

$$\vec{x}_{LDA} = W^{\mathrm{T}}\vec{x}\,, \quad \vec{y}_{LDA} = W^{\mathrm{T}}\vec{y}\,,$$
$$\vec{z}_{LDA} = W^{\mathrm{T}}\vec{z}\,, \quad \vec{w}_{LDA} = W^{\mathrm{T}}\vec{w}$$

- To classify the face
  - Project it onto the LDA-space
  - Run a nearest-neighbor classifier

Fisherfaces, the algorithm

- Problems
  - Small databases
  - The face to classify must be in the DB

PCA & Fisher's Linear Discriminant

- Between-class scatter

$$S_B = \sum_{i=1}^{c} |\chi_i| (\mu_i - \mu)(\mu_i - \mu)^T$$

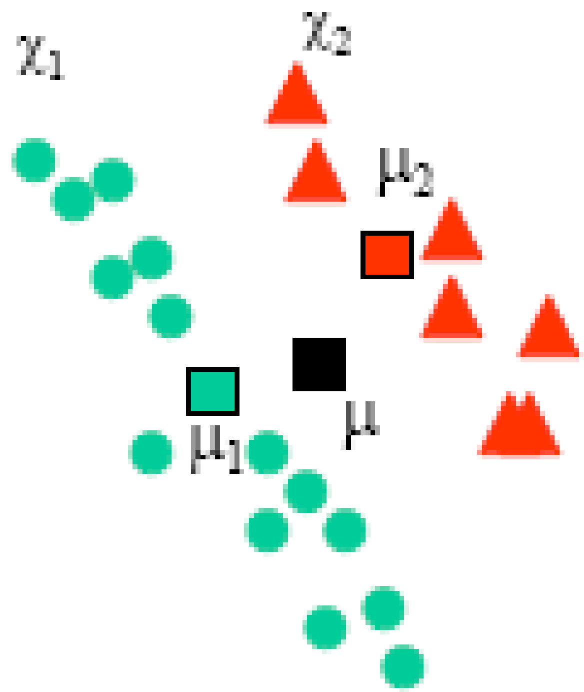
- Within-class scatter

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

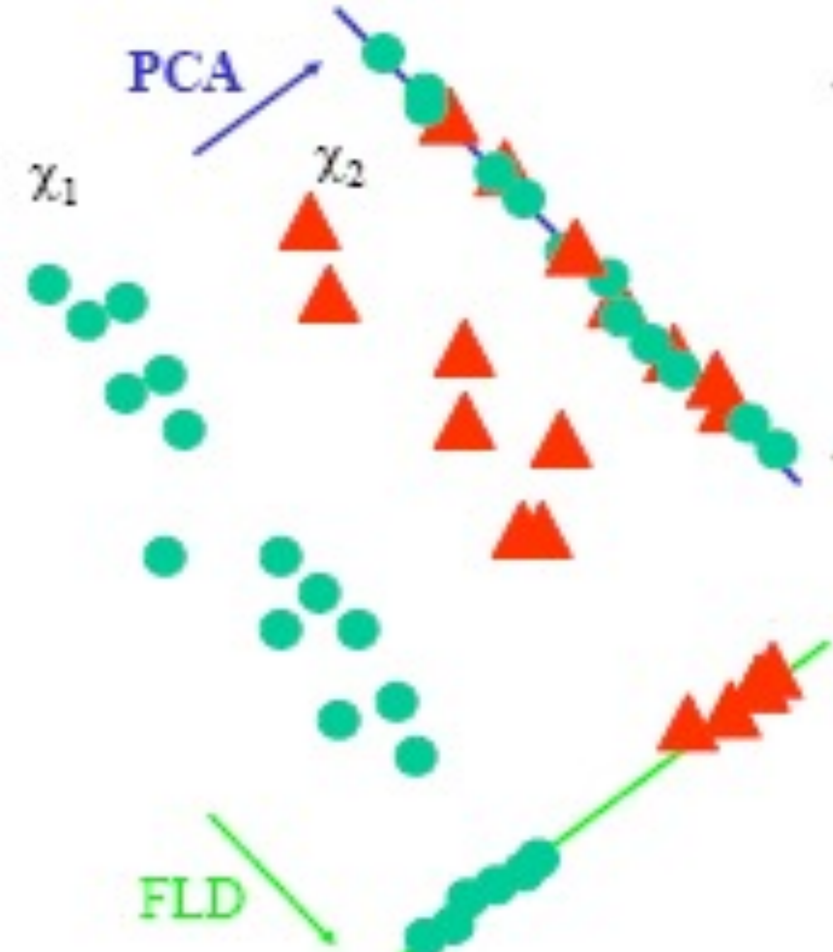
- Total scatter

$$S_T = \sum_{i=1}^{c} \sum_{x_k \in \chi_i} (x_k - \mu)(x_k - \mu)^T = S_B + S_W$$

- Where
  - c is the number of classes
  - μ_i is the mean of class χ_i
  - $|\chi_i|$ is number of samples of χ_i.



PCA & Fisher’s Linear Discriminant



- PCA (Eigenfaces)

$$\boxed{W_{PCA} = \arg\max_{W} \left| W^T S_T W \right|}$$

Maximizes projected total scatter

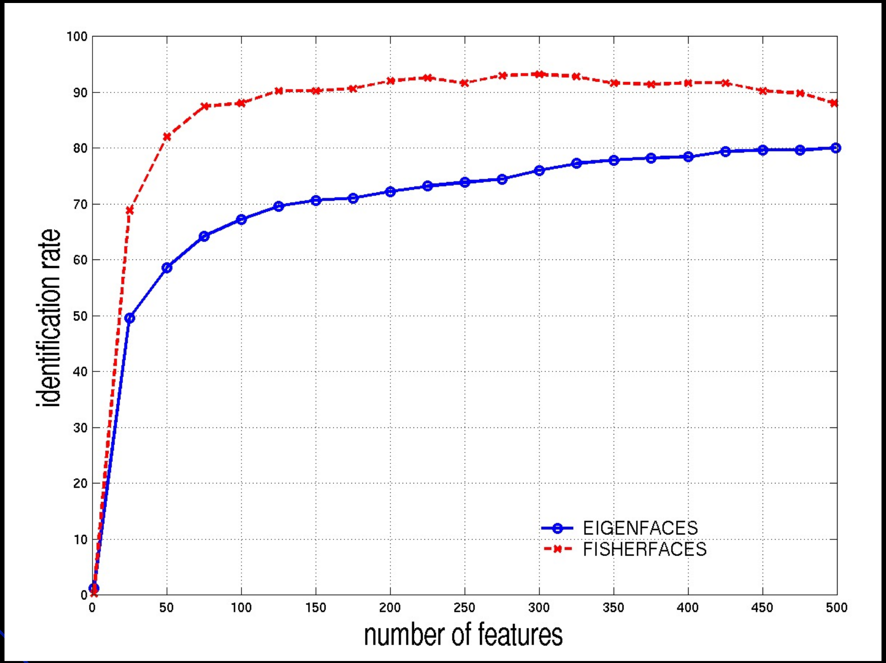
- Fisher’s Linear Discriminant

$$\boxed{W_{fld} = \arg\max_{W} \frac{\left| W^T S_B W \right|}{\left| W^T S_W W \right|}}$$

Maximizes ratio of projected between-class to projected within-class scatter

Comparison

- FERET database



best ID rate: eigenfaces 80.0%, fisherfaces 93.2%

Comparison

- Eigenfaces
  - project faces onto a lower dimensional sub-space
  - no distinction between inter- and intra-class variabilities
  - optimal for representation but not for discrimination

Comparison

- Fisherfaces
  - find a sub-space which maximizes the ratio of inter-class and intra-class variability
  - same intra-class variability for all classes

Local Feature Analysis

-- Elastic Bunch-Graph Matching

Face Features

- Facial recognition utilizes distinctive features of the face – including: distinct micro elements like:
  - Mouth, Nose, Eye, Cheekbones, Chin, Lips, Forehead, Ears
- Upper outlines of the eye sockets, the areas surrounding the cheekbones, the sides of the mouth, and the location of the nose and eyes.
- The distance between the eyes, the length of the nose, and the angle of the jaw.

Face Features

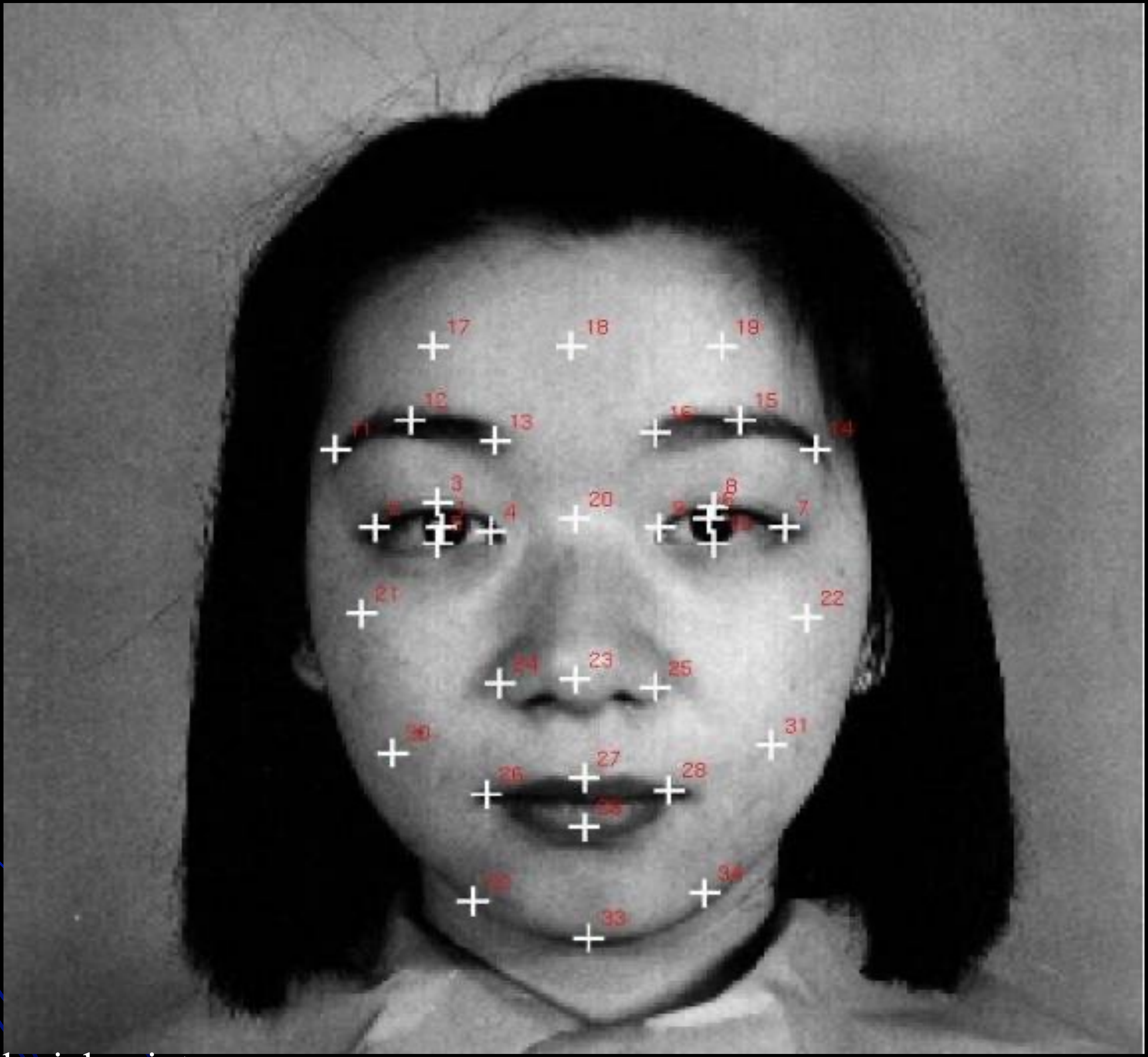
- Some technologies do not utilize areas of the face located near the hairline, so they are somewhat resistant to moderate changes in hairstyle.
- When used in identification mode, facial recognition technology generally returns candidate lists of close matches as opposed to returning a single definitive match as does fingerprint and iris-scan.
- The file containing facial micro features is called a "template."
- Using templates, the software then compares that image with another image and produces a score that measures how similar the images are to each other.

Face Features

- Typical sources of images for use in facial recognition include video camera signals and pre-existing photos such as those in driver's license databases.
  including:
  - Distance between the micro elements
  - A reference feature
  - Size of the micro element
  - Amount of head radiated from the face (unseen by human eye). Heat can be measured using an infrared camera.

A face recognition based on local feature analysis

- A face is represented as a graph, whose nodes, positioned in correspondence to the facial fiducial points.
  - A fiducial point is a point or line on a scale used for reference or comparison purposes.
- A face recognition system uses an automatic approach to localize the facial fiducial points.
- It then determines the head pose and compares the face with the gallery images.
- This approach is invariant to rotation, light and scale.



A template for the 34 fiducial points on a face image:

EBGM

- **Elastic Bunch-Graph Matching (**EBGM) algorithm locates landmarks on an image, such as the eyes, nose, and mouth.
- Gabor jets are extracted from each landmark and are used to form a face graph for each image. A face graph serves the same function as the projected vectors in the PCA or LDA algorithm; they represent the image in a low dimensional space.
- After a face graph has been created for each test image, the algorithm measures the similarity of the face graphs.
- Paper:http://www.snl.salk.edu/~fellous/posters/Bu97poster/BUPoster.pdf

Summary

- Three algorithms have been introduced
  - Eigenfaces
    - Reduce the dimension of the data from N^2 to M
    - Verify if the image is a face at all
    - Allow online training
    - Fast recognition of faces
    - Problems with illumination, head pose etc

Summary

- Fisherfaces
  - Reduce dimension of the data from N^2 to $P-1$
  - Can outperform eigenfaces on a representative DB
  - Works also with various illuminations etc
  - Can only classify a face which is "known" to DB

Summary

- **Elastic Bunch-Graph Matching**
  - Reduce the dimension of the data from N^2 to M
  - Recognize face with different poses
  - Recognize face with different expressions

References

[1] M. Turk, A. Pentland, “Face Recognition Using Eigenfaces”

[2] J. Ashbourn, Avanti, V. Bruce, A. Young, ”Face Recognition Based on Symmetrization and Eigenfaces”

[3] http://www.markus-hofmann.de/eigen.html

[4] P. Belhumeur, J. Hespanha, D. Kriegman, “Eigenfaces vs Fisherfaces: Recognition using Class Specific Linear Projection”

[5] R. Duda, P. Hart, D. Stork, “Pattern Classification”, ISBN 0-471-05669-3, pp. 121-124

[6] F. Perronin, J.-L. Dugelay, “Deformable Face Mapping For Person Identification”, ICIP 2003, Barcelona

[7] B. Moghaddam, C. Nastar, and A. Pentland. A bayesian similarity measure for direct image matching. ICPR, B:350–358, 1996.

http://www.face-rec.org/interesting-papers/

Wednesday (Nov. 17th)

- **Present one of the following algorithms**
  - **Elastic Bunch-Graph Matching (**EBGM) algorithm
  - Bayesian Intrapersonal/Extrapersonal Classifier, or
  - One from http://www.face-rec.org/interesting-papers/
- Hands-on Lab of Face Biometrics
  - http://www.cs.colostate.edu/evalfacerec/
  - User Guide