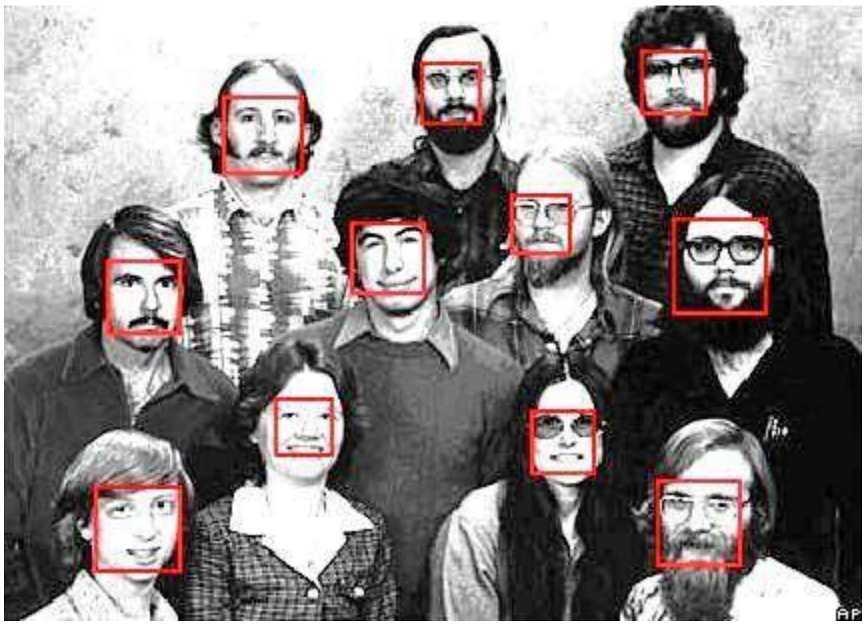
Image Analysis: Face Detection and Recognition

UNIVERSITY OF WASHINGTON
ELECTRICAL ENGINEERING

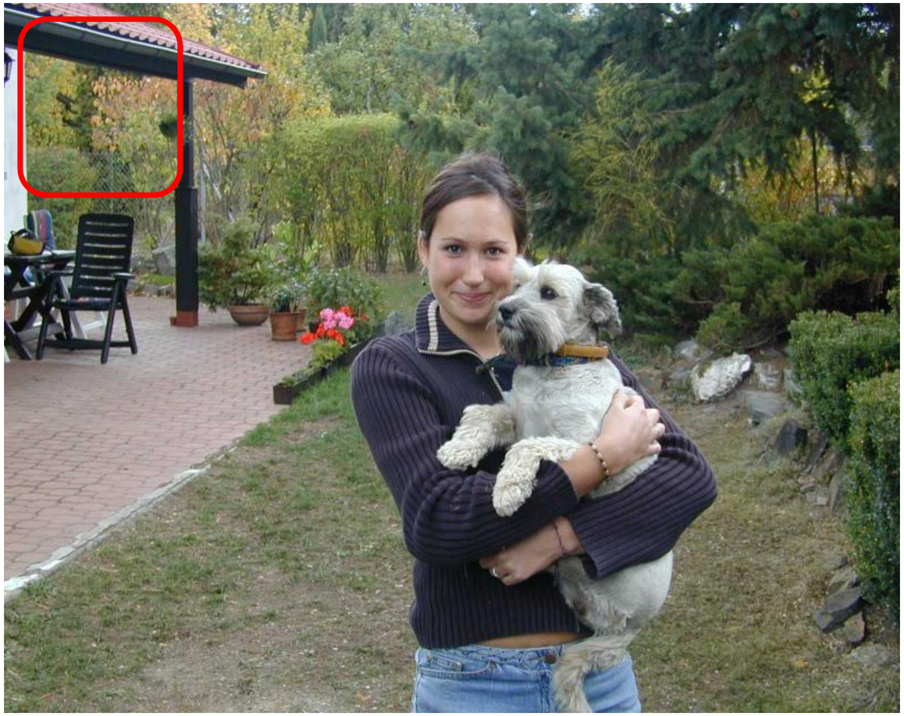
Face detection and recognition



Many slides adapted from K. Grauman and D. Lowe

Face detection and recognition

Face detection



Challenges of face detection

- Sliding window detector must evaluate tens of thousands of location/scale combinations
  - This evaluation must be made as efficient as possible
- Faces are rare: 0–10 per image
  - At least 1000 times as many non-face windows as face windows
  - This means that the false positive rate must be extremely low
  - Also, we should try to spend as little time as possible on the non-face windows

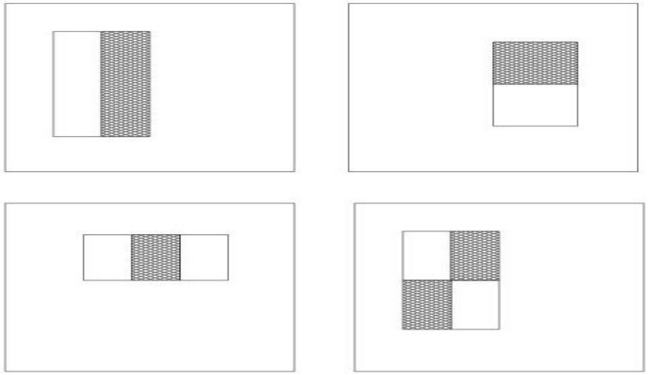
The Viola/Jones Face Detector

- A “paradigmatic” method for real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

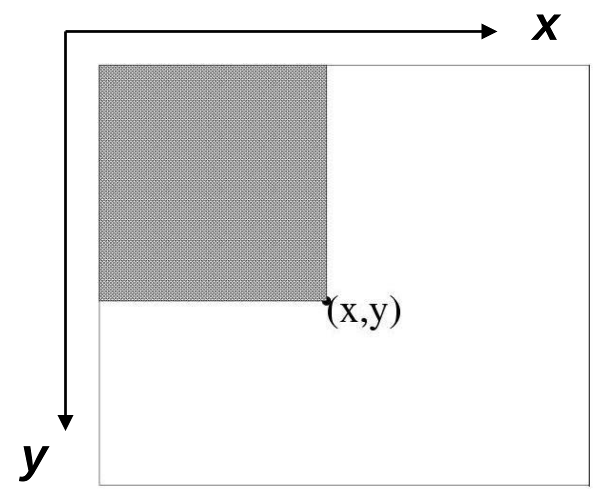
Features

- Can a simple (rectangular) feature (filter) indicate the existence of a face?
- All faces share some similar properties
  - The eyes region is darker than the upper-cheeks.
  - The nose bridge region is brighter than the eyes.
  - That is useful domain knowledge
- Need for encoding of Domain Knowledge:
  - ***Location - Size:*** eyes & nose bridge region
  - ***Value:*** darker / brighter (-1/+1 valued)



Integral Image Representation

- Given a detection resolution of 24x24 (smallest sub-window), the set of different rectangle features is ~160,000! (need fast speed)
- The Integral image can be computed in a single pass and only once for each sub-window!



formal definition:

$$ii(x,y) = \sum_{x' \le x,\, y' \le y} i(x', y')$$

Recursive definition:

$$s(x,y) = s(x, y-1) + i(x,y)$$

$$ii(x,y) = ii(x-1, y) + s(x,y)$$

0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

0	1	2	3
1	4	7	11
2	7	11	16
3	11	16	21

Computing sum within a rectangle

- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = A – B – C + D

- Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision

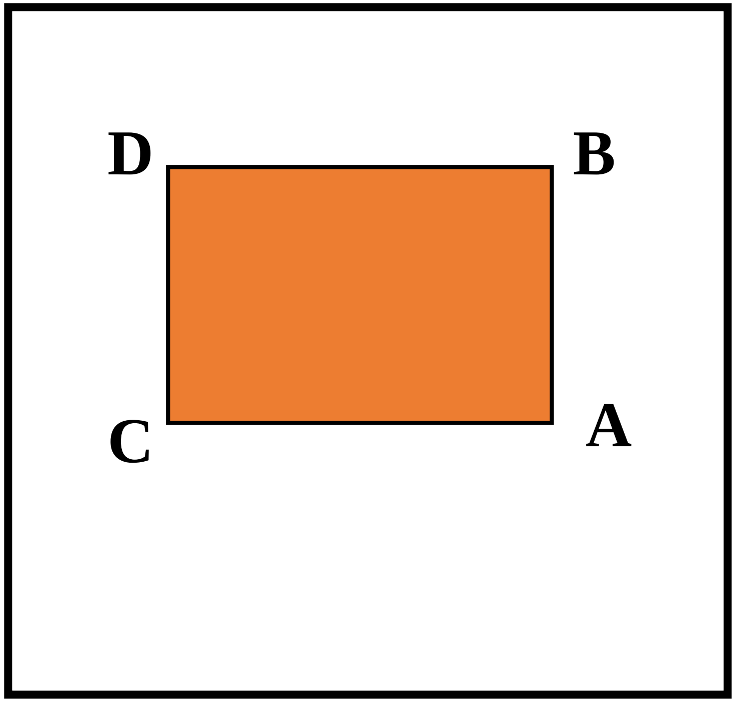
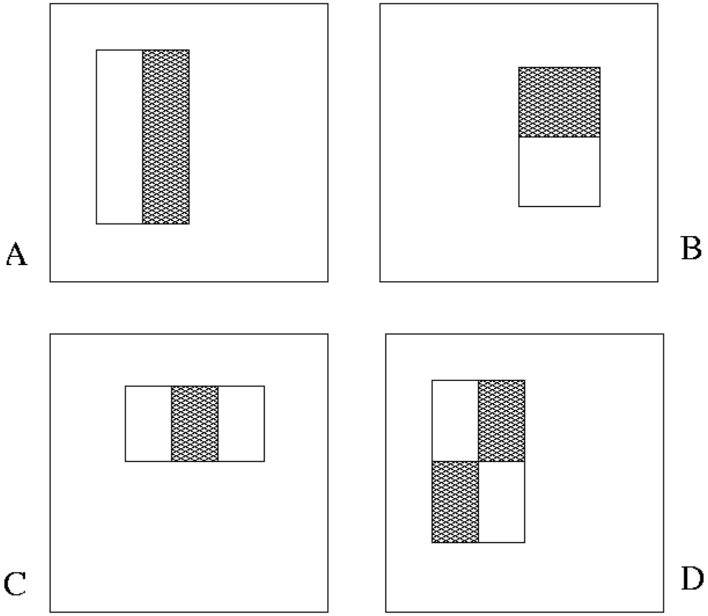


Image Features

"Rectangle filters"



Value =

∑ (pixels in white area) –
∑ (pixels in black area)

(2-D convolution)

AdaBoost

- Stands for "**Ada**ptive" **boost**
- Constructs a "strong" classifier as a linear combination of weighted simple "weak" classifiers (each one corresponds to a feature)

$$H(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \ldots..$$

Weak classifier

Strong classifier

Image

Weight

AdaBoost - *Characteristics*

- Features as weak classifiers
  - Each single rectangle feature may be regarded as a simple weak classifier
- An iterative algorithm
  - AdaBoost performs a series of trials, each time selecting a new weak classifier
  - Choose the most efficient (the one that best separates the examples – the lowest "weighted" error)
  - Choice of a classifier corresponds to choice of a feature
- Weights are being applied over the set of the example images
  - During each iteration, each example/image receives a weight determining its importance

Constructing the classifier

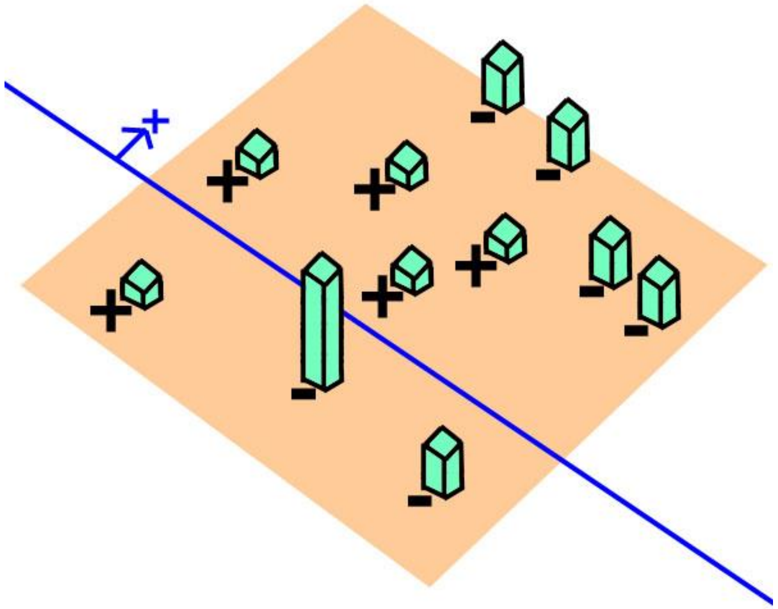
- For each round of boosting:
- Evaluate each rectangle filter on each example
- Sort examples by filter values
- Select best threshold for each filter (min error)
  - Use sorting to quickly scan for optimal threshold
- Select best filter/threshold combination

Adaboost Algorithm

- Given samples $(x_1,y_1),\ldots,(x_N,y_N)$, where $y_i \in \{-1,1\}$
- There are m positive samples, and l negative samples
- Initialize $\omega_{1,i}=1/N$
- For $t=1,\ldots,T$
  - Normalize $\omega_{t,i} = \omega_{t,i}/(\Sigma_j\, \omega_{t,j})$
  - Train the base learner (classifier) $h_t \in \{-1,1\}$ using distribution $\{\omega_{t,j}\}$
  - Choose h_t that minimize $\varepsilon_t = \min \Sigma_i\, \omega_{t,i}\, (1/2)|h_t(x_i) - y_i|$
  - Update $w_{t+1,i} = w_{t,i} \exp(-\alpha_t y_i h_t(x_i))$
  - where $\alpha_t = \frac{1}{2}\log(\frac{1+r_i}{1-r_i})$, where $r_i = \sum_{i=1}^{N} w_{t,i} y_i h_t(x_i)$
  - $\varepsilon_t > 0.5$, stop
- Output the final classifier $H(x) = sign(\, \Sigma_t\, \alpha_t\, h_t(x)\,)$

AdaBoost example

- AdaBoost starts with a uniform distribution of "weights" over training examples.
- Select the classifier with the lowest weighted error (i.e., a "weak" classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)



- At the end, make a linear combination of the weak classifiers obtained at all iterations.

$$h_{\text{strong}}(\mathbf{x}) = \begin{cases} 1 & \alpha_1 h_1(\mathbf{x}) + \ldots + \alpha_n h_n(\mathbf{x}) \geq \frac{1}{2}(\alpha_1 + \ldots + \alpha_n) \\ 0 & \text{otherwise} \end{cases}$$

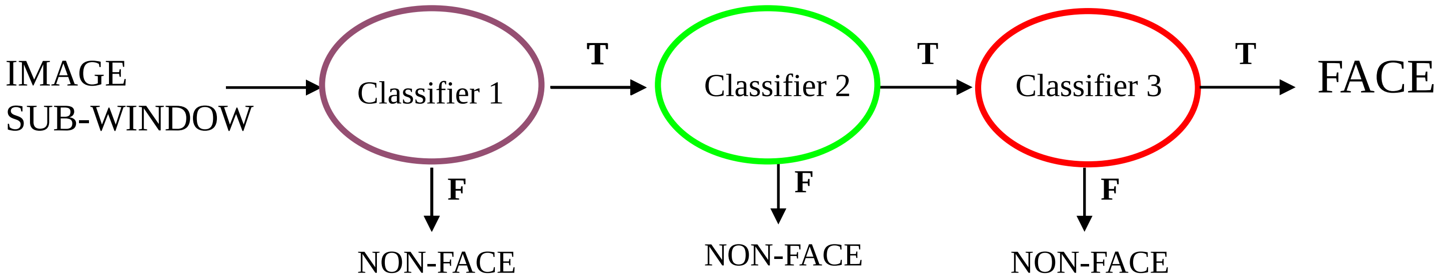
Boosting Summary

- Features are extracted from sub windows of a sample image.
  - The base size for a sub window is 24 by 24 pixels.
  - Each of the four feature types are scaled and shifted across all possible combinations
  - In a 24 pixel by 24 pixel sub window there are ~160,000 possible features to be calculated.
- Initially, give equal weight to each training example
- Iterative training procedure
  - Find best weak learner for current weighted training set
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is related to its accuracy)

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

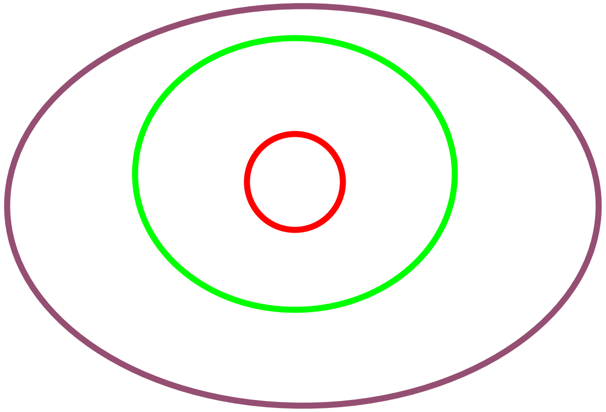
Cascading classifiers

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive results from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window

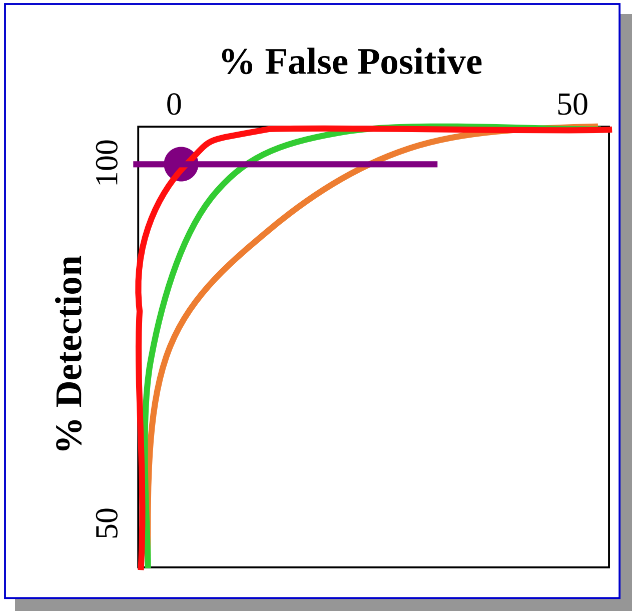


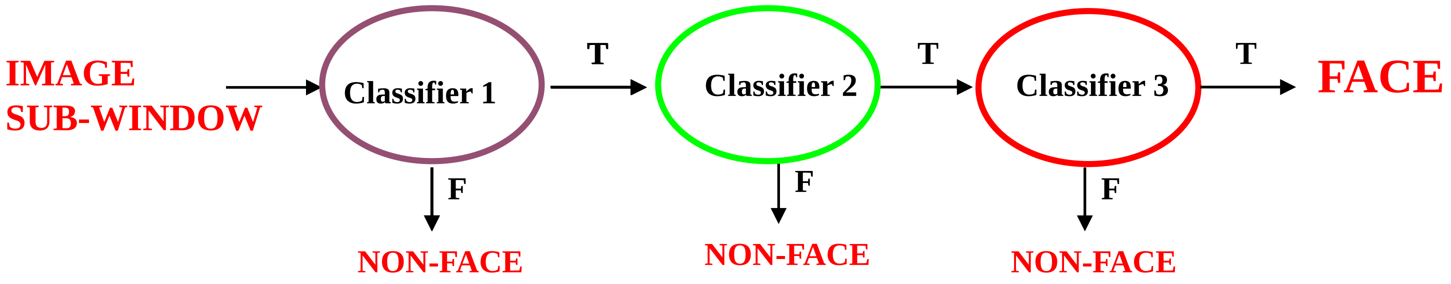
Cascading classifiers

- Chain classifiers that are progressively more complex and have lower false positive rates:



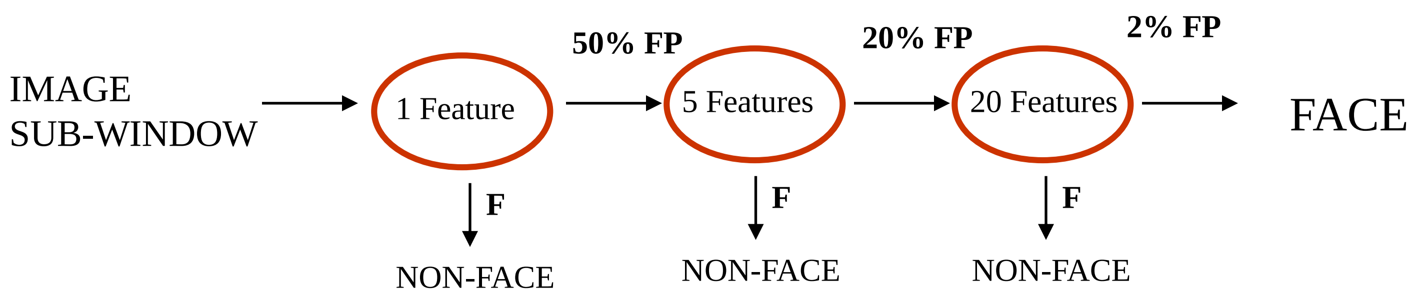
Receiver operating characteristic (ROC)



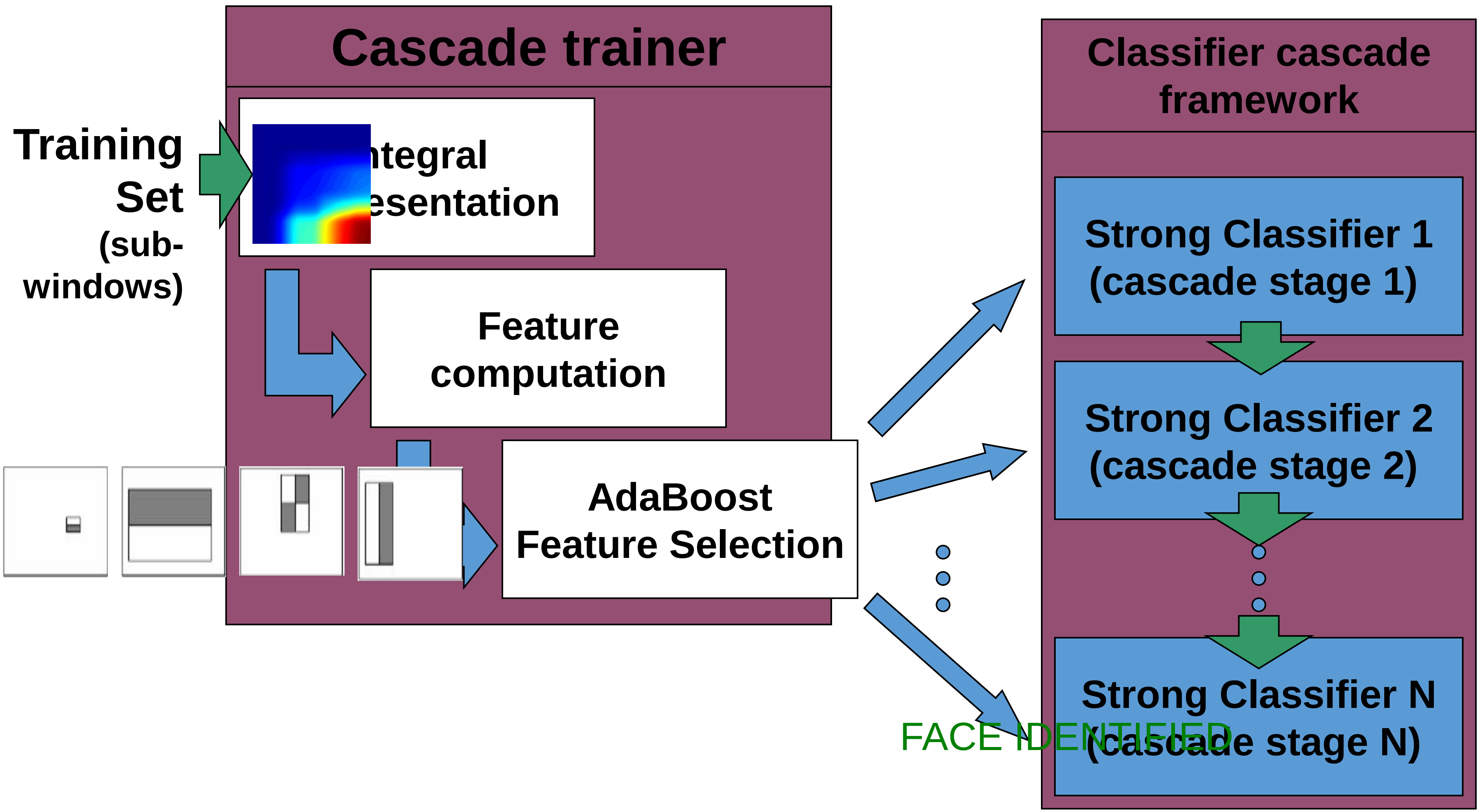


Training the cascade

- Adjust weak learner threshold to minimize *false positives (FP)* (as opposed to total classification error)
- Each classifier trained on false positives of previous stages
  - A single-feature classifier achieves 100% detection rate and about 50% false positive rate
  - A five-feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - A 20-feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)



Testing phase



The implemented system

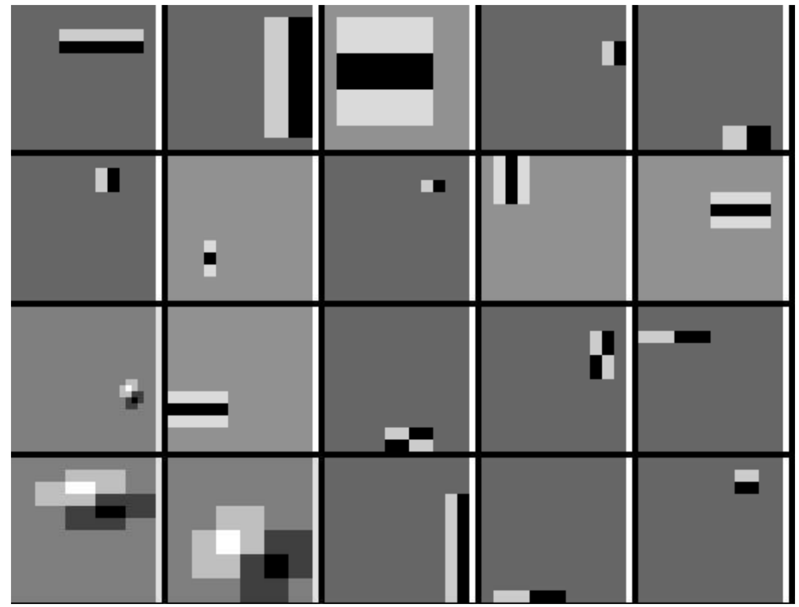
- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

(from Paul Viola)

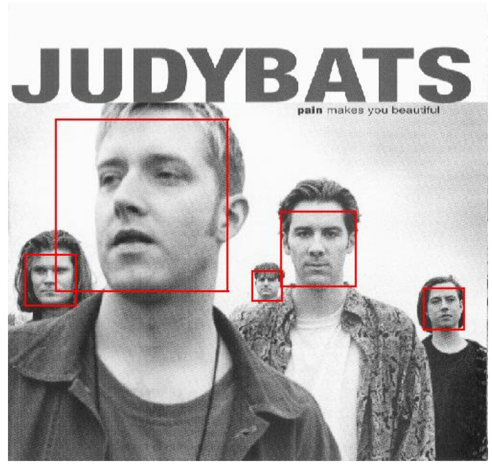
System performance

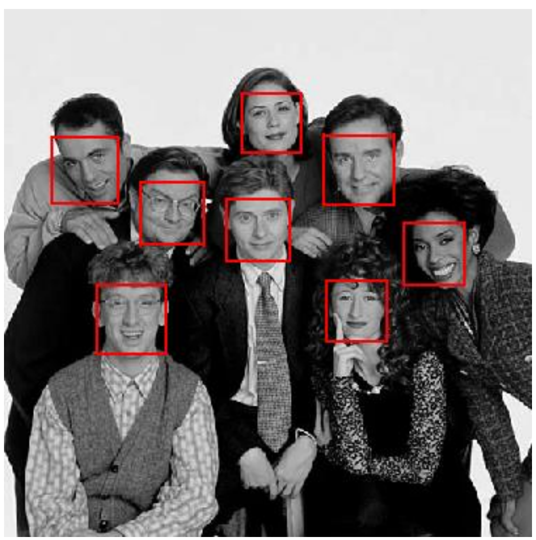
- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 MHz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

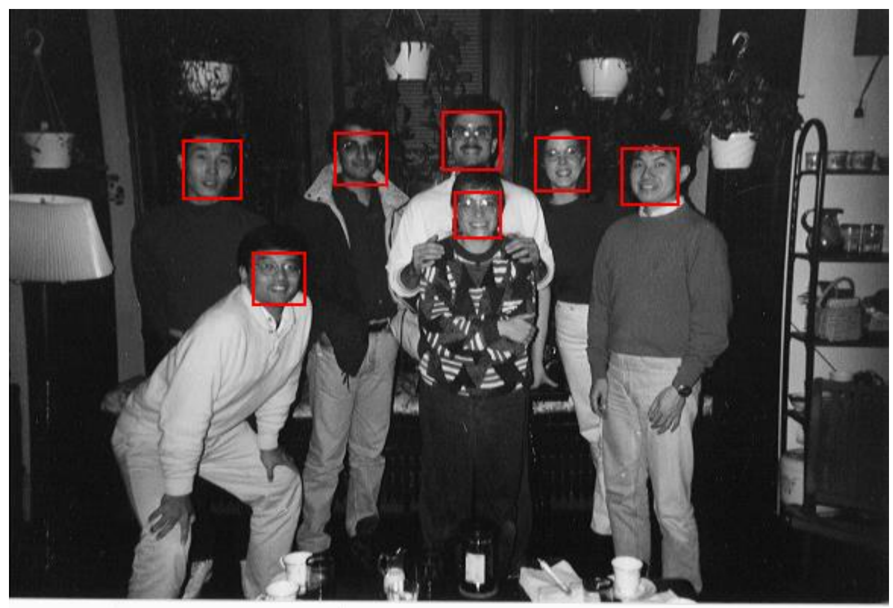
Profile Features



Output of Face Detector on Test Images

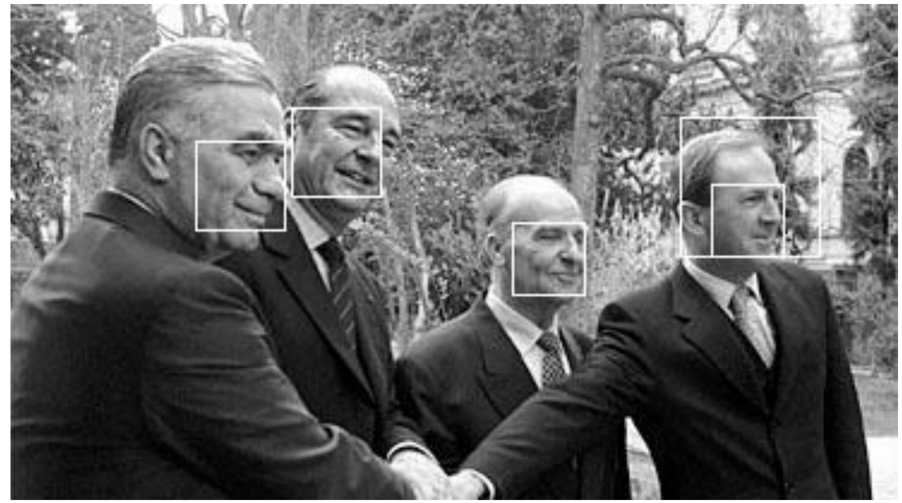






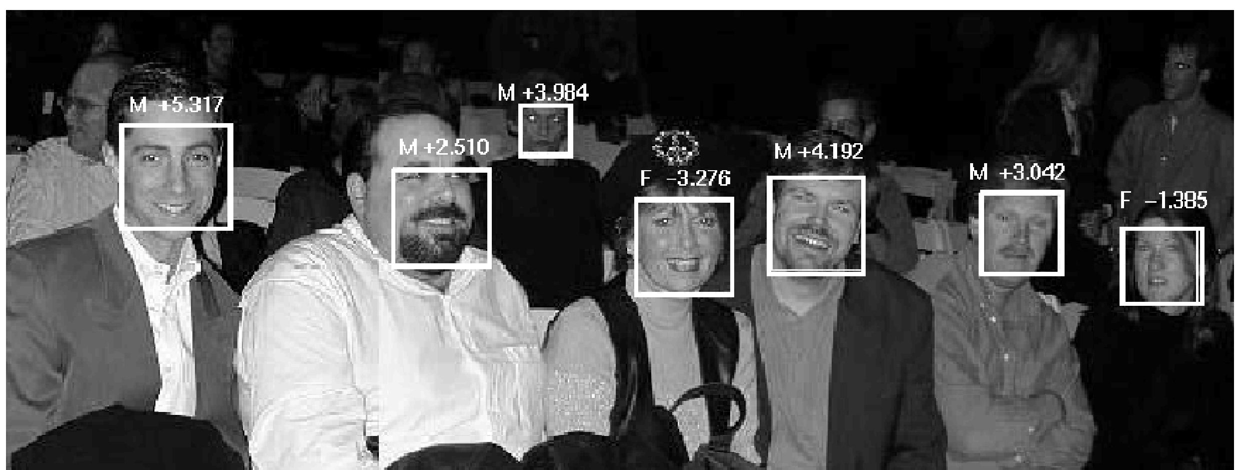
Other detection tasks

Facial Feature Localization



Profile Detection

Male vs. female



pros …

- Extremely fast feature computation
- Efficient feature selection
- Scale and location invariant detector
  - Instead of scaling the image itself (e.g. pyramid-filters), we scale the features.
- Such a generic detection scheme can be trained for detection of other types of objects (e.g., cars, hands)

… and cons

- Detector is most effective only on frontal images of faces
  - can hardly cope with 45^{o} face rotation
- Sensitive to lighting conditions
- We might get multiple detections of the same face, due to overlapping sub-windows.

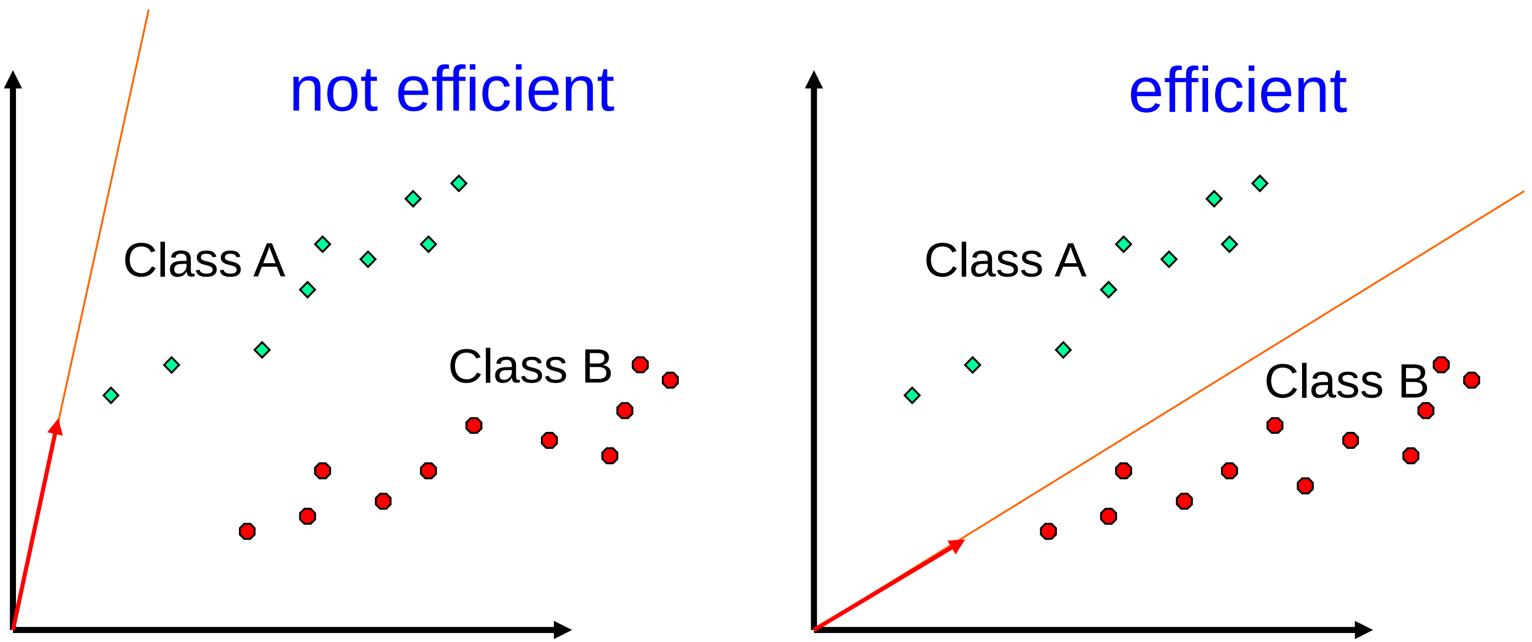
Face Recognition

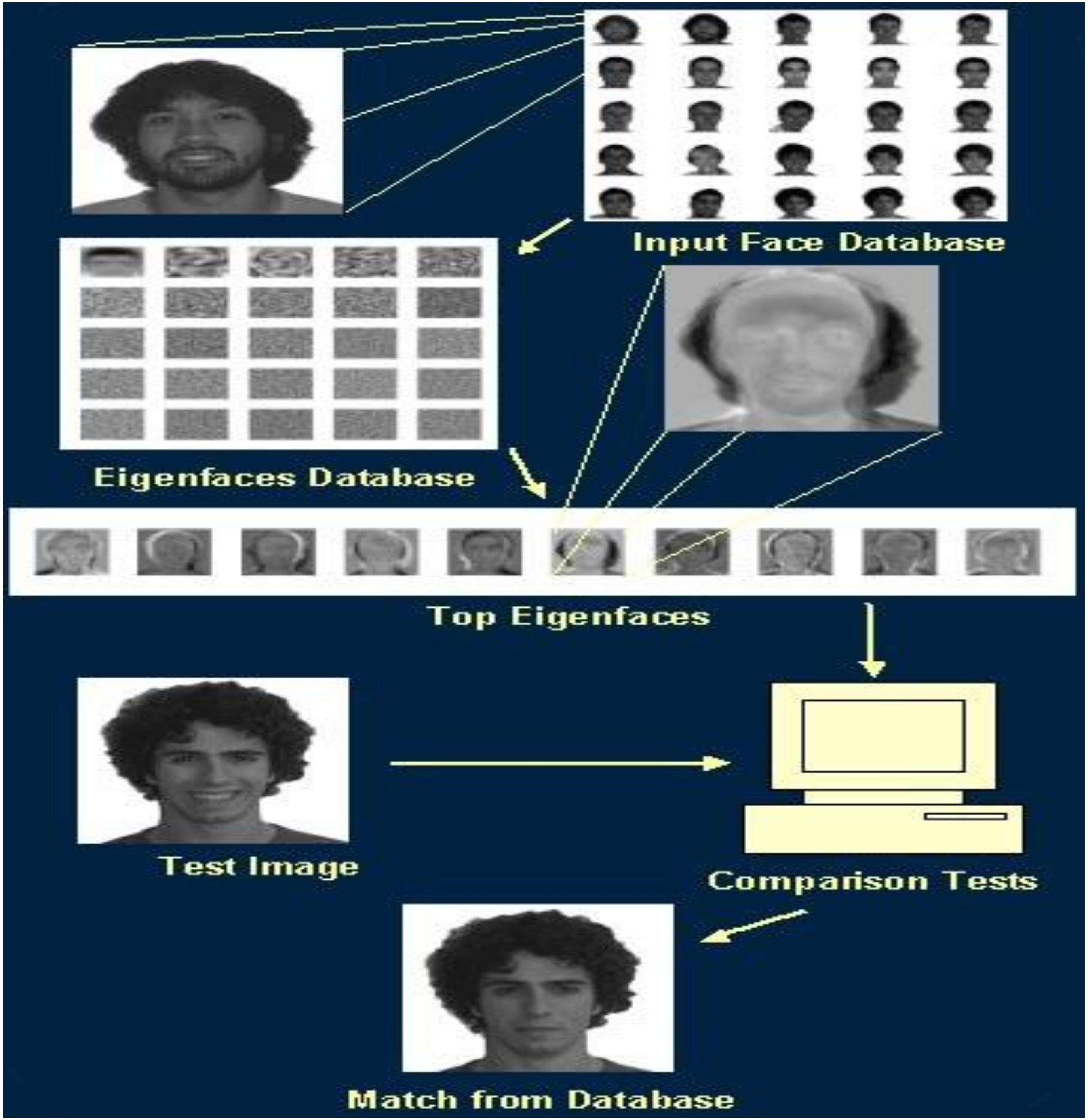
- Eigenfaces
- Fisherfaces

Ref: CPSC 4600/5600 Course: Biometrics and Cryptography, The University of Tennessee at Chattanooga

Eigenface from PCA

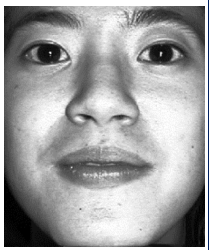
- Developed in 1991 by M.Turk, based on Principal Component Analysis (PCA)
- PCA seeks directions (reduced dimensions) that are efficient for representing the data -- maximizes the total scatter

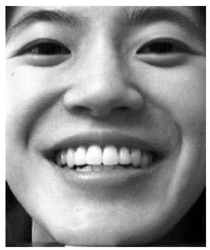


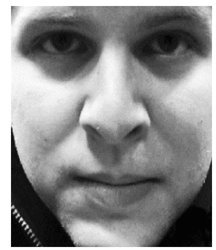

Input Face Database
Eigenfaces Database
Top Eigenfaces
Test Image
Comparison Tests
Match from Database

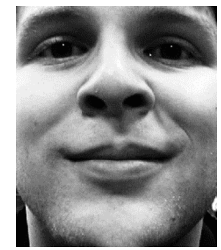
Eigenfaces, the algorithm

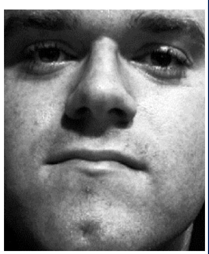
- The database

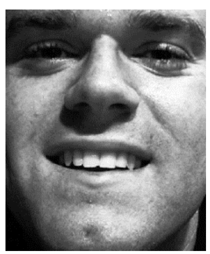

$$= \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{N^2} \end{pmatrix}$$

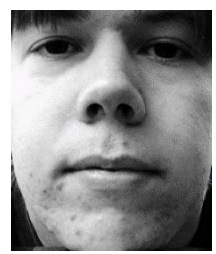

$$= \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N^2} \end{pmatrix}$$

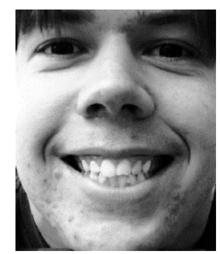

$$= \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N^2} \end{pmatrix}$$


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


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


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


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


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

  - 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

- We compute the average face

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

- Then subtract it from the training faces

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

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

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

$$Cov = AA^T = U\Sigma U^T$$

- Find eigenvalues of the covariance matrix

$$\Sigma = diag(\sigma_1, \sigma_2, \sigma_3, \ldots\ldots \sigma_{N^2})$$

- We are interested in at most M eigenvectors from *U*
- Build matrix U (*N^2 by M^**) from the eigenvectors of *L*

$$L = A^T A = V\Sigma_L V^T, \quad U = AV \quad \text{eigenfaces, } N^2 \times M^*$$

Eigenfaces

- Examples of some Eigenfaces

Eigenfaces, the algorithm

- Project each face onto the eigenface space Ω (*M** by 1)

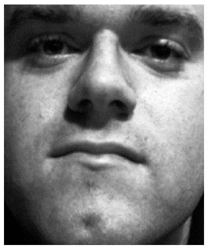
$$\Omega_1 = U^{\mathrm{T}}\left(\vec{a}_m\right), \quad \Omega_2 = U^{\mathrm{T}}\left(\vec{b}_m\right), \quad \Omega_3 = U^{\mathrm{T}}\left(\vec{c}_m\right), \quad \Omega_4 = U^{\mathrm{T}}\left(\vec{d}_m\right),$$
$$\Omega_5 = U^{\mathrm{T}}\left(\vec{e}_m\right), \quad \Omega_6 = U^{\mathrm{T}}\left(\vec{f}_m\right), \quad \Omega_7 = U^{\mathrm{T}}\left(\vec{g}_m\right), \quad \Omega_8 = U^{\mathrm{T}}\left(\vec{h}_m\right)$$

- Compute the threshold (half of max intra-class distance) to determine is the face matching is reasonable or not

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

Eigenfaces: **Recognition Procedure**

- To recognize an unknown face, first subtract the average face from it



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

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

- Compute its projection onto the face space U
- **Compute the distance in the face space between** the face **and** all known faces

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

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

- Reconstruct the face from eigenfaces and compute the distance between the face and its reconstruction

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

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

Eigenfaces, the algorithm

- Distinguish between
  - If $\xi \geq \theta$ then it is not a face; the distance between the face and its reconstruction is larger than threshold
  - If $\xi < \theta, \text{ and } \min\{\varepsilon_i\} > \theta$ then it is a new face
  - If $\xi < \theta, \text{ and } \min\{\varepsilon_i\} < \theta$ 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

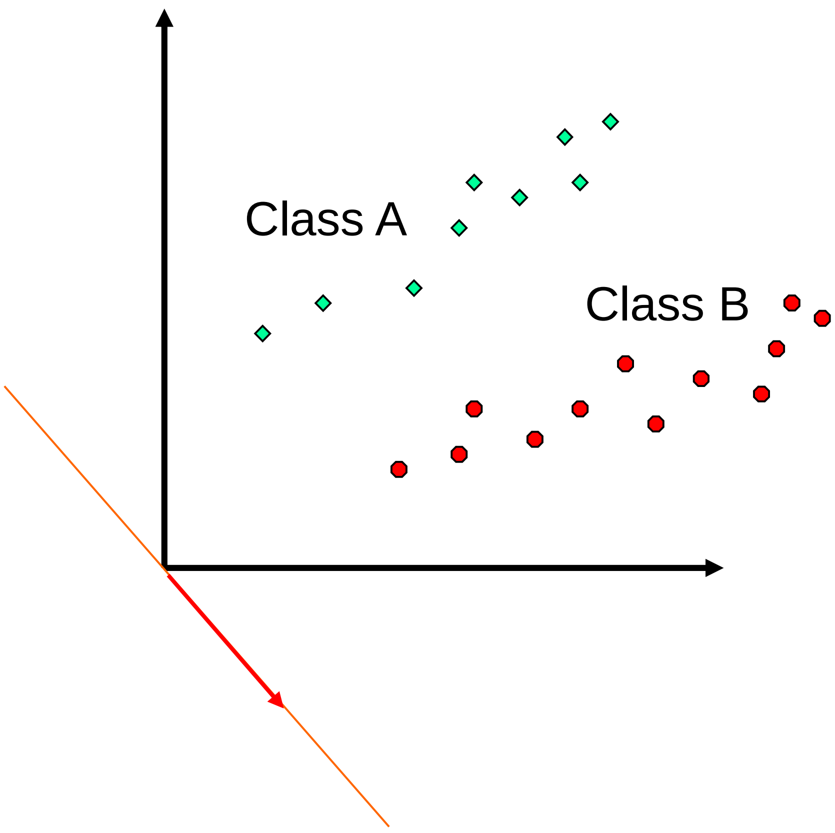
- Problems with eigenfaces
  - Different illumination, and different head pose, different alignment, different facial expression

"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

Fisherfaces

- Developed in 1997 by P. Belhumeur et al.
- Based on Fisher's Linear Discriminant Analysis (LDA), has lower error rates
- Works well even if different illumination and different facial expressions
- LDA maximizes the between-class scatter and minimizes the within-class scatter



Fisherfaces, the algorithm

- The average face of all persons and each person

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

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

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

Fisherfaces, the algorithm

- And subtract them from the training faces

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

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

Fisherfaces, the algorithm

- Build scatter matrices for each person 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

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

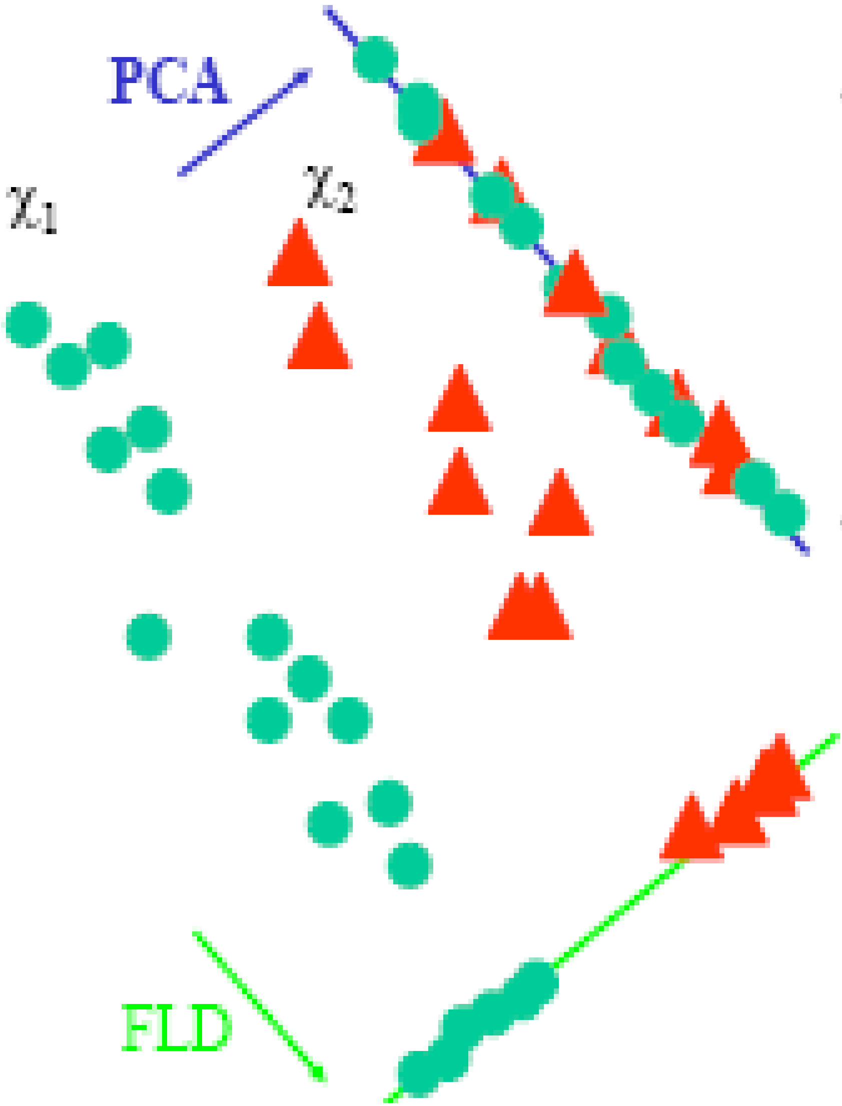
- We are seeking the matrix W maximizing

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

- LDA maximizes th s scatter and minimizes the within-class scatter
- To classify the face: 1) Project it onto the LDA-space; 2) Run a nearest-neighbor classifier

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

PCA & Fisher’s Linear Discriminant



- PCA (Eigenfaces)

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

Maximizes projected total scatter

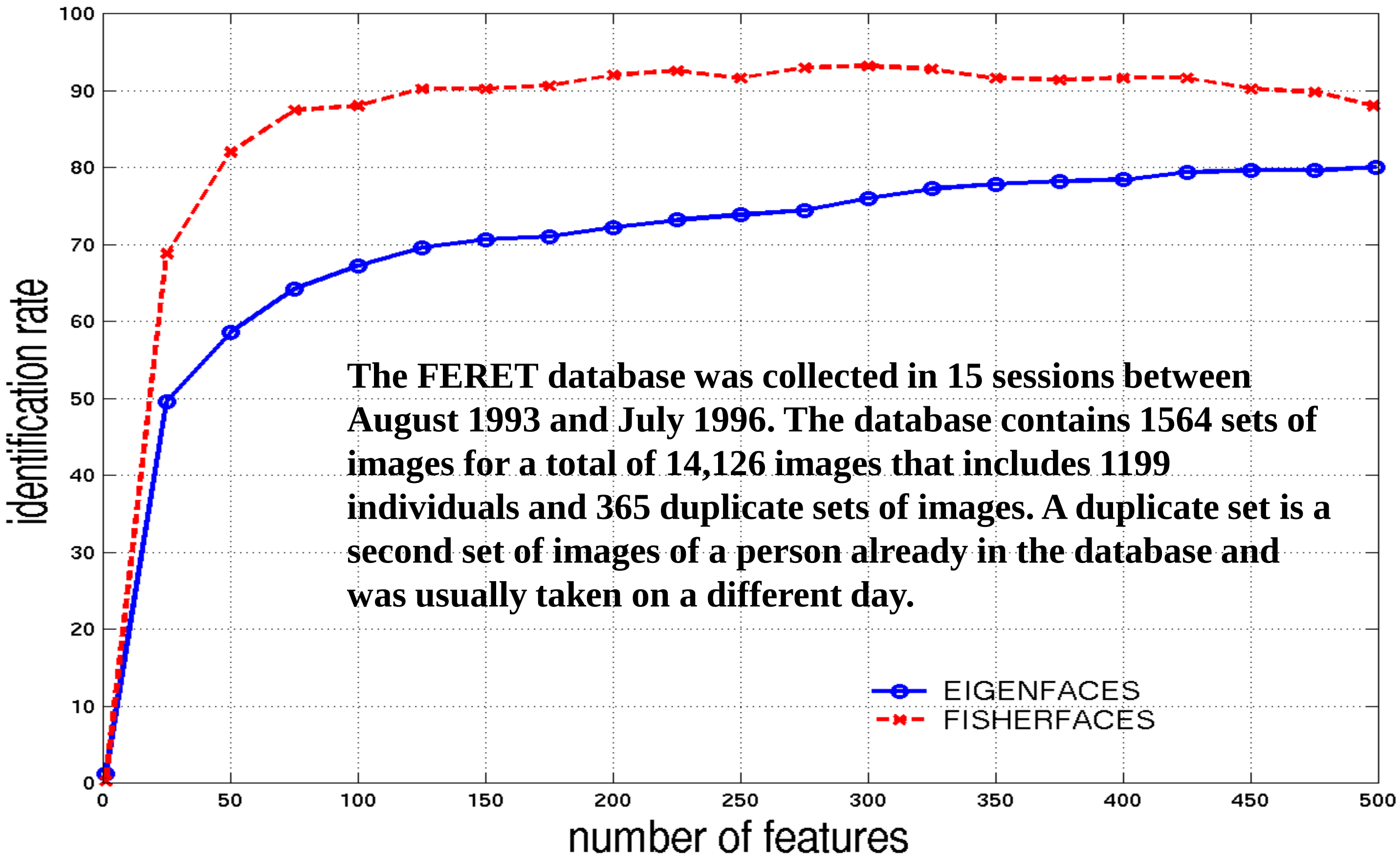
- Fisher’s Linear Discriminant

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

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

Comparison

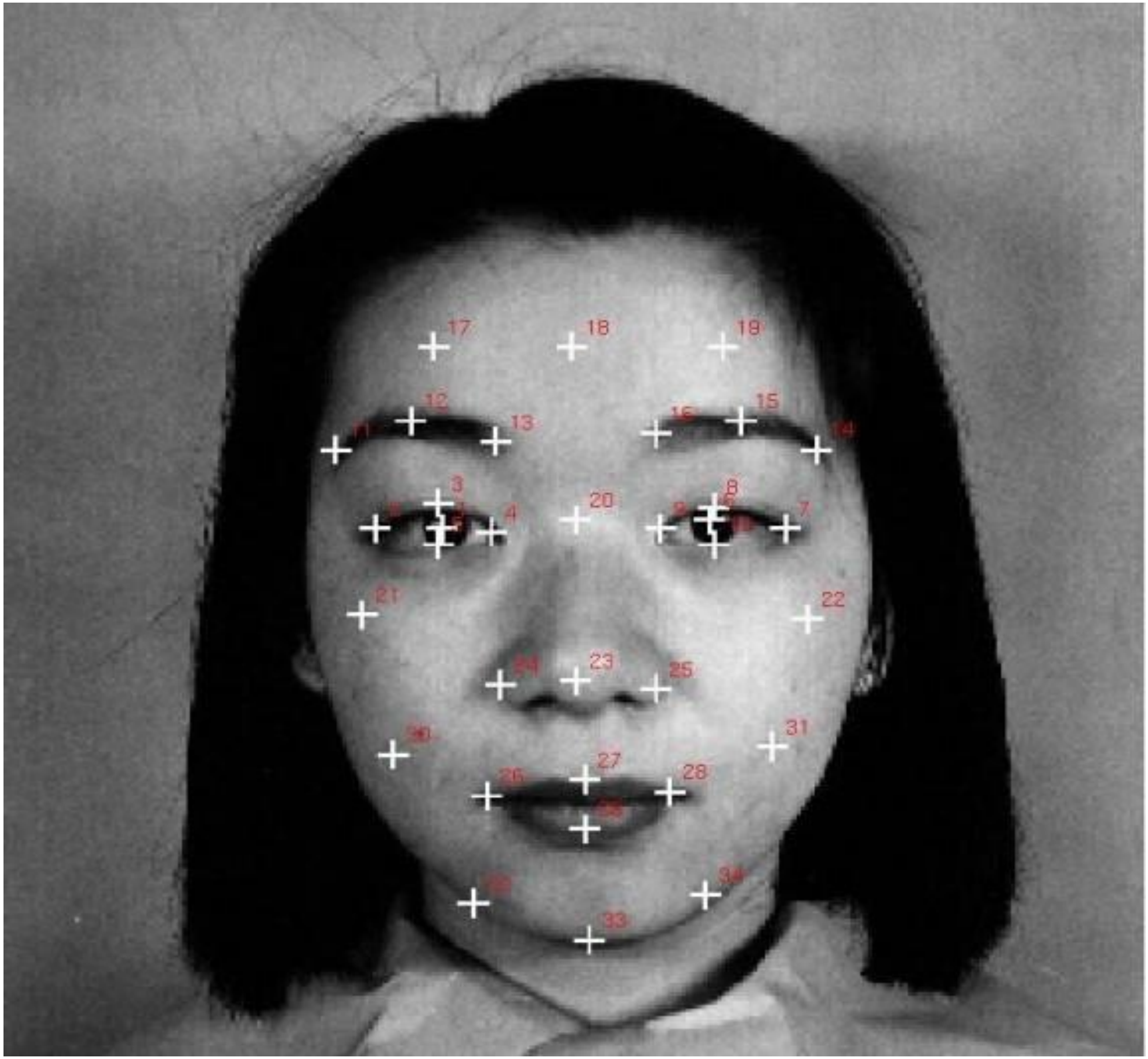
- FERET database http://www.nist.gov/itl/iad/ig/feret.cfm



best ID rate: eigenfaces 80.0%, fisherfaces 93.2%

Local Features based Face Recognition

- 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.



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

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/