6.891

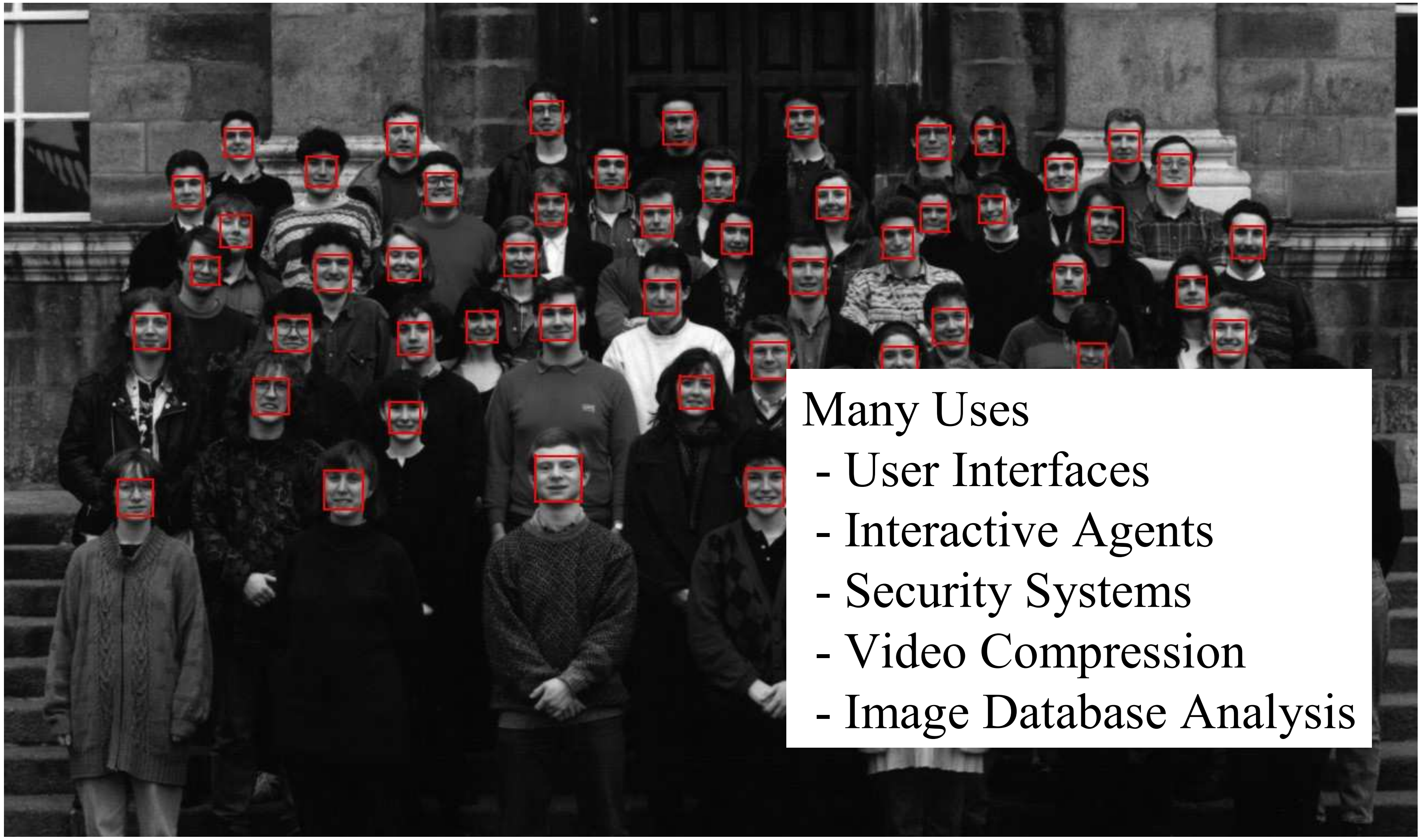
Computer Vision and Applications

Prof. Trevor. Darrell

Lecture 12: (Face) Detection

- Template matching
- Backprop
- SVM
- Boosting

Face Detection Example

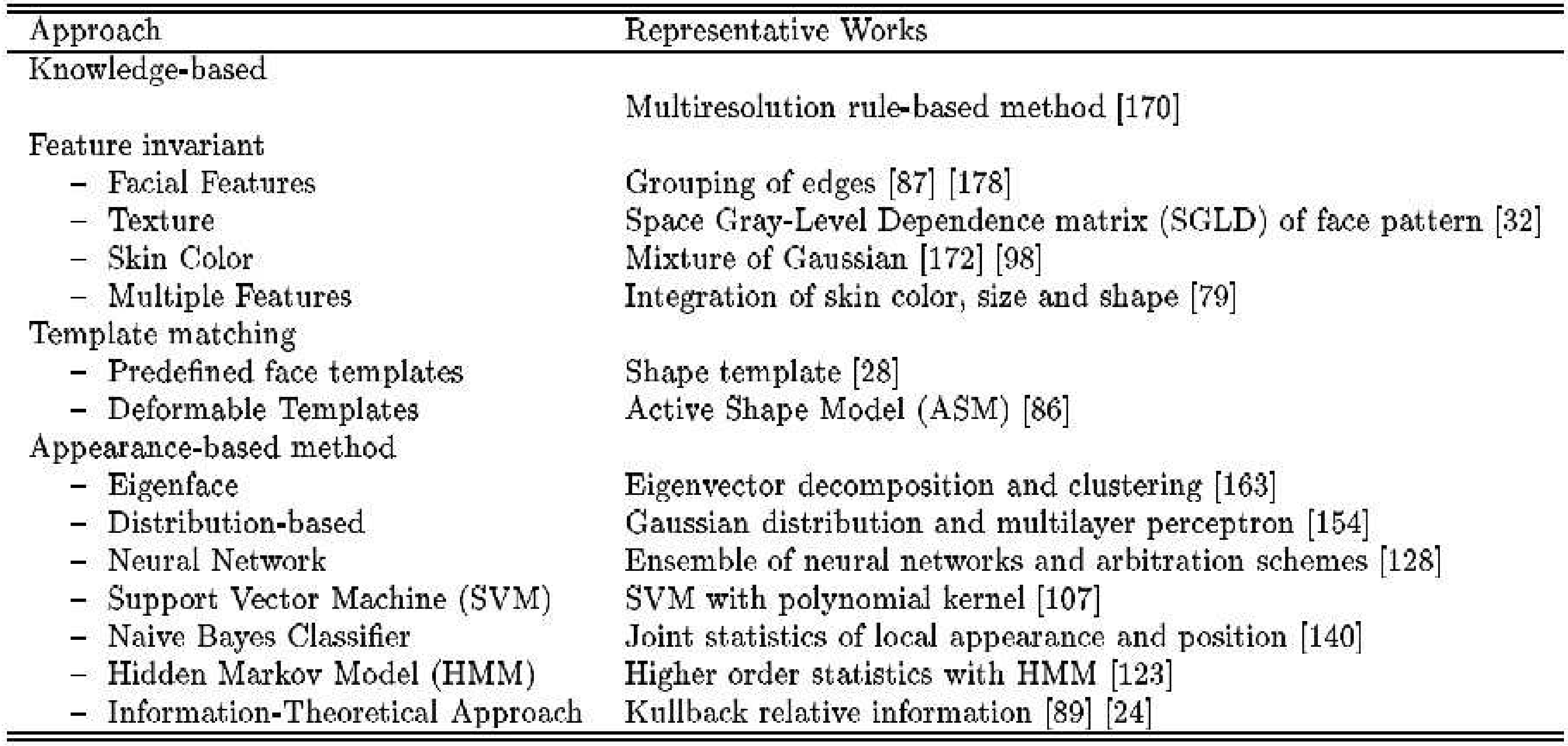


Many Uses
- User Interfaces
- Interactive Agents
- Security Systems
- Video Compression
- Image Database Analysis

Why Face Detection is Difficult?

- **Pose**: Variation due to the relative camera-face pose (frontal, 45 degree, profile, upside down), and some facial features such as an eye or the nose may become partially or wholly occluded.
- **Presence or absence of structural components**: Facial features such as beards, mustaches, and glasses may or may not be present, and there is a great deal of variability amongst these components including shape, color, and size.
- **Facial expression**: The appearance of faces are directly affected by a person's facial expression.
- **Occlusion**: Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.
- **Image orientation**: Face images directly vary for different rotations about the camera's optical axis.
- **Imaging conditions**: When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face.

Face Detection Methods



Approach	Representative Works
Knowledge-based	
	Multiresolution rule-based method [170]
Feature invariant	
– Facial Features	Grouping of edges [87] [178]
– Texture	Space Gray-Level Dependence matrix (SGLD) of face pattern [32]
– Skin Color	Mixture of Gaussian [172] [98]
– Multiple Features	Integration of skin color, size and shape [79]
Template matching	
– Predefined face templates	Shape template [28]
– Deformable Templates	Active Shape Model (ASM) [86]
Appearance-based method	
– Eigenface	Eigenvector decomposition and clustering [163]
– Distribution-based	Gaussian distribution and multilayer perceptron [154]
– Neural Network	Ensemble of neural networks and arbitration schemes [128]
– Support Vector Machine (SVM)	SVM with polynomial kernel [107]
– Naive Bayes Classifier	Joint statistics of local appearance and position [140]
– Hidden Markov Model (HMM)	Higher order statistics with HMM [123]
– Information-Theoretical Approach	Kullback relative information [89] [24]

M.H. Yang, D. Kriegman, N. Ahuja, *Detecting faces in images, a survey*", PAMI vol.24,no.1, January, 2002.

Detecting Human Faces in Color Images

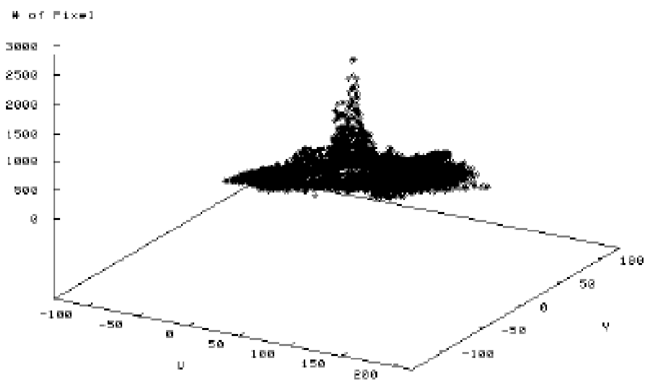


Fig.1

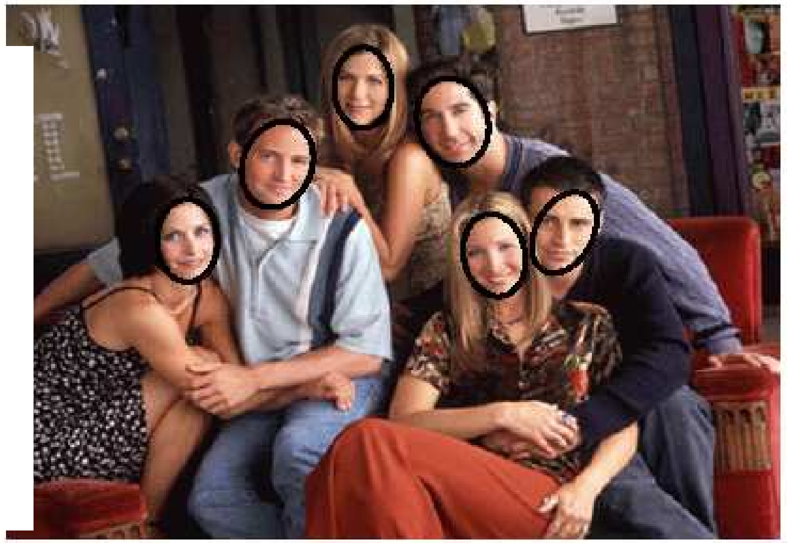


Fig.2

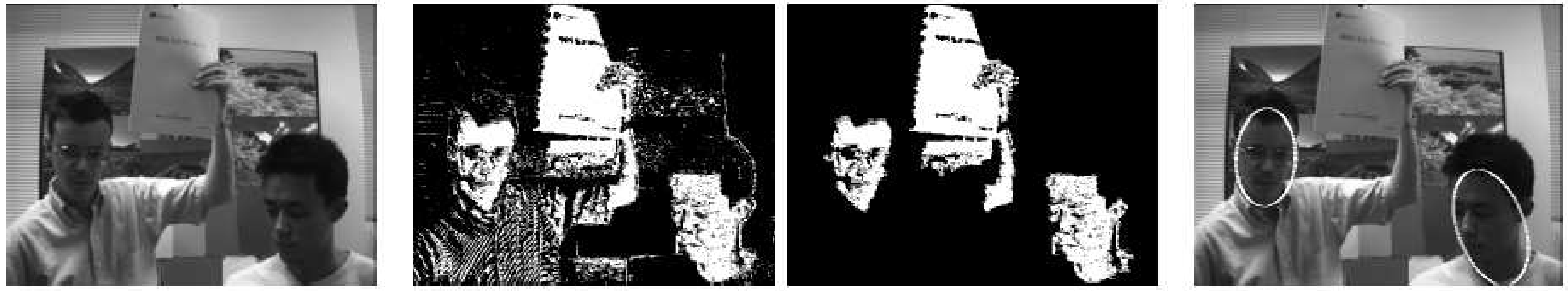
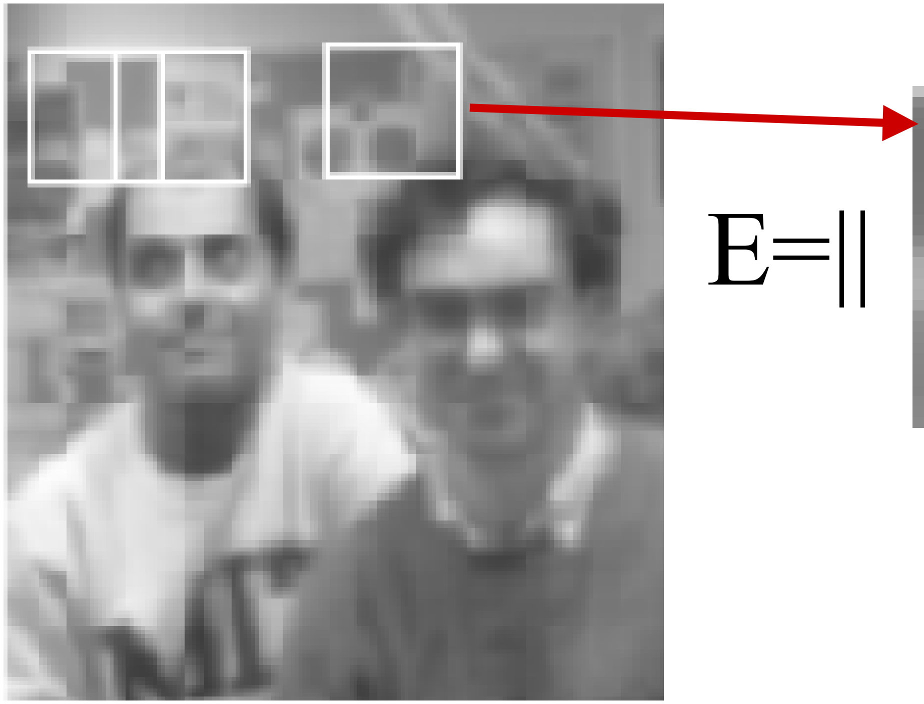


Fig.3

Template Matching



$E = \|$ $-$ $\|^2$

Structured templates

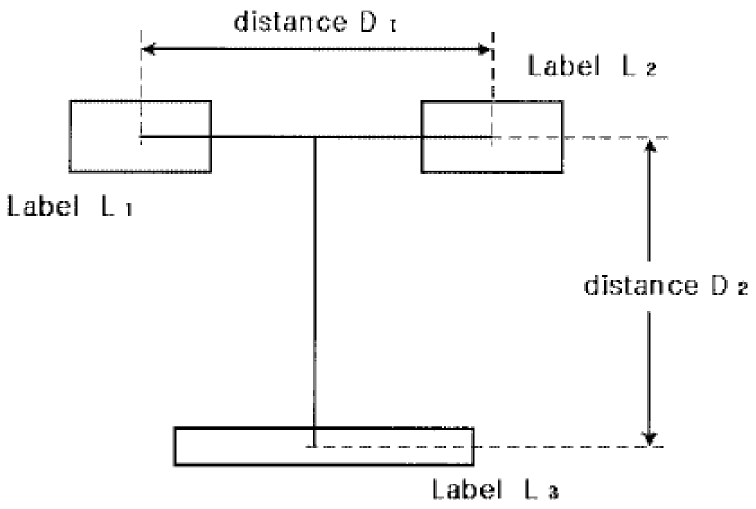


Fig.1

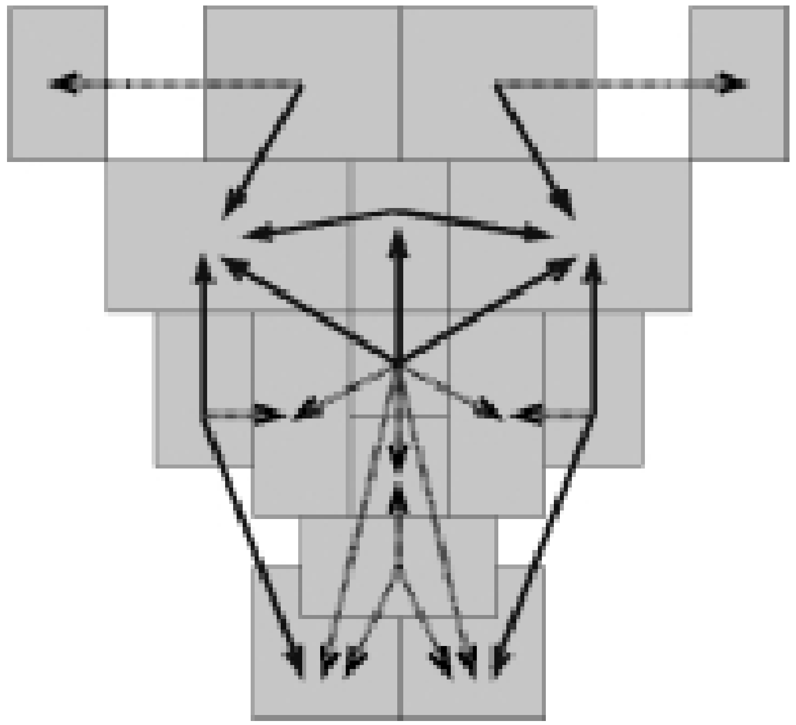
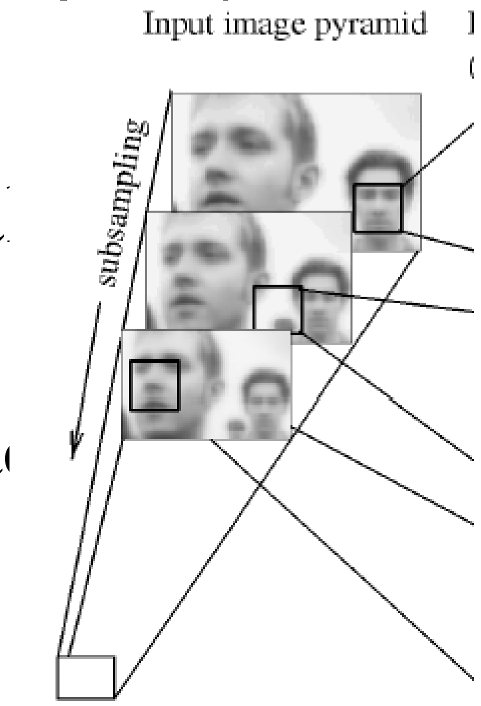


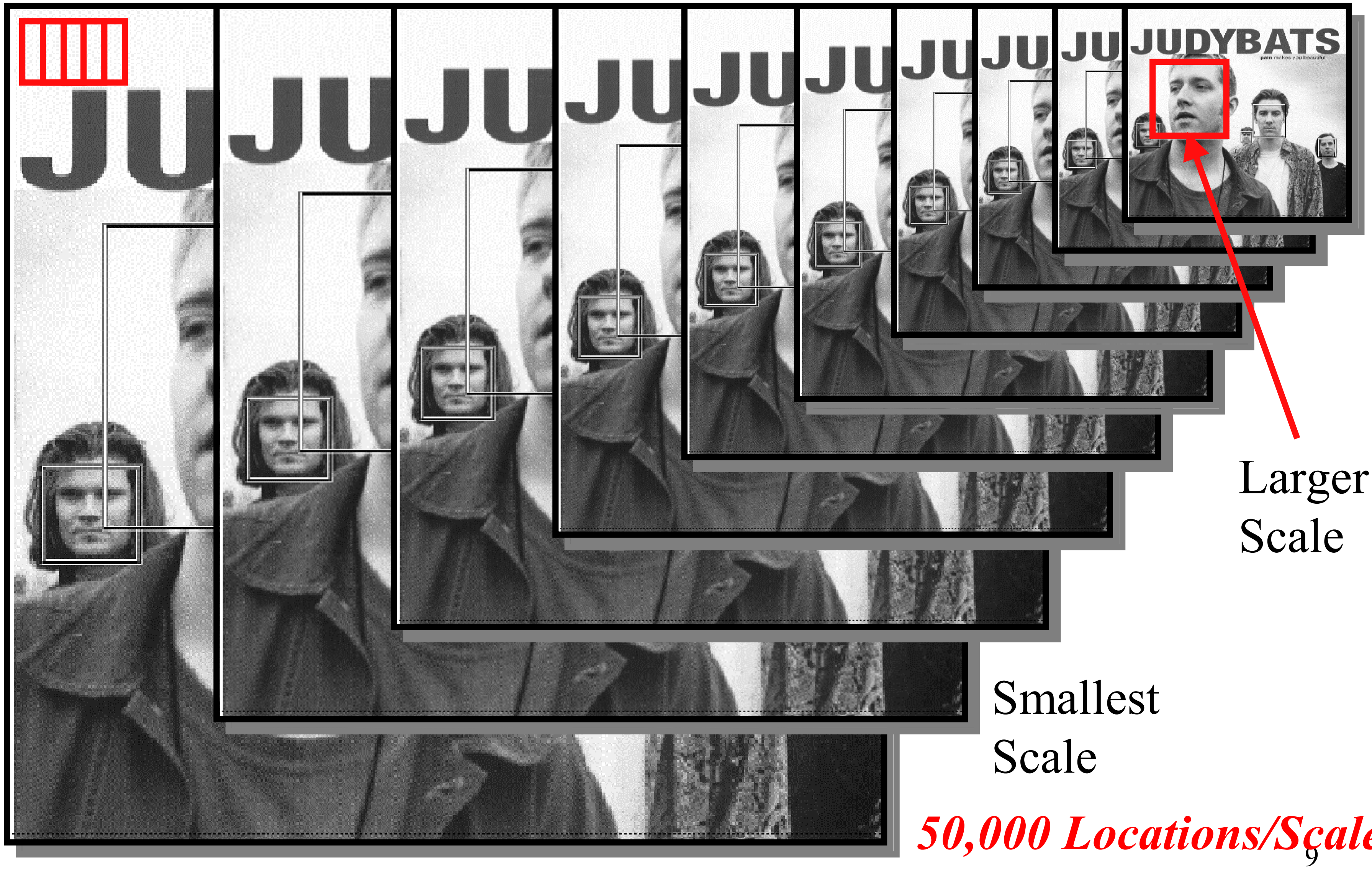
Fig.2

Multi-scale search

- Search at multiple scales (and pose)
- Multiple templates
- Single template, multiple sca
- Image Pyramid
  - decimate image by constant fa
  - efficient search



The Classical Face Detection Process



Larger Scale

Smallest Scale

50,000 Locations/Scales

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

Learning approach

- Learn Classifier Parameters
- Benefits:
  - no human domain experience necessary
  - parameters can be derived from large data sets, and thus be more reliable
  - opportunity to improve performance by correcting mistakes and including in training set

Too many templates…

Image templates (simplest view-based method – straw man)

- keep an image of every object from different viewing directions, lighting conditions, etc.
- nearest neighbor cross-correlation matching with images in model database (or robust matching for clutter & occlusion)

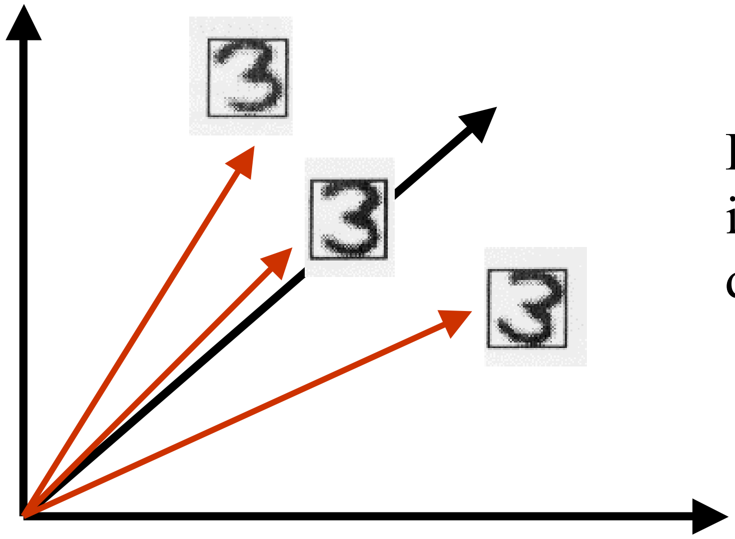
Obvious problems:

- storage and computation costs become unreasonable as the number of objects increases
- may require very large ensemble of ‘training’ images

Subspace Methods

How can we find more efficient representations for the ensemble of views, and more efficient methods for matching?

- Idea: images are not random… especially images of the same object that have similar appearance

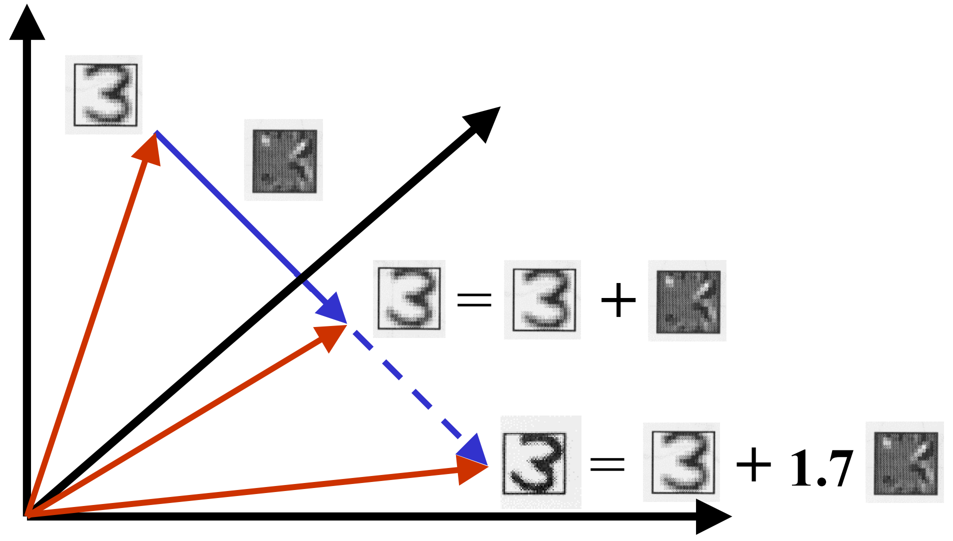


E.g., let images be represented as points in a high-dimensional space (e.g., one dimension per pixel)

Fleet & Szeliksi

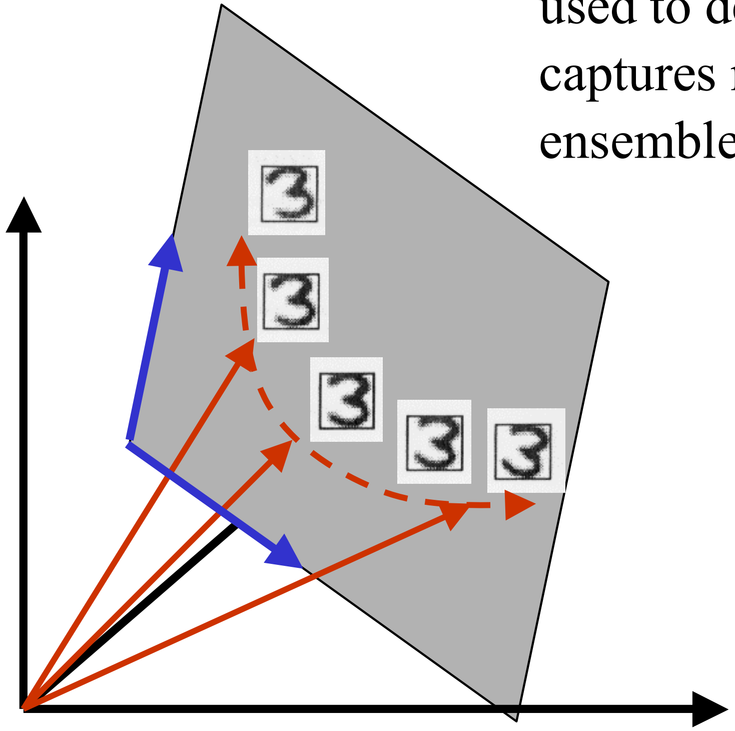
Linear Dimension Reduction

Given that differences are structured, we can use *'basis images'* to transform images into other images in the same space.



Fleet & Szeliksi

Linear Dimension Reduction



What linear transformations of the images can be used to define a lower-dimensional subspace that captures most of the structure in the image ensemble?

Fleet & Szeliksi

Observation

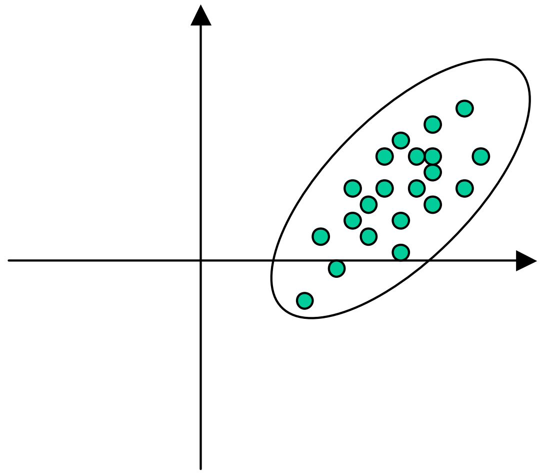
$$\vec{x}^{n} \approx \sum_{i=1}^{M} z_i^n \vec{u}_i + \sum_{j=M+1}^{D} b_j \vec{u}_j$$

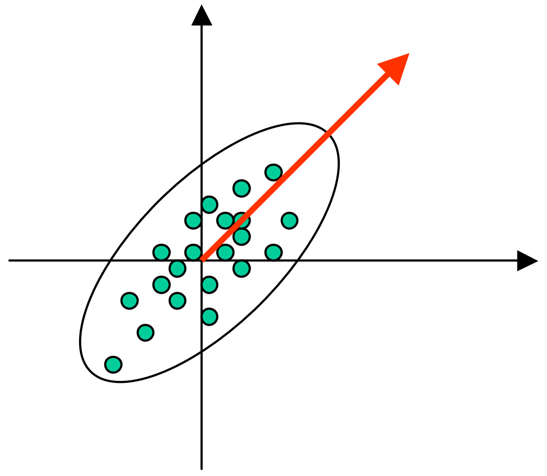
Approximation $\tilde{x}_n$ Error

Want the M bases that minimize the mean squared error over the training data

$$\min E_M = \sum_{n=1}^{N} \left\| \vec{x}^{n} - \tilde{x}^{n} \right\|^2$$

Intuition





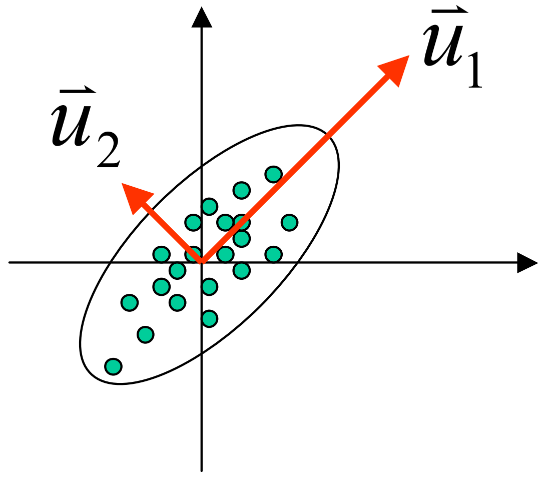
If I give you the mean and one vector to represent the data, what vector would you choose?

Why?

Intuition

$$\vec{x}^n \approx \sum_{i=1}^{M} z_i^n \vec{u}_i + \sum_{j=M+1}^{D} b_j \vec{u}_j$$

$$\min E_M = \sum_{n=1}^{N} \left\| \vec{x}^n - \tilde{x}^n \right\|^2$$



Projecting onto $\vec{u}_1$ captures the majority of the variance and hence projecting onto it minimizes the error

Principal Component Analysis

- Sample mean and covariance:

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} \vec{x}^{\,n} \qquad C = \frac{1}{N-1}\sum_{n=1}^{N} (\vec{x}^{\,n} - \bar{x})(\vec{x}^{\,n} - \bar{x})^T$$

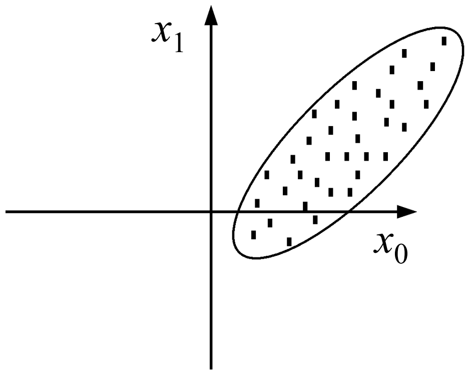
- Let the eigenvectors and eigenvalues of C be $\vec{\mathbf{e}}_k$ and λ_k for $k < D$ (i.e., $C\vec{e}_k = \lambda_k \vec{e}_k$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$)
- In matrix form: $C\,E = E\,L$, where $L = \operatorname{diag}(\lambda_1, \ldots, \lambda_D)$ and $E = [\vec{e}_1, \ldots \vec{e}_D]$
- Because C is symmetric positive-definite, we know $E^{-1} = E^T$

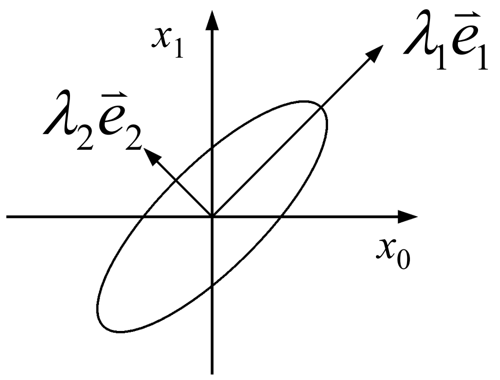
Fleet & Szeliski

Principal Component Analysis

• Eigenvectors are the *principal directions*, and the eigenvalues represent the variance of the data along each principal direction

* λ_k is the marginal variance along the principal direction $\vec{e}_k$





Principal Component Analysis

- The first principal direction $\vec{\mathbf{e}}_1$ is the direction along which the variance of the data is maximal, i.e. it maximizes

$$\vec{\mathbf{e}}^T C \vec{\mathbf{e}} \quad \text{where} \quad \vec{\mathbf{e}}^T \vec{\mathbf{e}} = 1$$

- The second principal direction maximizes the variance of the data in the orthogonal complement of the first eigenvector.
- etc.

Principal Component Analysis

- PCA Approximate Basis: If $\lambda_k \approx 0$ for $k > M$ for some $M << D$, then we can approximate the data using only M of the principal directions (basis vectors):

  - If $\mathbf{B} = [\vec{e}_1, \ldots, \vec{e}_M]$, then for all points

$$\vec{x}^n \approx \mathbf{B}\vec{a}^n + \overline{x}$$

where $a_k^n = (\vec{x}^n - \overline{x})^T \vec{e}_k$

Fleet & Szeliksi

PCA

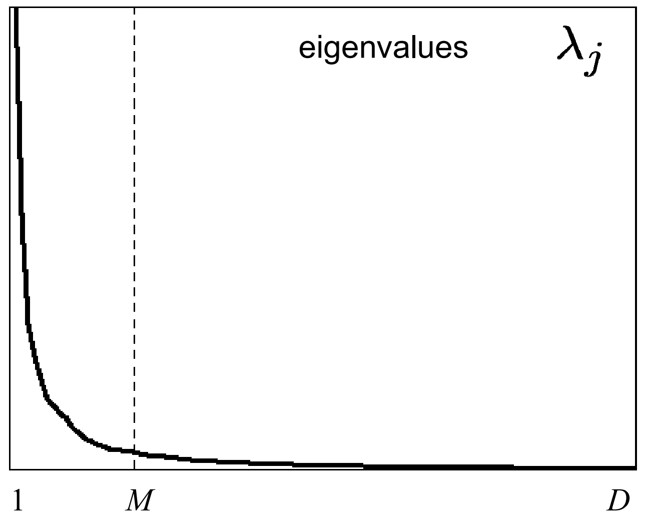
– Over all rank M bases, $\mathbf{B}$ minimizes the MSE of approximation

$$\sum_{j=M+1}^{D} \lambda_j$$

•Choosing subspace dimension M:

– look at decay of the eigenvalues as a function of M

– Larger M means lower expected error in the subspace data approximation



Fleet & Szeliski

Computing using SVD

Let $X = [\vec{x}^1 \cdots \vec{x}^D]$

Compute the mean column vector: $\bar{x} = \frac{1}{D}\sum_{i=1}^{D} x^i$

Subtract the mean from each column.

$$A = X - \bar{x} = [(\vec{x}^1 - \bar{x})\cdots(\vec{x}^D - \bar{x})]$$

Singular Value Decomposition allows us to write *A* as:

$$A = U\Sigma V^T$$

SVD and PCA

$$A = U\Sigma V^T$$

Orthonormal columns

$$\begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_D \end{bmatrix}$$

Diagonal matrix of *singular values*

SVD and PCA

Note:

$$
\begin{aligned}
C &= \frac{1}{D} A A^T \\
&= \frac{1}{D} U \Sigma V^T (U \Sigma V^T)^T \\
&= \frac{1}{D} U \Sigma V^T V \Sigma U^T \\
&= \frac{1}{D} U \Sigma^2 U^T
\end{aligned}
$$

In other words

$$C \vec{u}_i = \frac{\sigma^2}{D} \vec{u}_i$$

i.e. the singular vectors of *A* are the eigenvectors of the covariance matrix *C*.

SVD and PCA

- So the columns of *U* are the eigenvectors
- And the eigenvalues are just

$$\lambda_k = \frac{\sigma_k^2}{D}$$

The benefit of eigenfaces over nearest neighbor

$$|\vec{y}_1 - \vec{y}_2|^2 = (\vec{y}_1 - \vec{y}_2)^T (\vec{y}_1 - \vec{y}_2)$$

image differences

eigenvalues

$$= (U\vec{x}_1 - U\vec{x}_2)^T (U\vec{x}_1 - U\vec{x}_2)$$

basis functions

$$= (\vec{x}_1^T U^T - \vec{x}_2^T U^T)(U\vec{x}_1 - U\vec{x}_2)$$

$$= \vec{x}_1^T \vec{x}_1 - \vec{x}_2^T \vec{x}_1 - \vec{x}_1^T \vec{x}_2 + \vec{x}_2^T \vec{x}_2$$

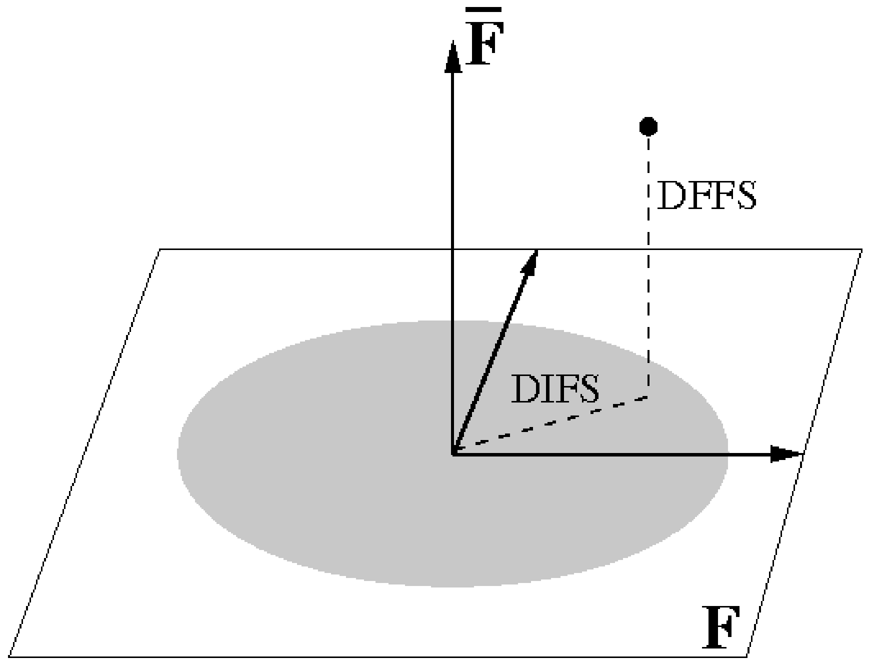
$$= (\vec{x}_1^T - \vec{x}_2^T)(\vec{x}_1 - \vec{x}_2)$$

$$= |\vec{x}_1 - \vec{x}_2|^2$$

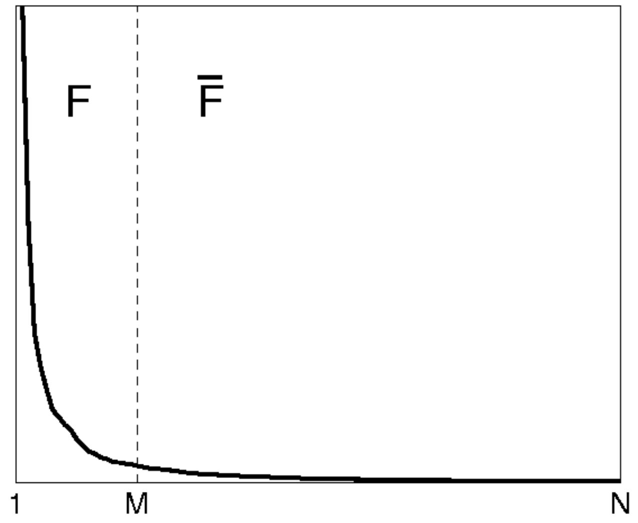
eigenvalue differences

Subspace Face Detector

- PCA-based Density Estimation p(x)
- Maximum-likelihood face detection based on DIFS + DFFS



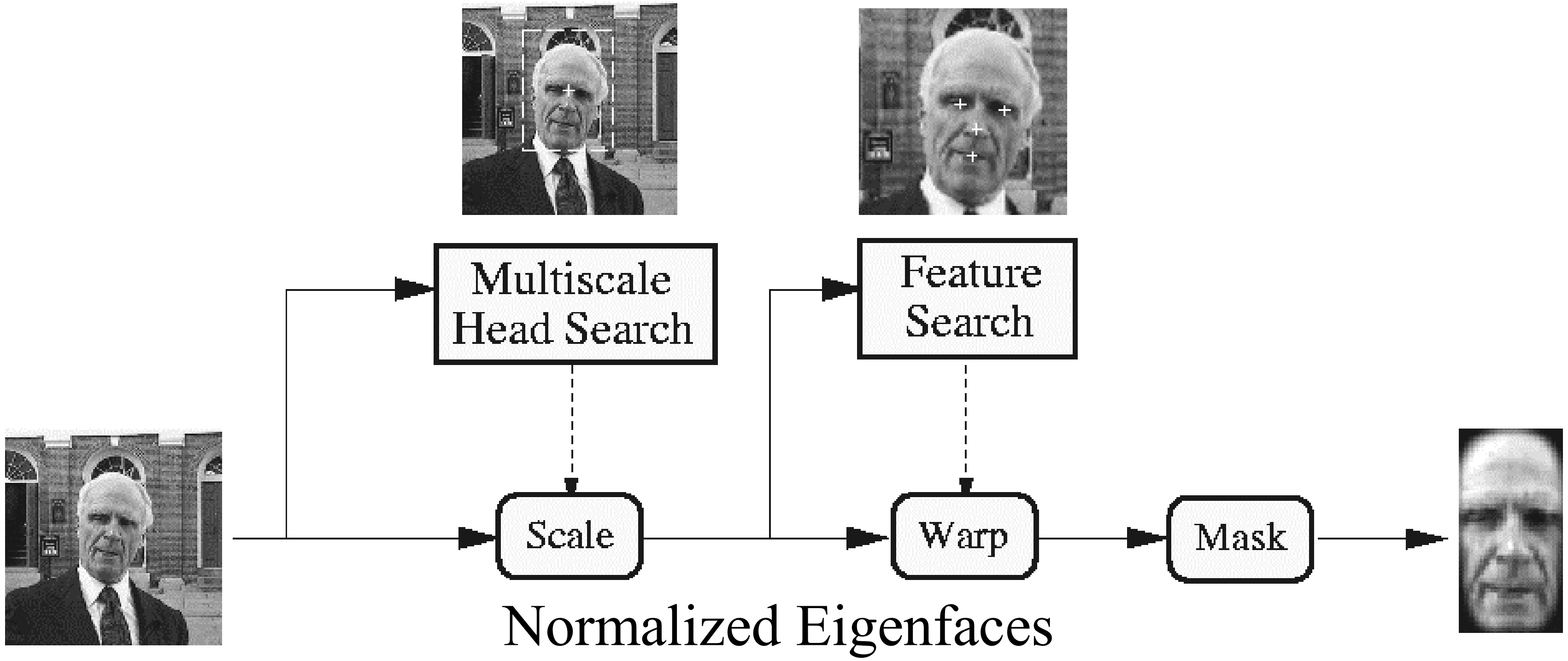
Eigenvalue spectrum



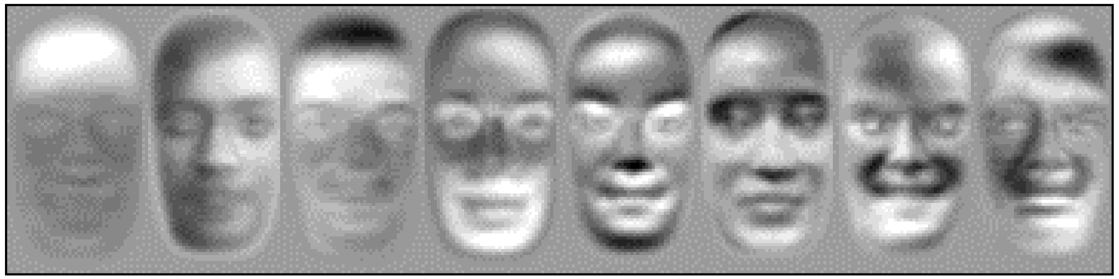
Moghaddam & Pentland, “Probabilistic Visual Learning for Object Detection,” ICCV’95.

Subspace Face Detector

- Multiscale Face and Facial Feature Detection & Rectification



Normalized Eigenfaces

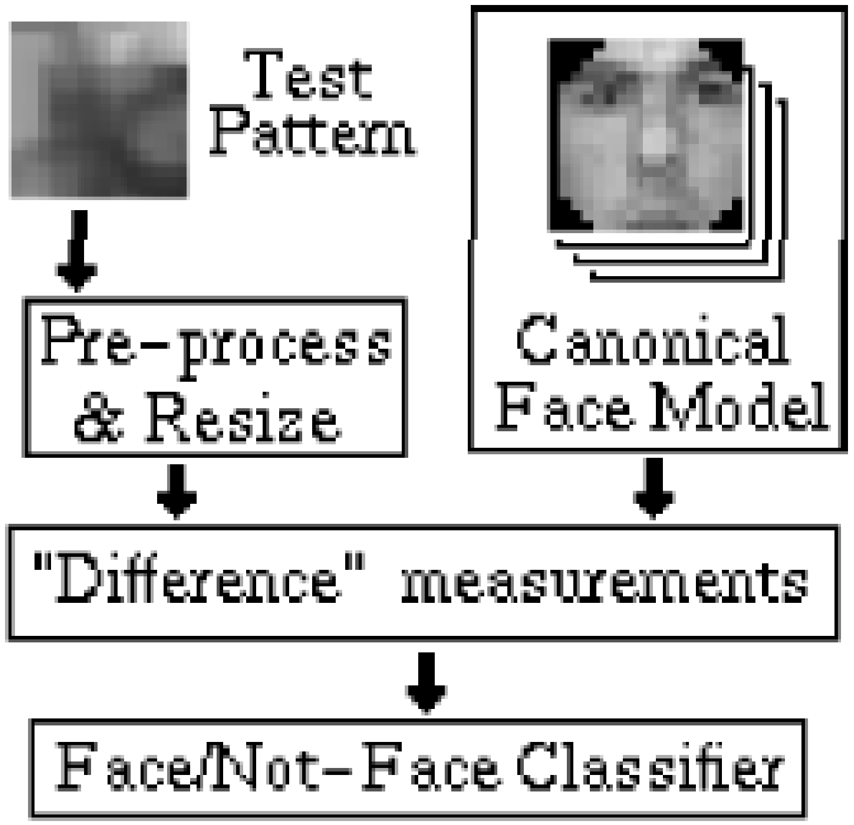


Moghaddam & Pentland, “Probabilistic Visual Learning for Object Detection,” ICCV’95.

Sung and Poggio

- Density learning approach
- Mixture of Gaussians for face and not-face
- One of the first applications of learning for face detection with large training sets.
  - Kah-Kay Sung and Tomaso Poggio, Example-Based Learning for View-based Human Face Detection, IEEE Trans. PAMI 20(1), January 1998
  - MIT AI TR 1572, 1996

Face detector architecture

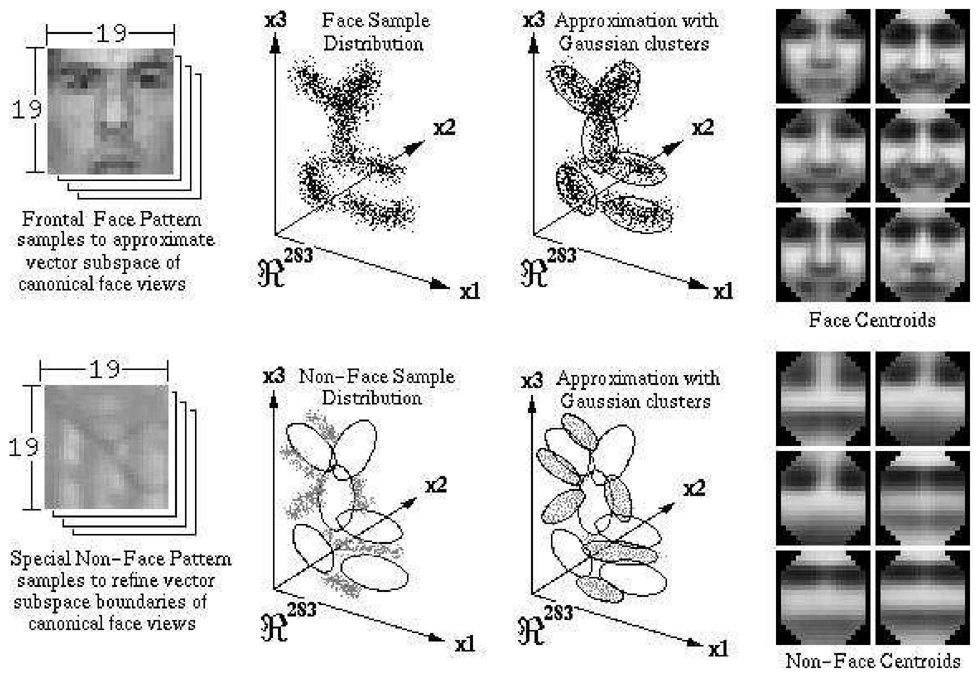


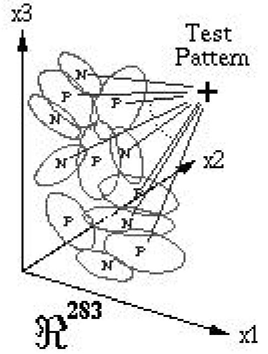
(multi-layer perceptron)

[Sung and Poggio]

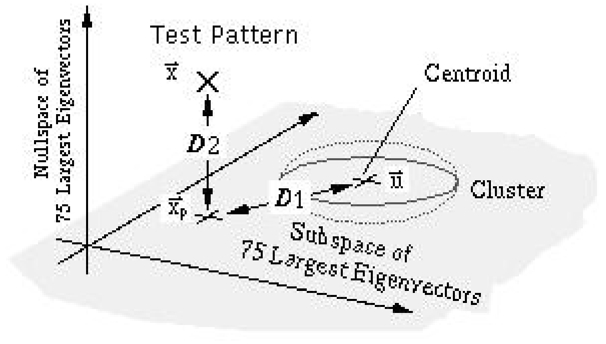
Distribution-Based Face Detector

- Learn face and nonface models from examples [Sung and Poggio 95]
- Cluster and project the examples to a lower dimensional space using Gaussian distributions and PCA
- Detect faces using distance metric to face and nonface clusters


19
19
Frontal Face Pattern samples to approximate vector subspace of canonical face views
x3
Face Sample Distribution
x2
$\Re^{283}$
x1
x3
Approximation with Gaussian clusters
x2
$\Re^{283}$
x1
Face Centroids
19
19
Special Non-Face Pattern samples to refine vector subspace boundaries of canonical face views
x3
Non-Face Sample Distribution
x2
$\Re^{283}$
x1
x3
Approximation with Gaussian clusters
x2
$\Re^{283}$
x1
Non-Face Centroids



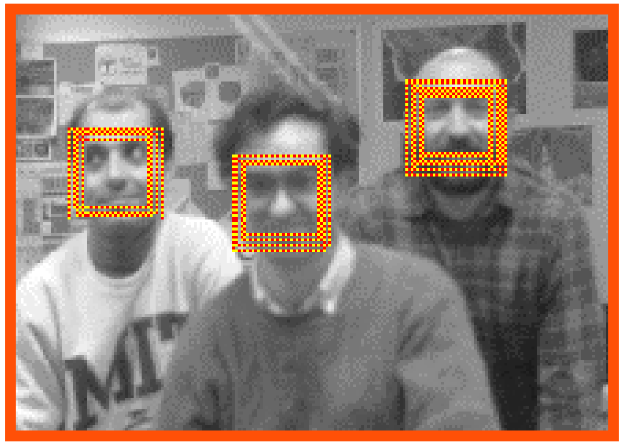
(a)

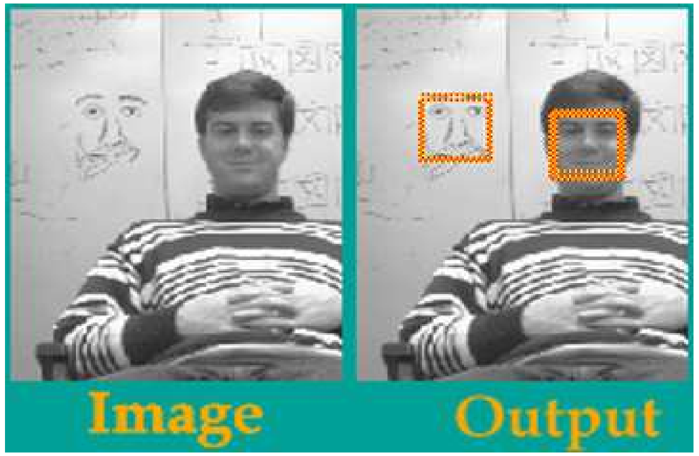

Nullspace of 75 Largest Eigenvectors
Test Pattern
$\vec{x}$
$D2$
Centroid
$\vec{u}$
Cluster
$\vec{x}_p$
$D1$
Subspace of 75 Largest Eigenvectors

(b)

Distribution-Based Face Detector

- Learn face and nonface models from examples [Sung and Poggio 95]

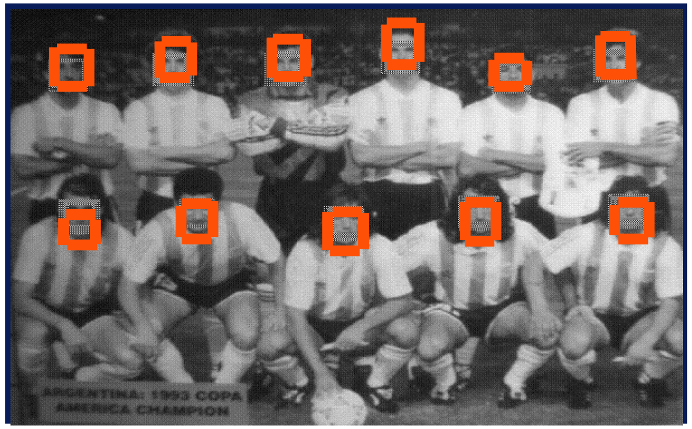




Training Database

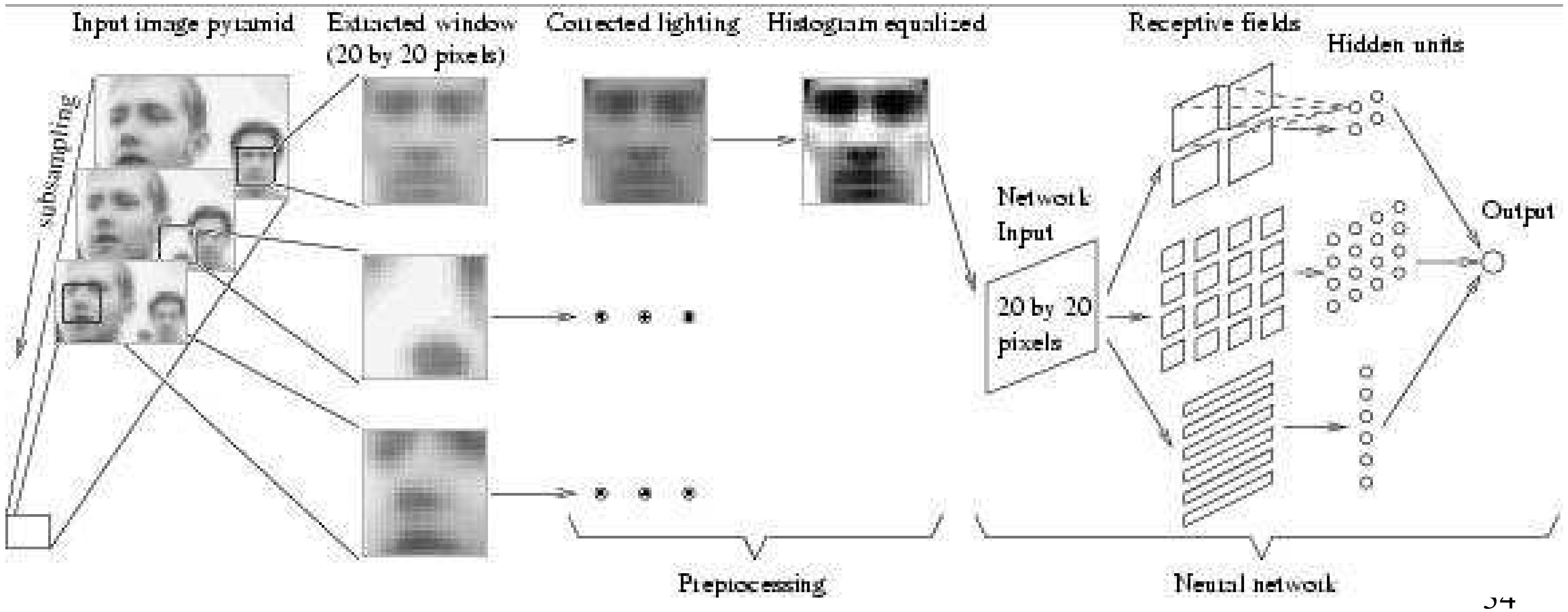
1000+ Real, 3000+ *VIRTUAL*

50,0000+ Non-Face Pattern

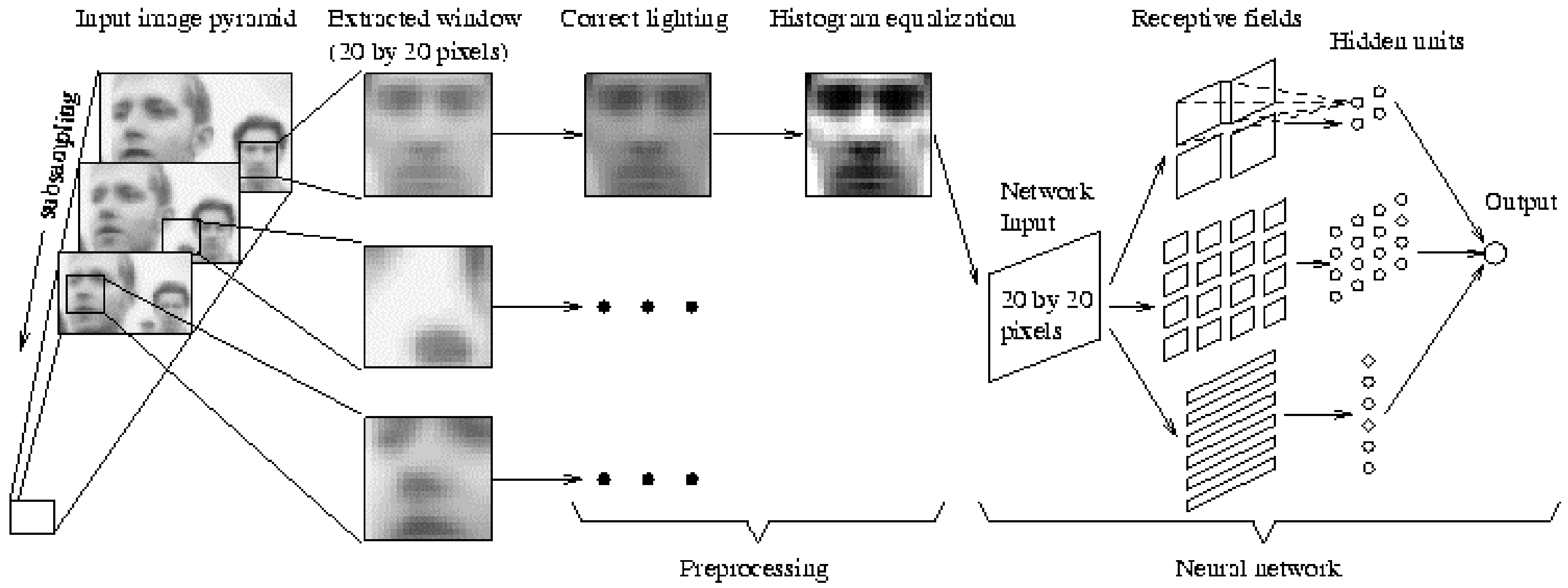


Neural Network-Based Face Detector

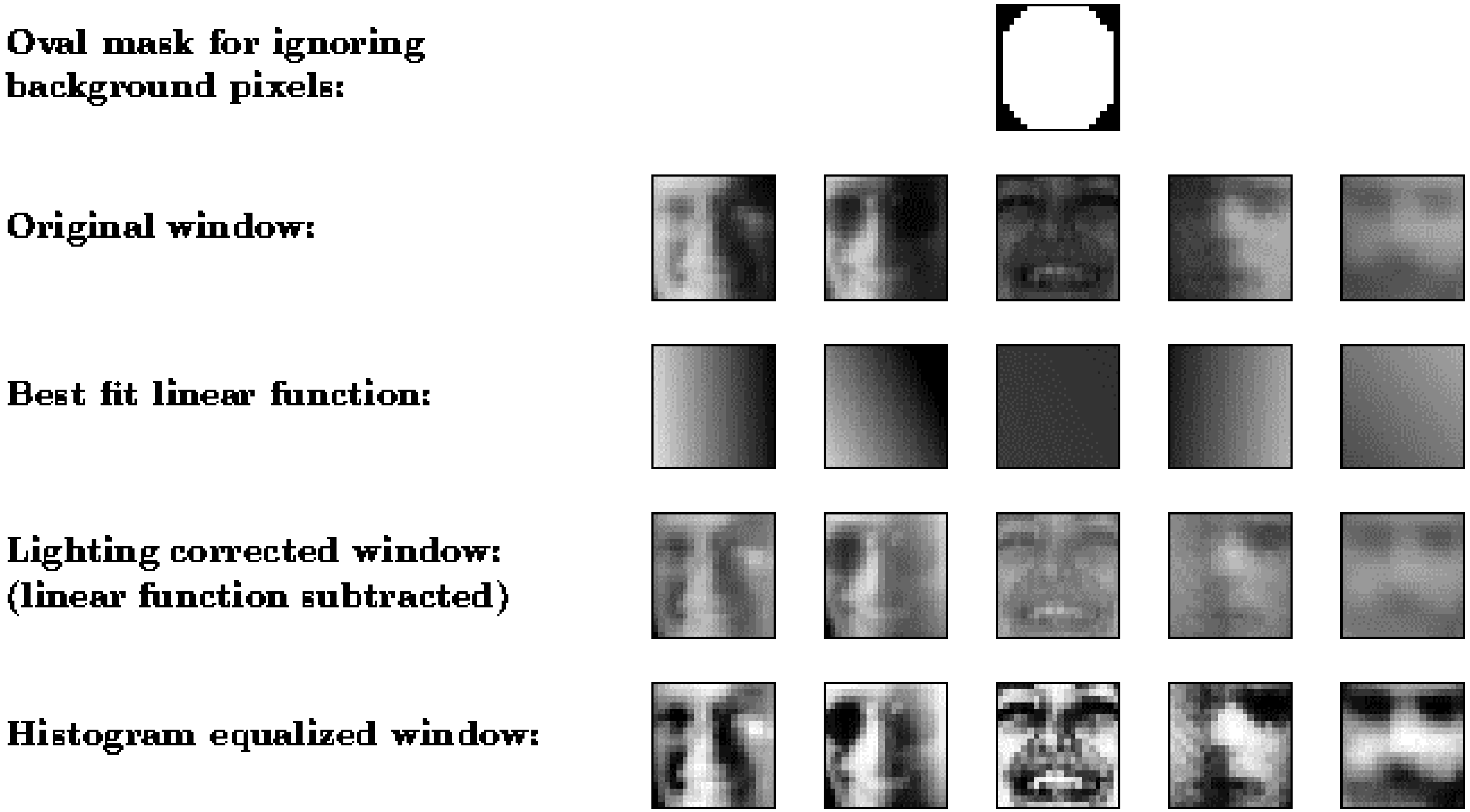
- Explicit generative model was too slow…
- Train a set of multilayer perceptrons and arbitrate a decision among all outputs [Rowley et al. 98]



The basic algorithm used for face detection



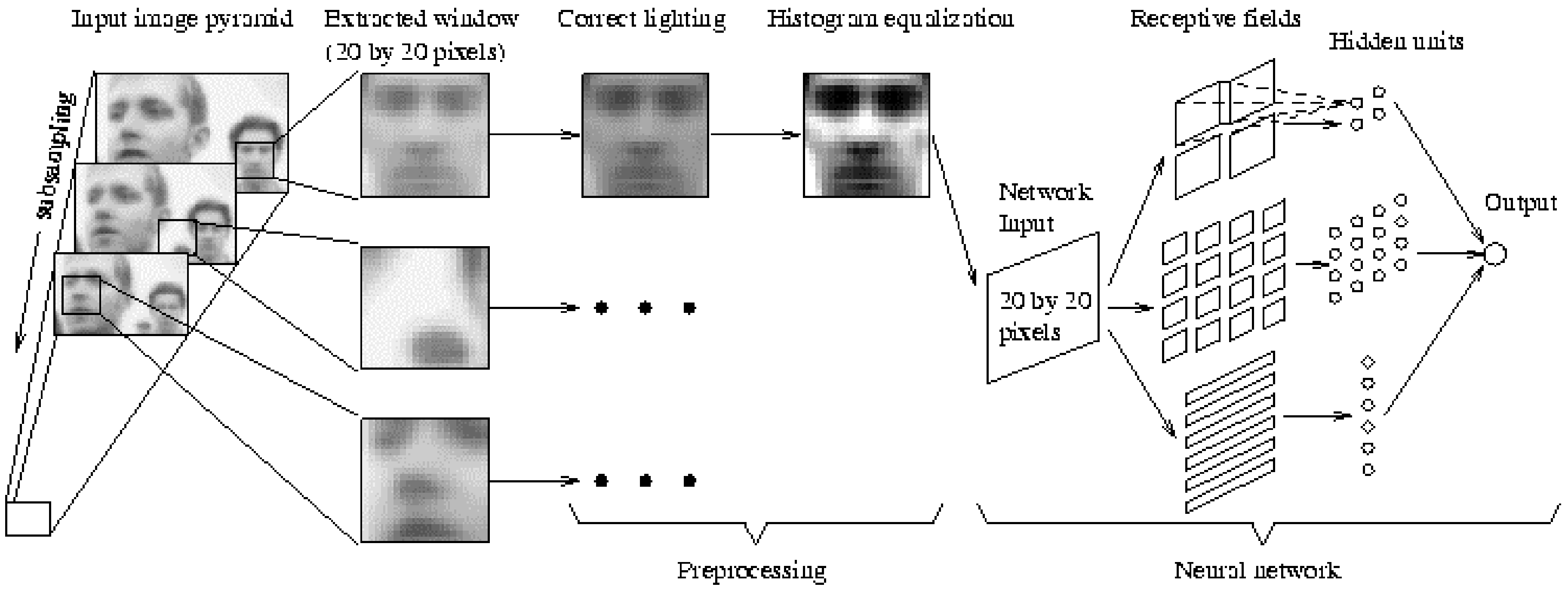
From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/


Oval mask for ignoring background pixels:
Original window:
Best fit linear function:
Lighting corrected window: (linear function subtracted)
Histogram equalized window:

The steps in preprocessing a window. First, a linear function is fit to the intensity values in the window, and then subtracted out, correcting for some extreme lighting conditions. Then, histogram equalization is applied, to correct for different camera gains and to improve contrast. For each of these steps, the mapping is computed based on pixels inside the oval mask, while the mapping is applied to the entire window.

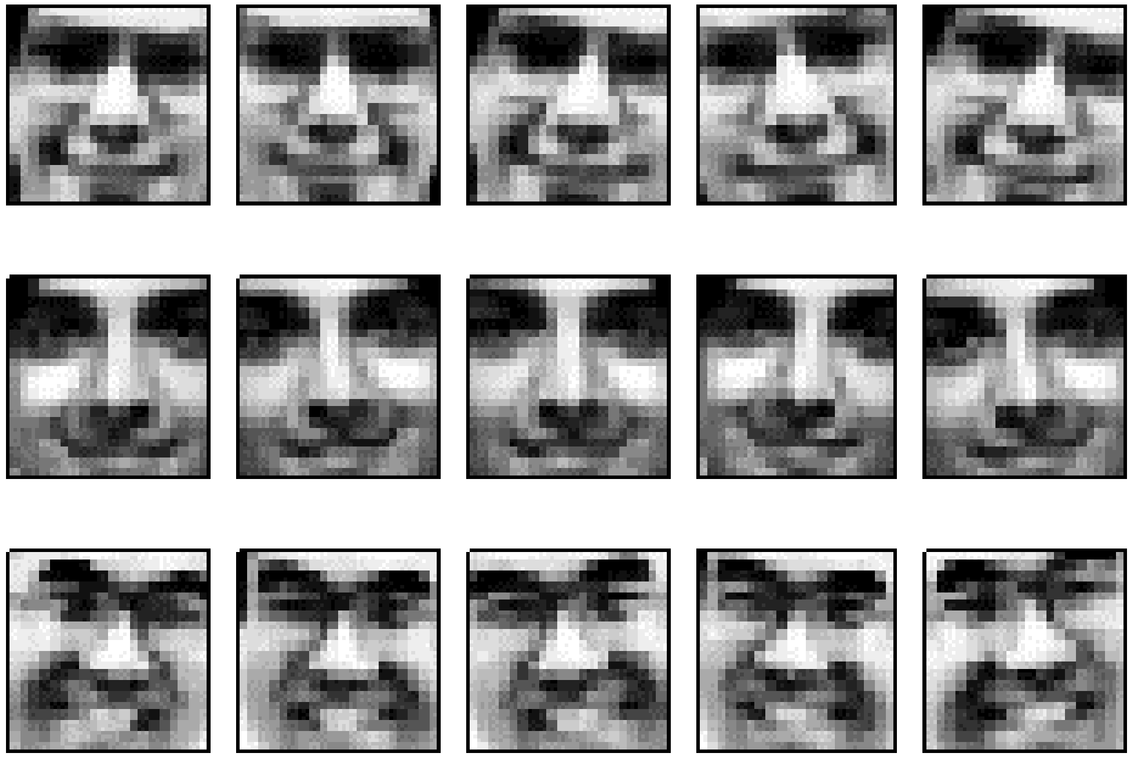
From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/

The basic algorithm used for face detection



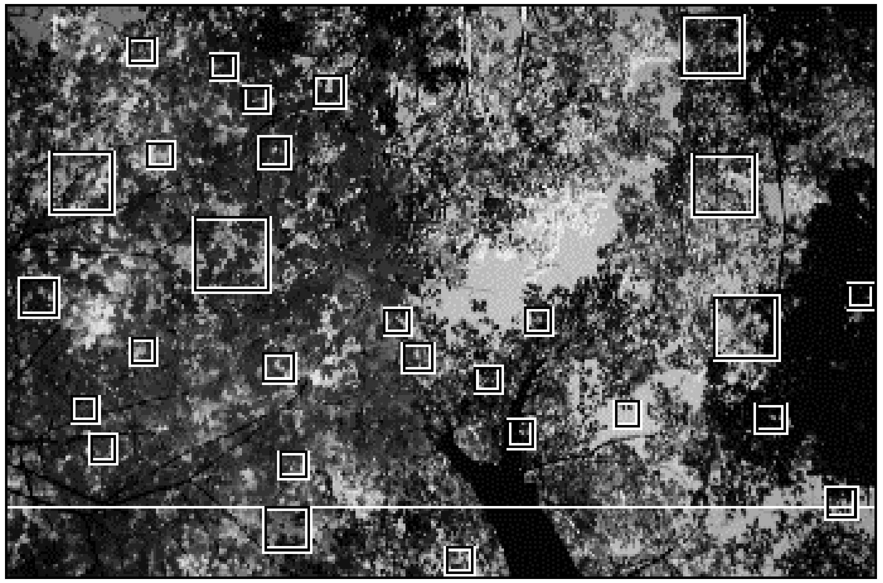
From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/

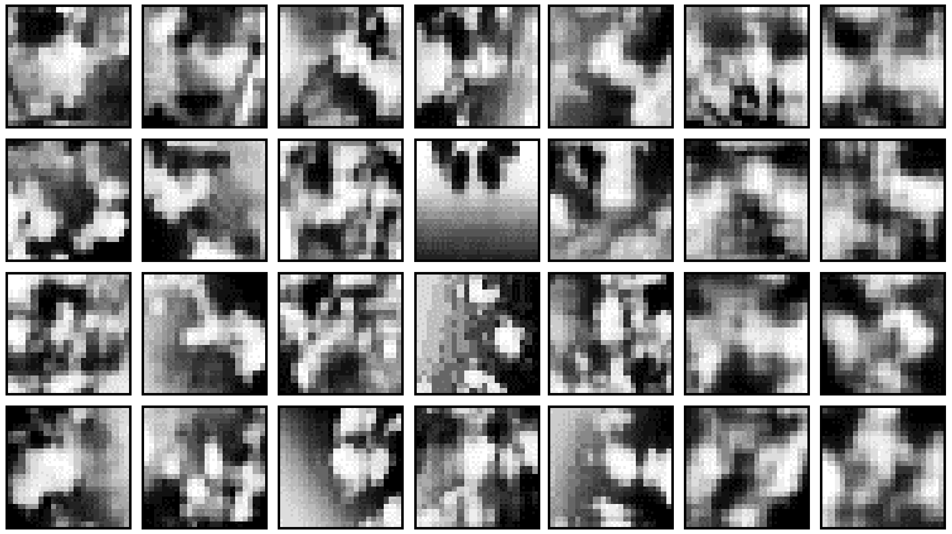
Example face images, randomly mirrored, rotated, translated, and scaled by small amounts (photos are of the three authors).



From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/

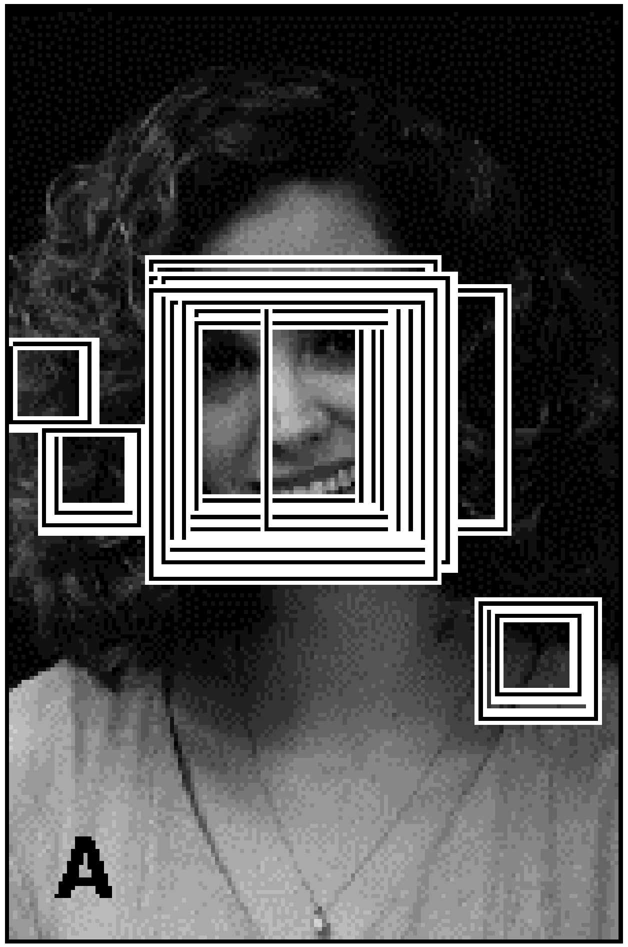
During training, the partially-trained system is applied to images of scenery which do not contain faces (like the one on the left). Any regions in the image detected as faces (which are expanded and shown on the right) are errors, which can be added into the set of negative training examples.

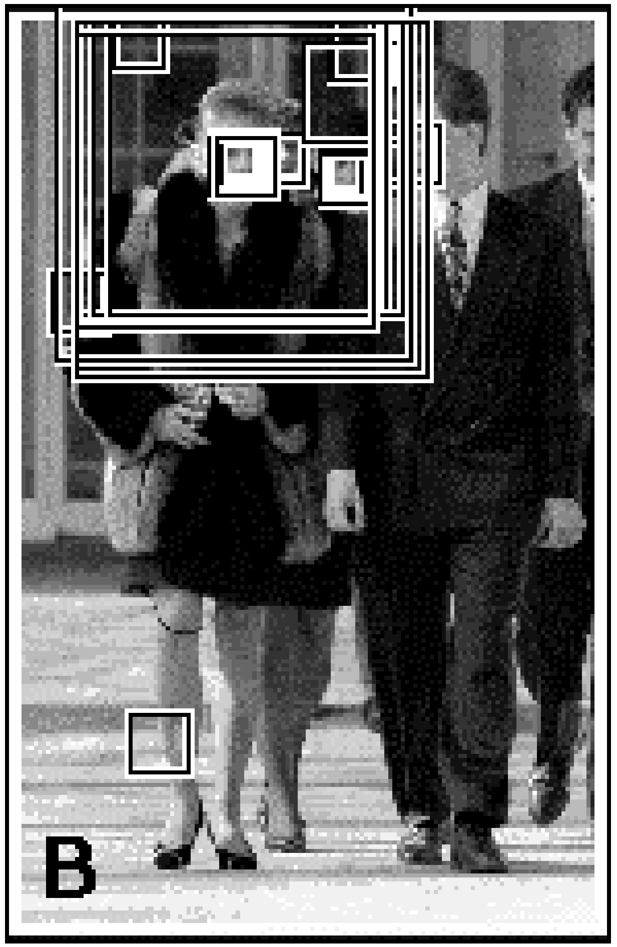


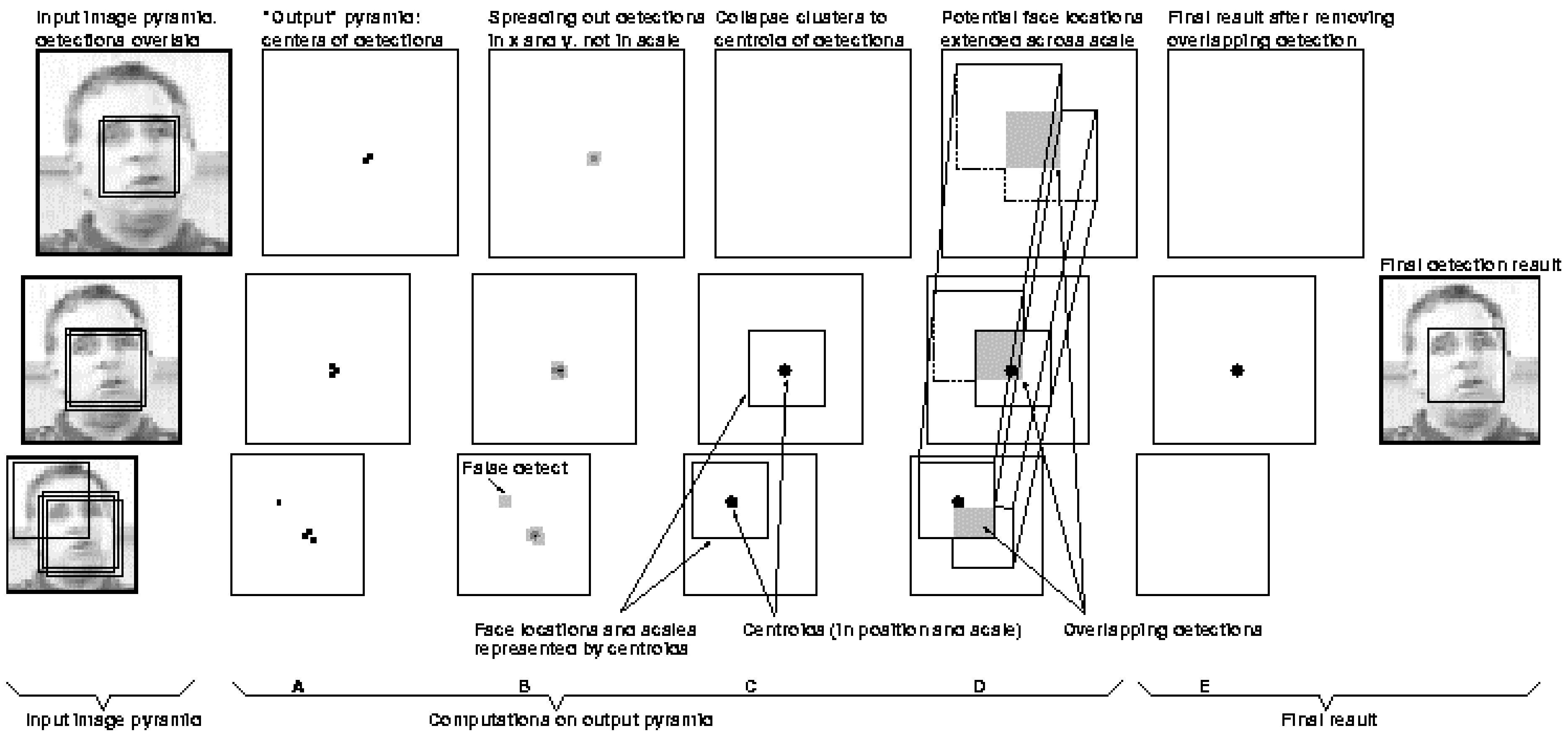


From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/

Images with all the above threshold detections indicated by boxes.

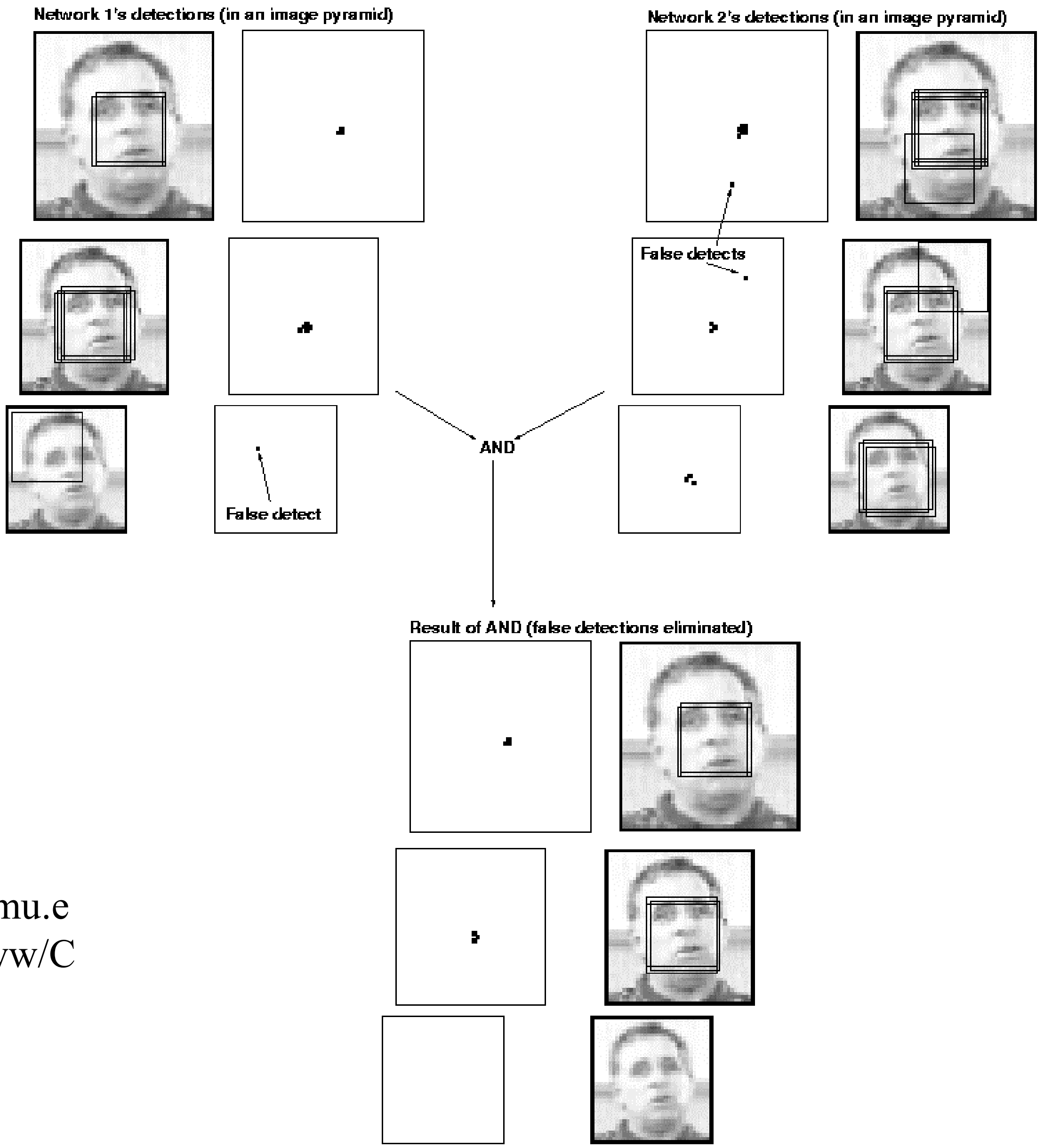





Input image pyramid, detections overlaid
"Output" pyramid: centers of detections
Spreading out detections in x and y, not in scale
Collapse clusters to centroid of detections
Potential face locations extended across scale
Final result after removing overlapping detection
Final detection result
False detect
Face locations and scales represented by centroids
Centroids (in position and scale)
Overlapping detections
A
B
C
D
E
Input image pyramid
Computations on output pyramid
Final result

The framework used for merging multiple detections from a single network: A) The detections are recorded in an image pyramid. B) The detections are ``spread out'' and a threshold is applied. C) The centroids in scale and position are computed, and the regions contributing to each centroid are collapsed to single points. In the example shown, this leaves only two detections in the output pyramid. D) The final step is to check the proposed face locations for overlaps, and E) to remove overlapping detections if they exist. In this example, removing the overlapping detection eliminates what would otherwise be a false positive.

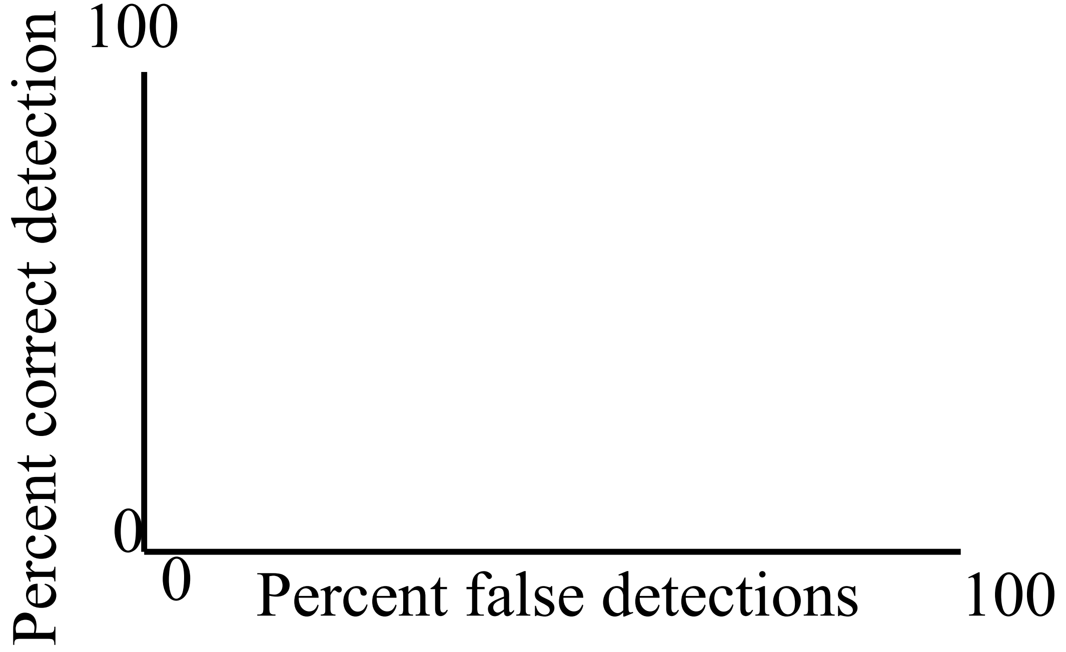
From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/


Network 1's detections (in an image pyramid)
Network 2's detections (in an image pyramid)
False detects
False detect
AND
Result of AND (false detections eliminated)

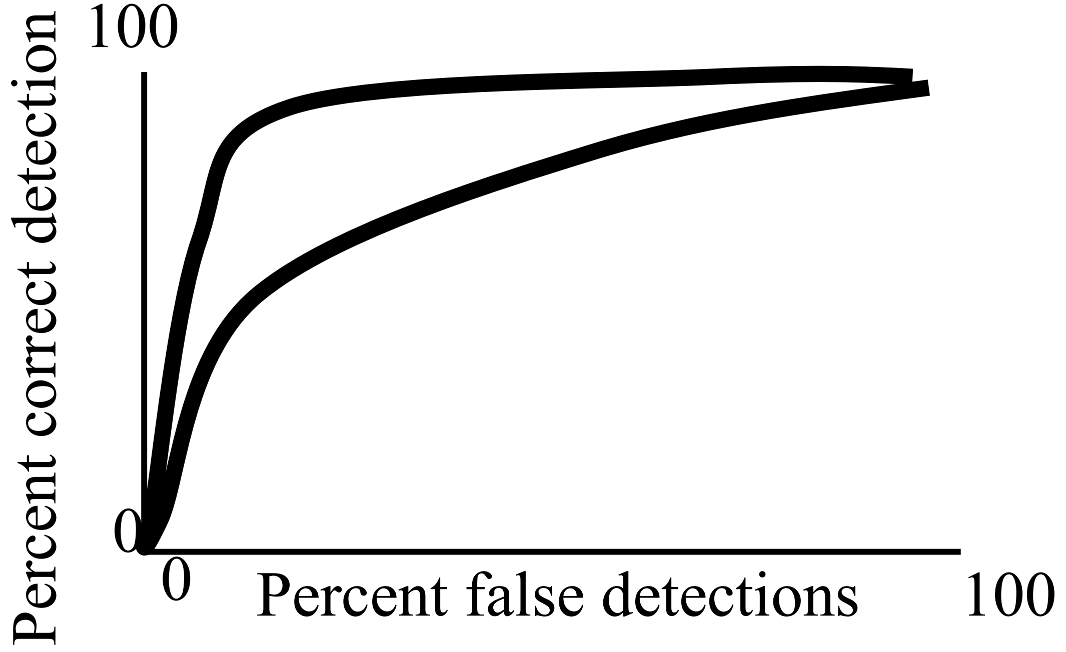
From:
http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/

ANDing together the outputs from two networks over different positions and scales can improve detection accuracy.

ROC (receiver operating characteristic) curve

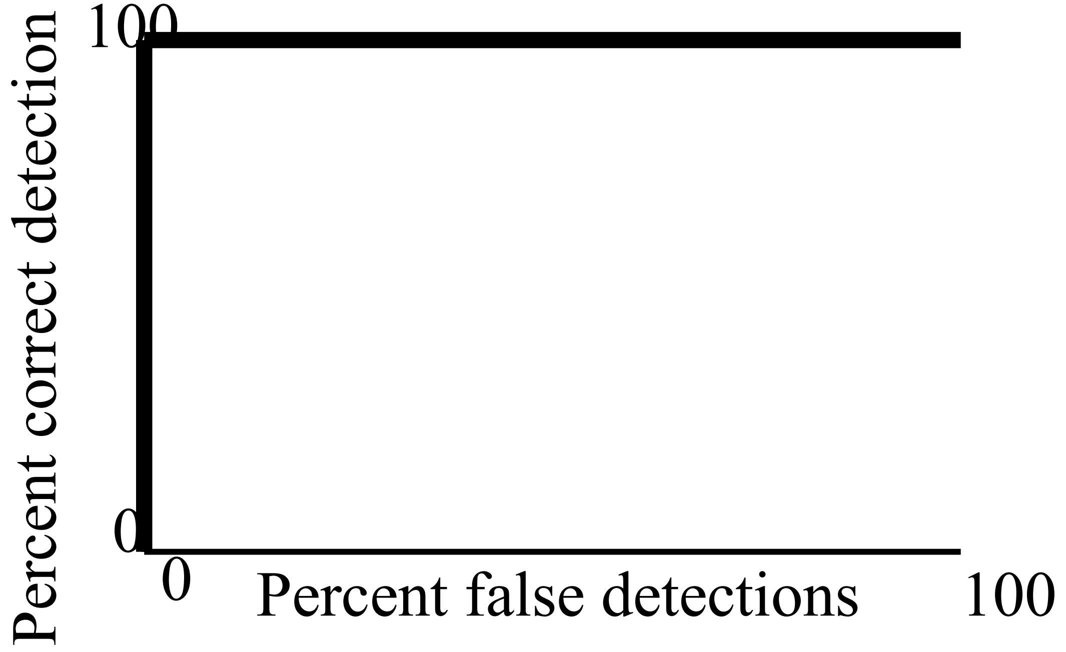


ROC (receiver operating characteristic) curve

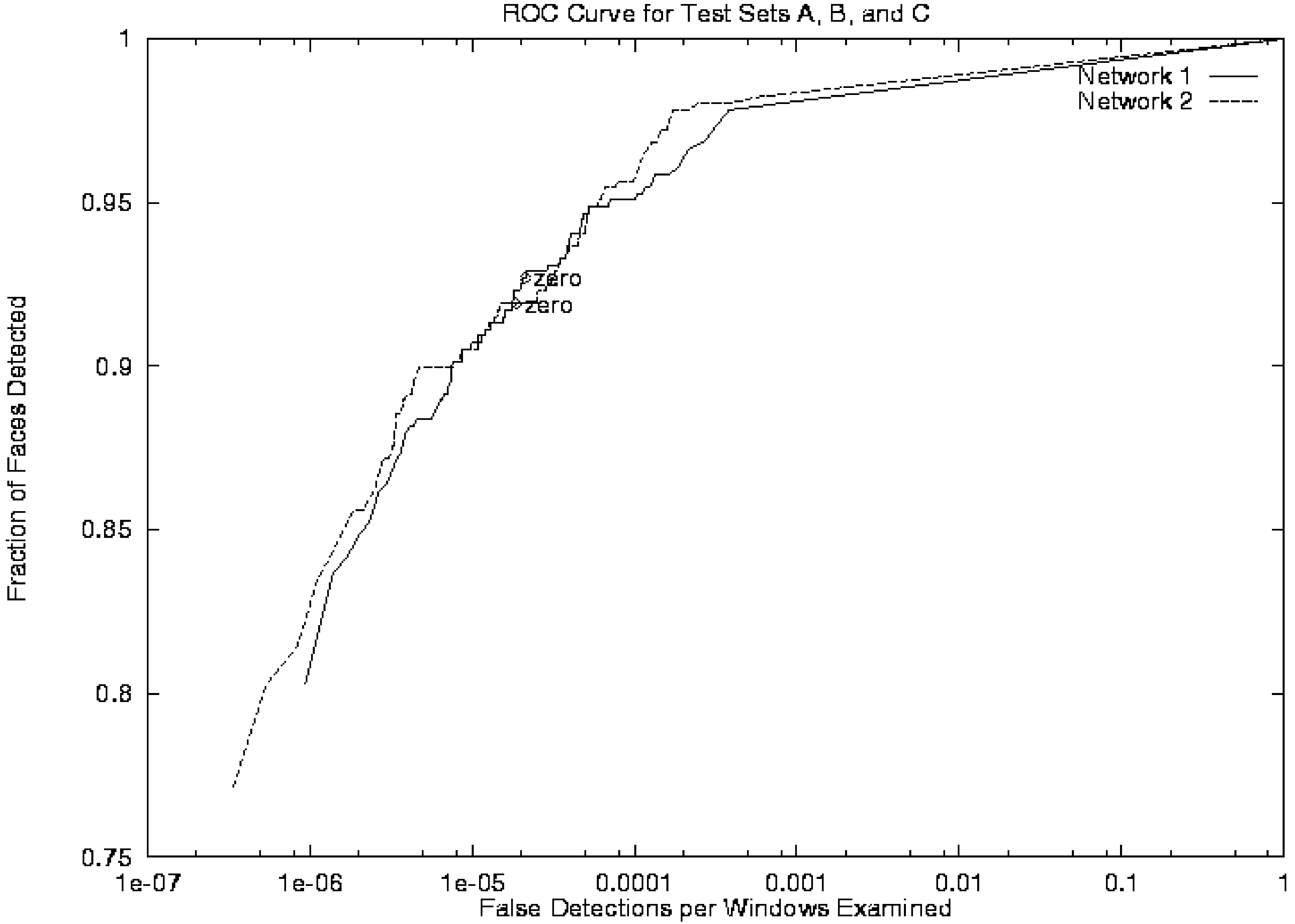


Realistic examples

ROC (receiver operating characteristic) curve



ideal



From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/

http://www.ius.cs.cmu.edu/demos/facedemo.html

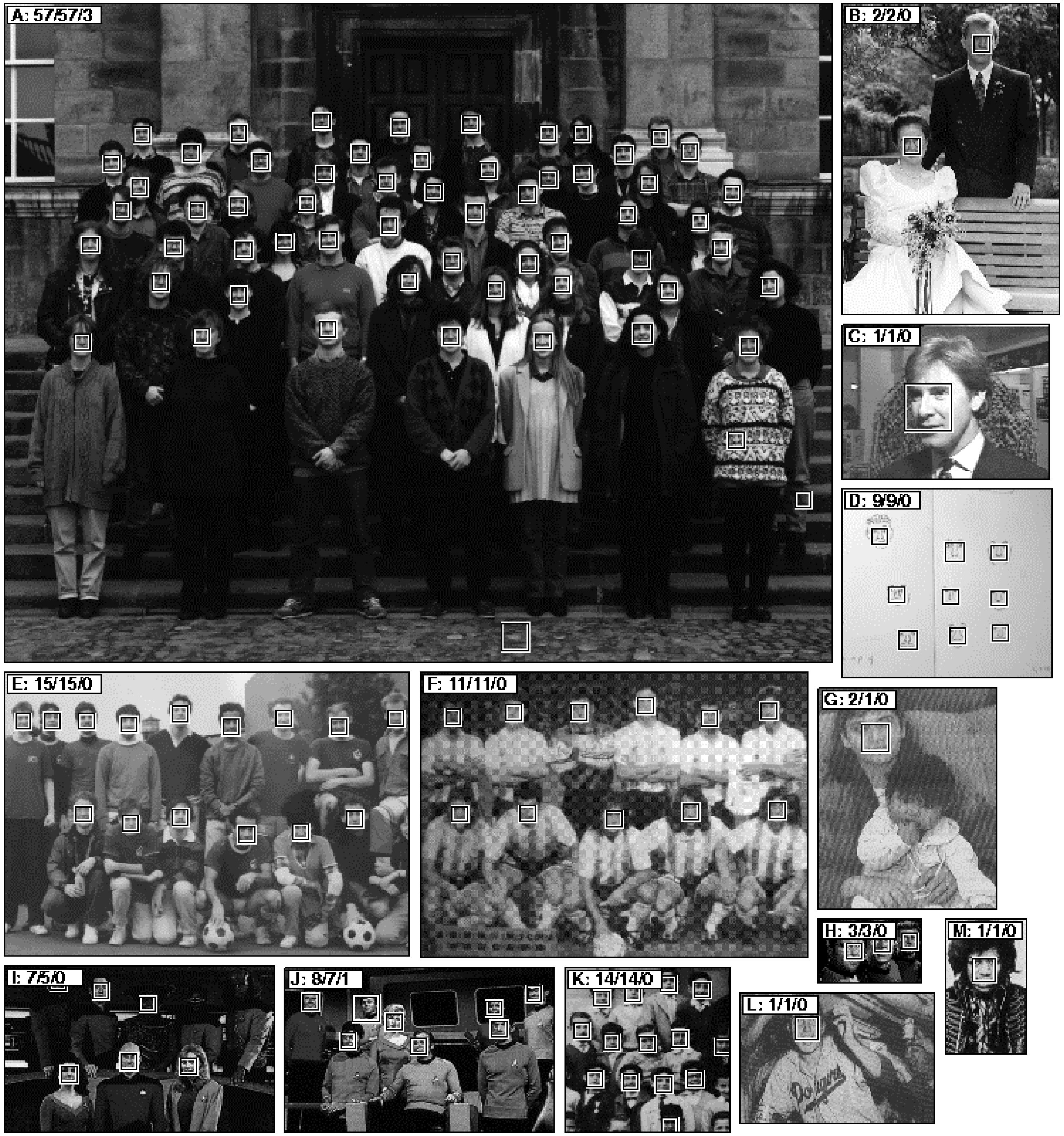
CMU's Face Detector Demo

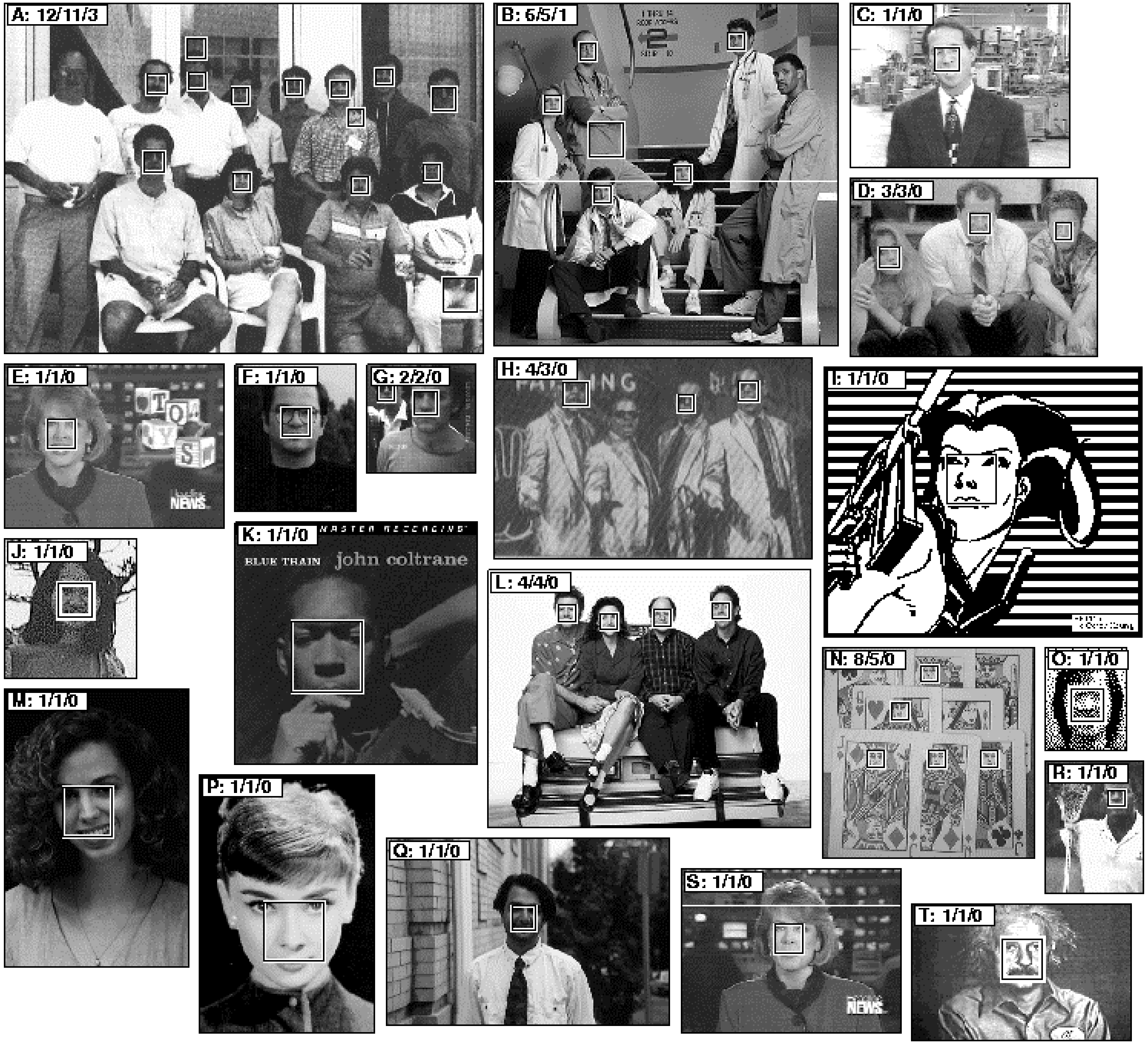
This is the front page for an interactive WWW demonstration of a face detector developed here at CMU. A detailed description of the system is available. The face detector can handle pictures of people (roughly) facing the camera in an (almost) vertical orientation. The faces can be anywhere inside the image, and range in size from at least 20 pixels hight to covering the whole image.

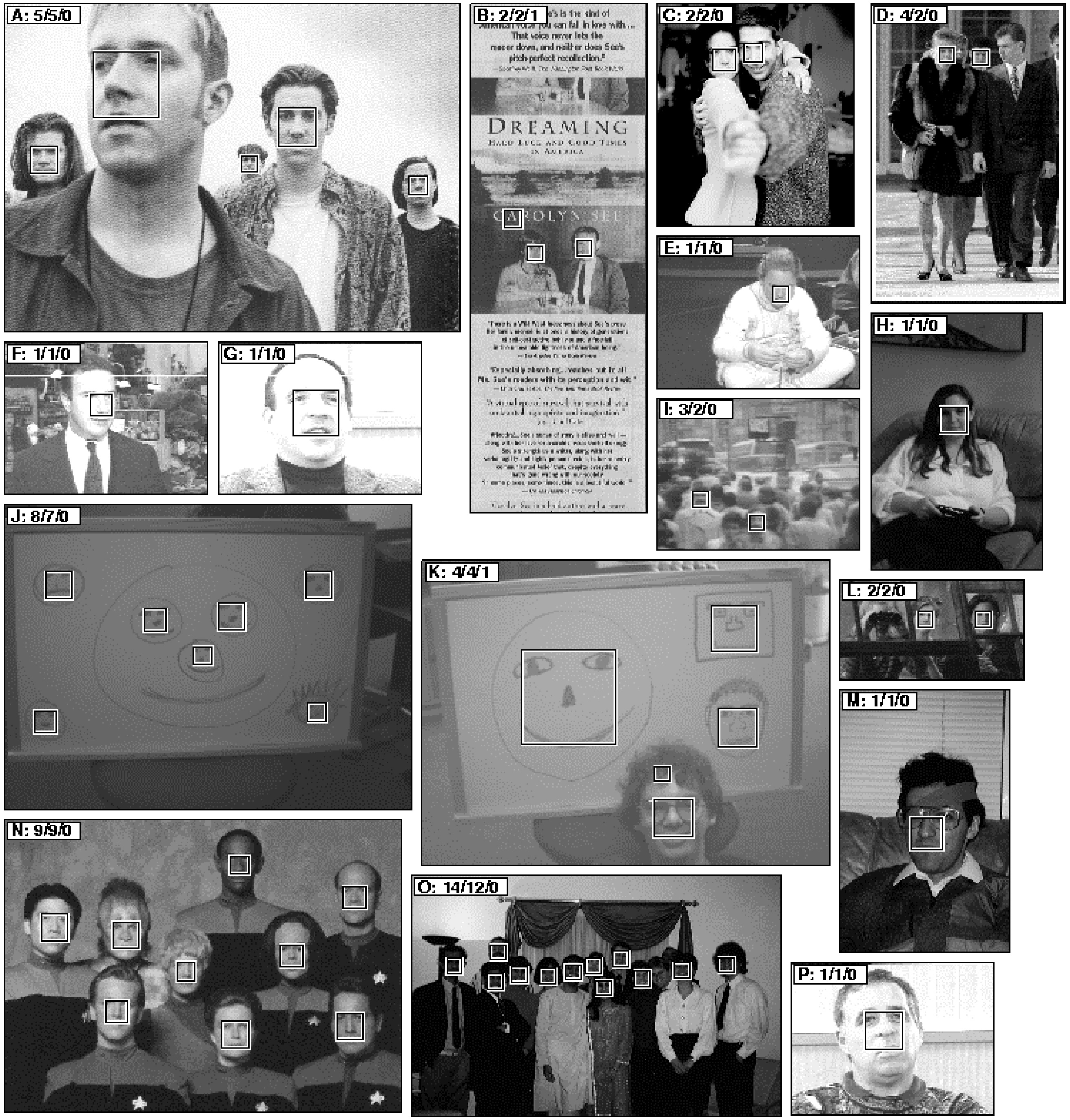
Since the system does not run in real time, this demonstration is organized as follows. First, you can submit an image to be processed by the system. Your image may be located anywhere on the WWW. After your image is processed, you will be informed via an e-mail message.

After your image is processed, you may view it in the gallery (gallery with inlined images). There, you can see your image, with green outlines around each location that the system thinks contains a face. You can also look at the results of the system on images supplied by other people.

Henry A. Rowley (har@cs.cmu.edu)
Shumeet Baluja (baluja@cs.cmu.edu)
Takeo Kanade (tk@cs.cmu.edu)


A: 57/57/3
B: 2/2/0
C: 1/1/0
D: 9/9/0
E: 15/15/0
F: 11/11/0
G: 2/1/0
H: 3/3/0
M: 1/1/0
I: 7/5/0
J: 8/7/1
K: 14/14/0
L: 1/1/0


A: 12/11/3
B: 6/5/1
C: 1/1/0
D: 3/3/0
E: 1/1/0
F: 1/1/0
G: 2/2/0
H: 4/3/0
I: 1/1/0
J: 1/1/0
K: 1/1/0
MASTER RECORDING
BLUE TRAIN
john coltrane
L: 4/4/0
M: 1/1/0
N: 8/5/0
O: 1/1/0
P: 1/1/0
Q: 1/1/0
R: 1/1/0
S: 1/1/0
T: 1/1/0
NEWS



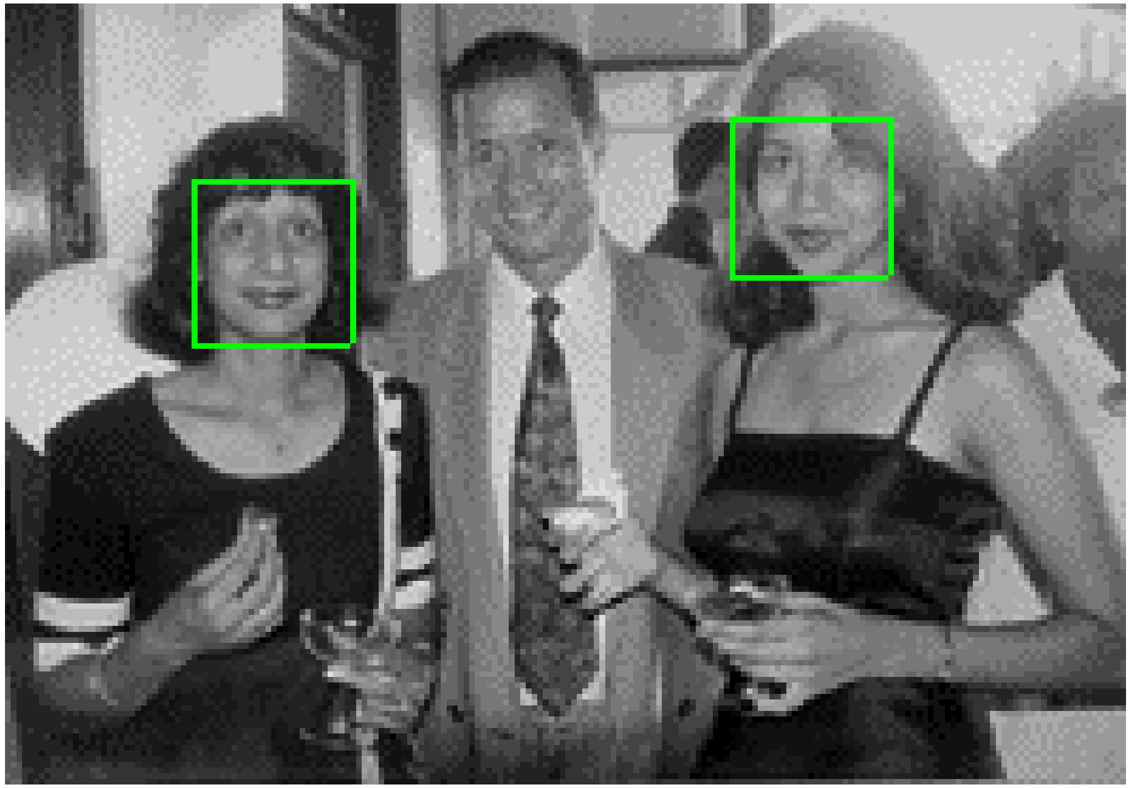
A: 5/5/0
B: 2/2/1
DREAMING
CAROLYN SEE
C: 2/2/0
D: 4/2/0
E: 1/1/0
F: 1/1/0
G: 1/1/0
H: 1/1/0
I: 3/2/0
J: 8/7/0
K: 4/4/1
L: 2/2/0
M: 1/1/0
N: 9/9/0
O: 14/12/0
P: 1/1/0

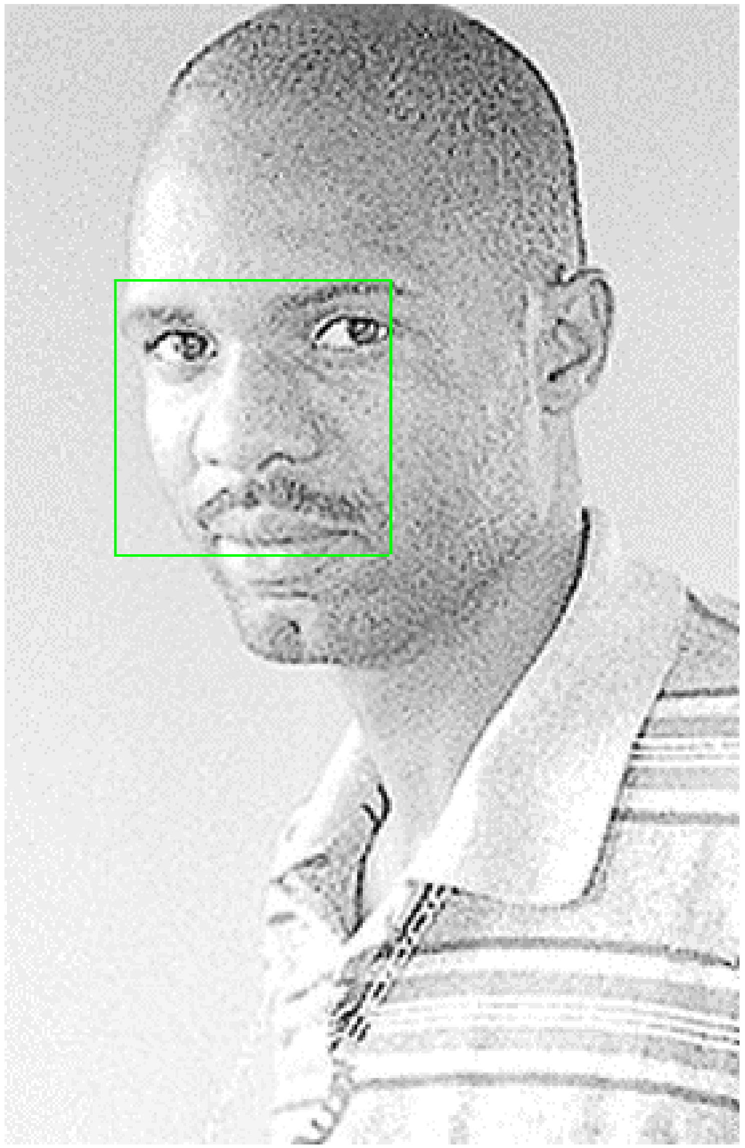
Example CMU face detector results

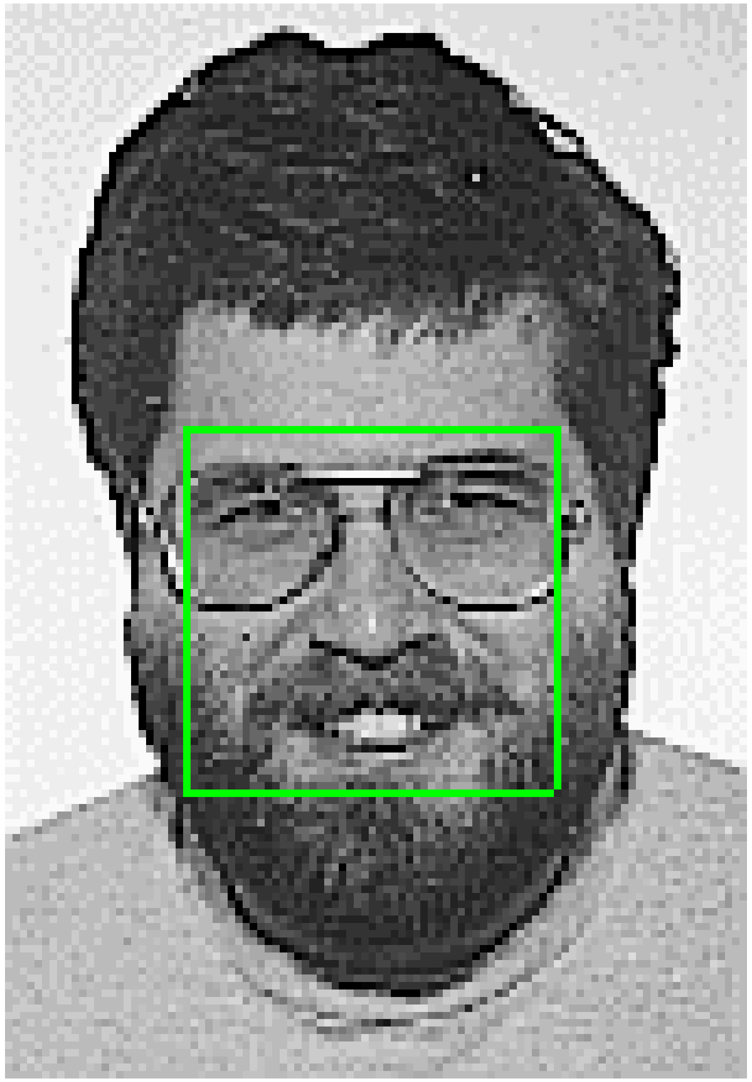
input

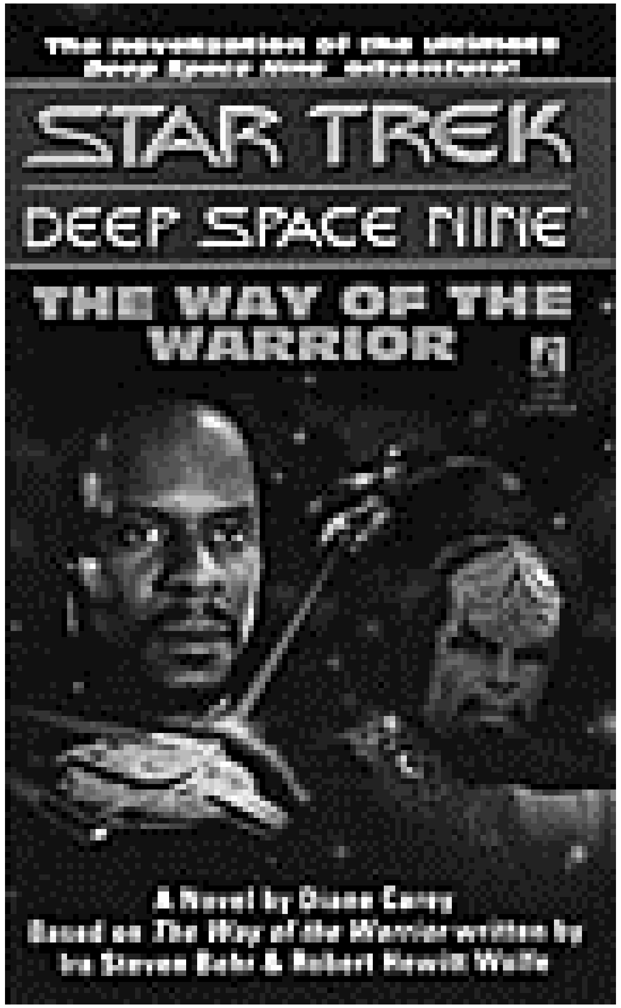
All images from: http://www.ius.cs.cmu.edu/demos/facedemo.html

output









The novelization of the ultimate *Deep Space Nine* adventure!

STAR TREK

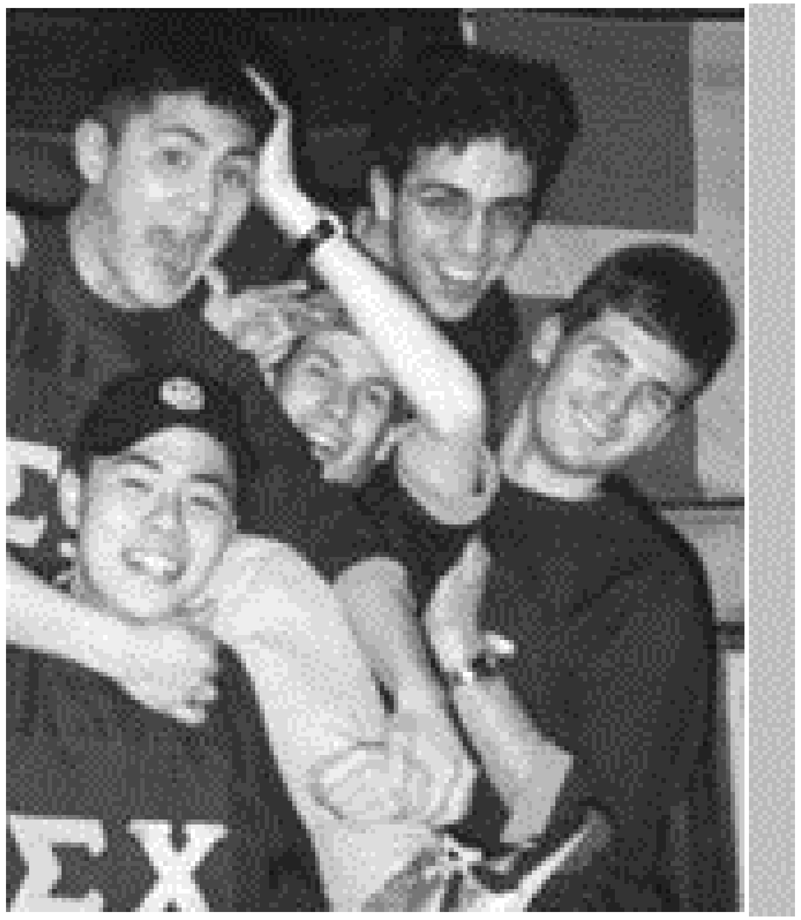
DEEP SPACE NINE

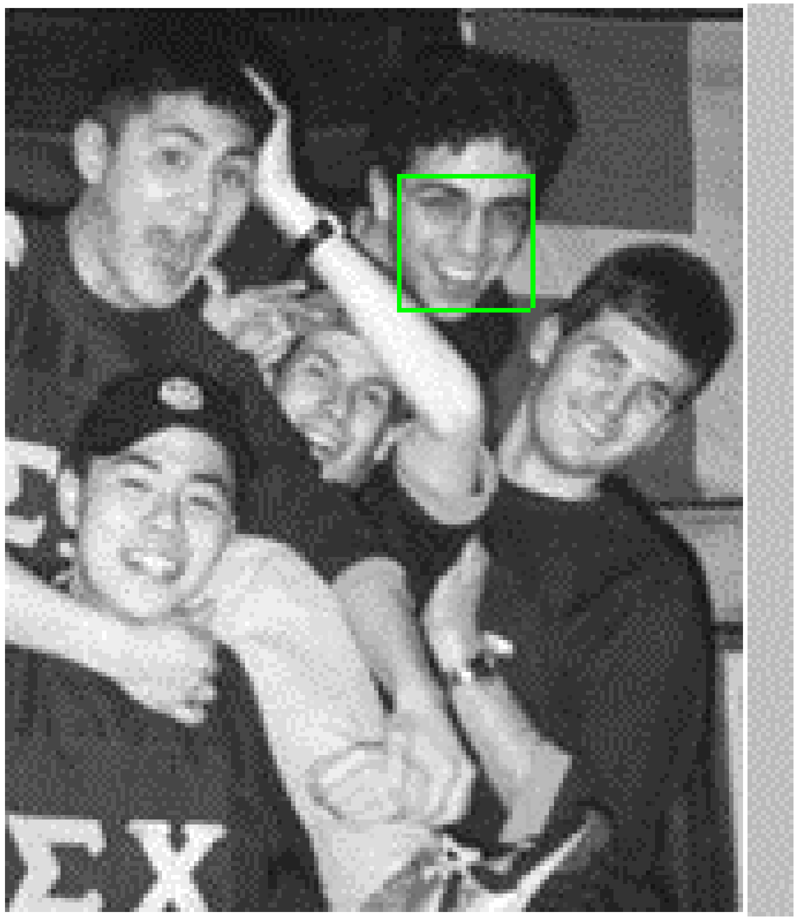
THE WAY OF THE WARRIOR

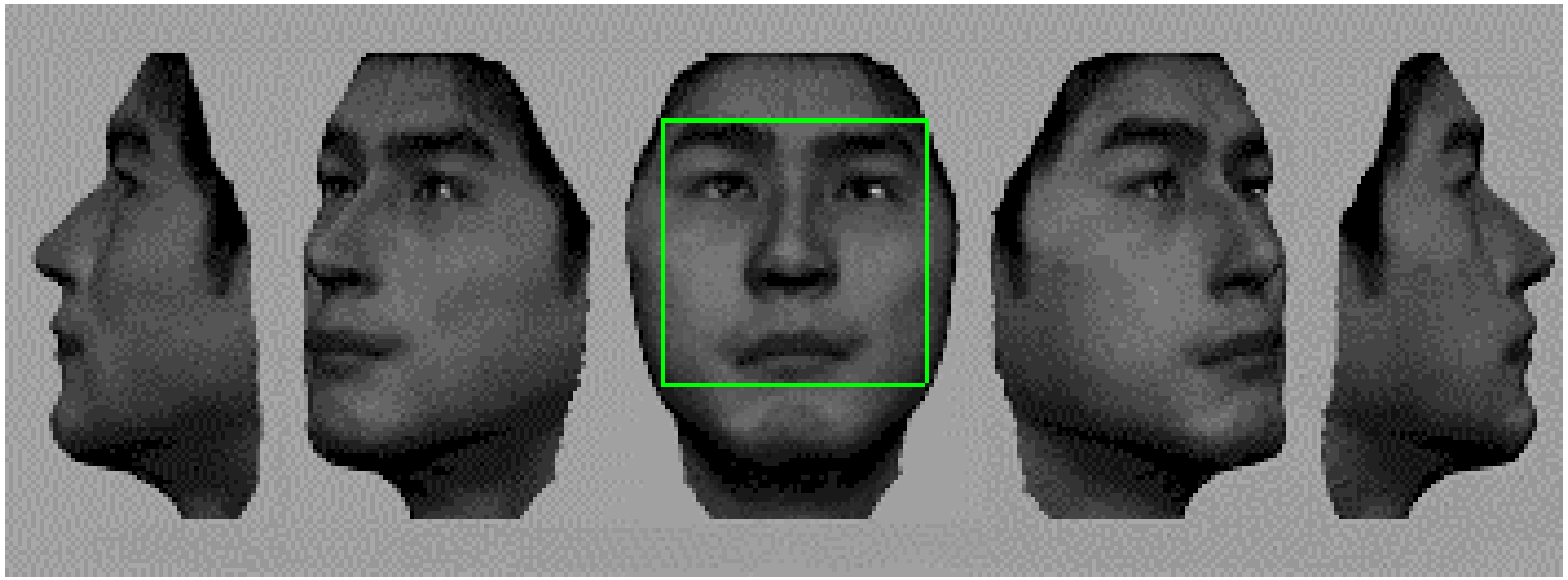
A Novel by Diane Carey

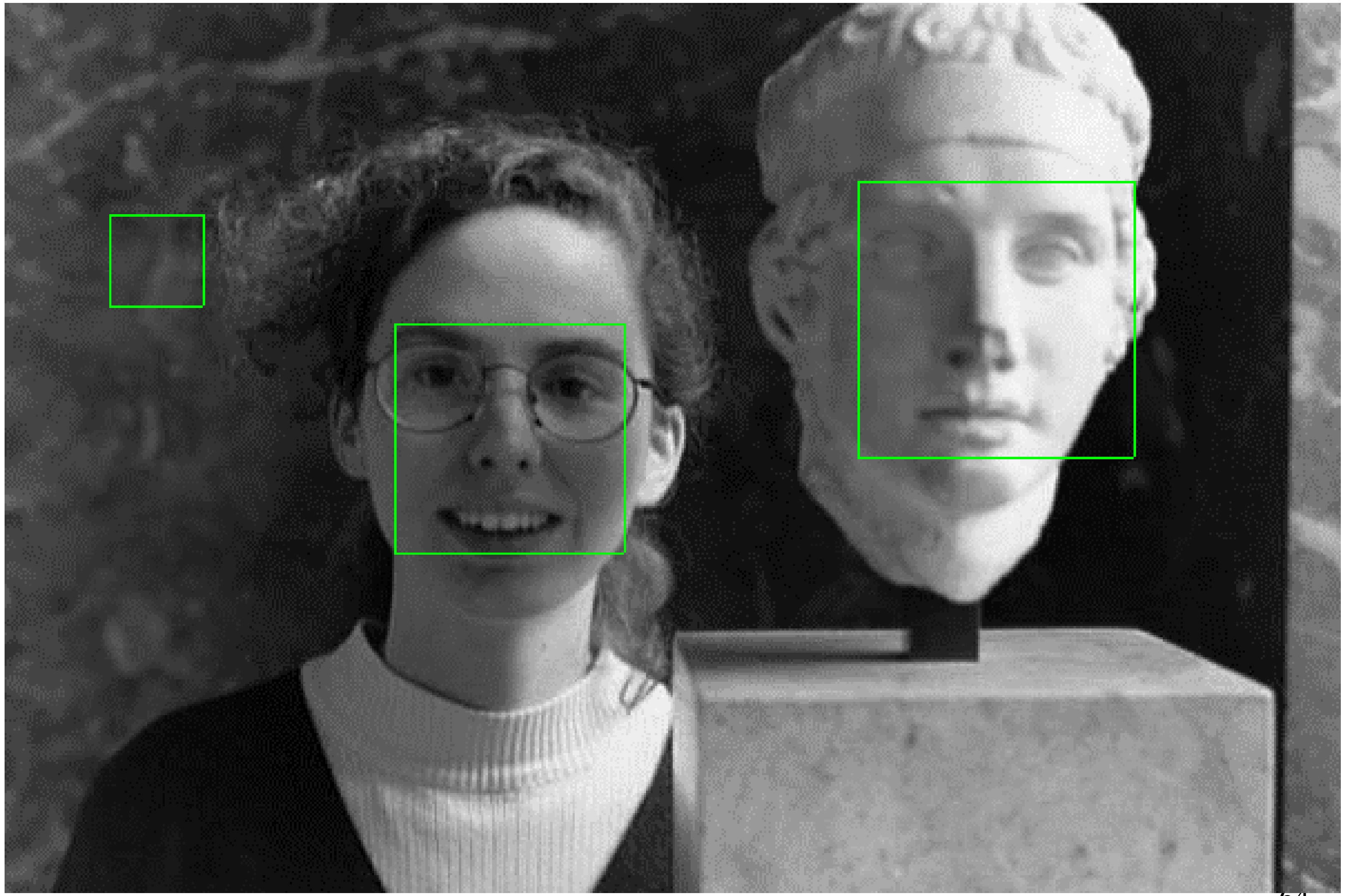
Based on *The Way of the Warrior* written by Ira Steven Behr & Robert Hewitt Wolfe

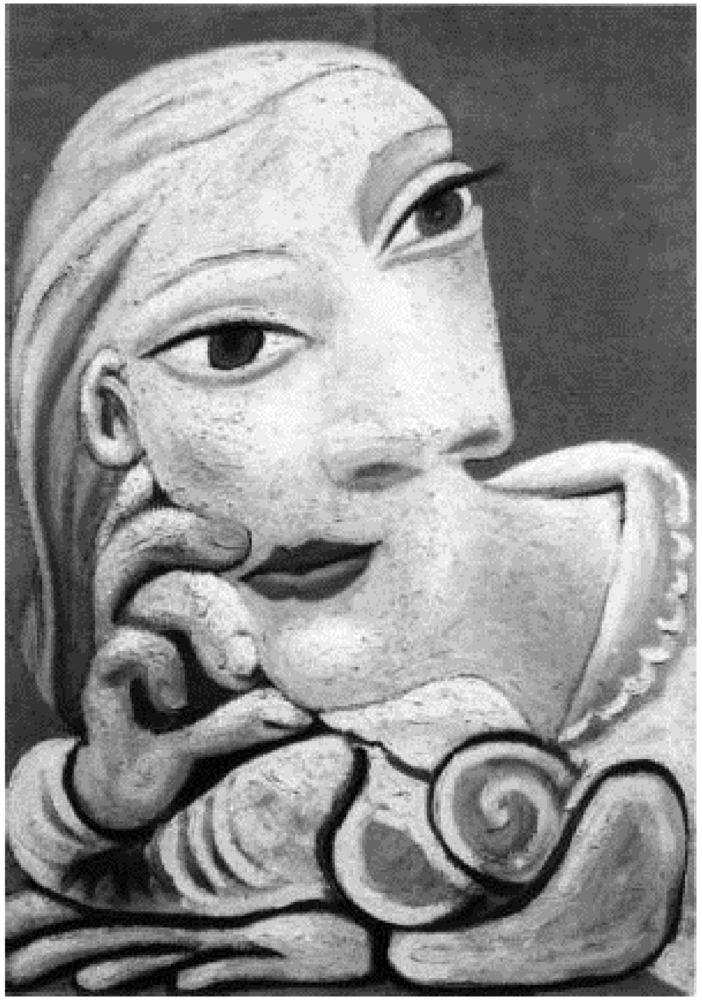
The novelization of the ultimate Deep Space Nine adventure!
STAR TREK
DEEP SPACE NINE
THE WAY OF THE WARRIOR
A Novel by Diane Carey
Based on The Way of the Warrior written by
Ira Steven Behr & Robert Hewitt Wolfe

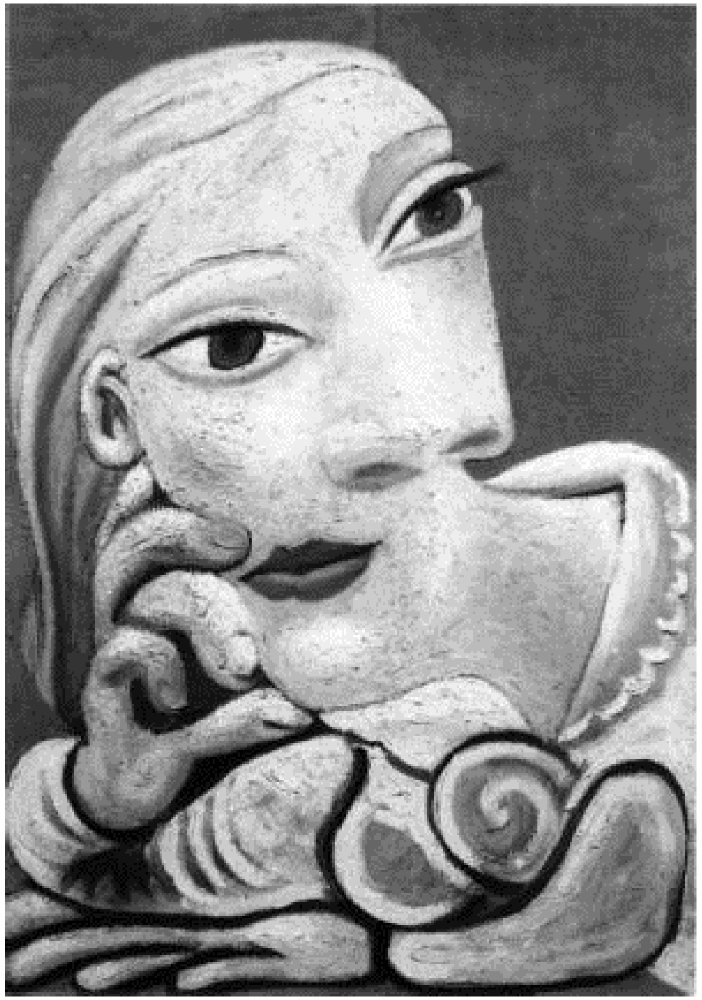

ΣΧ

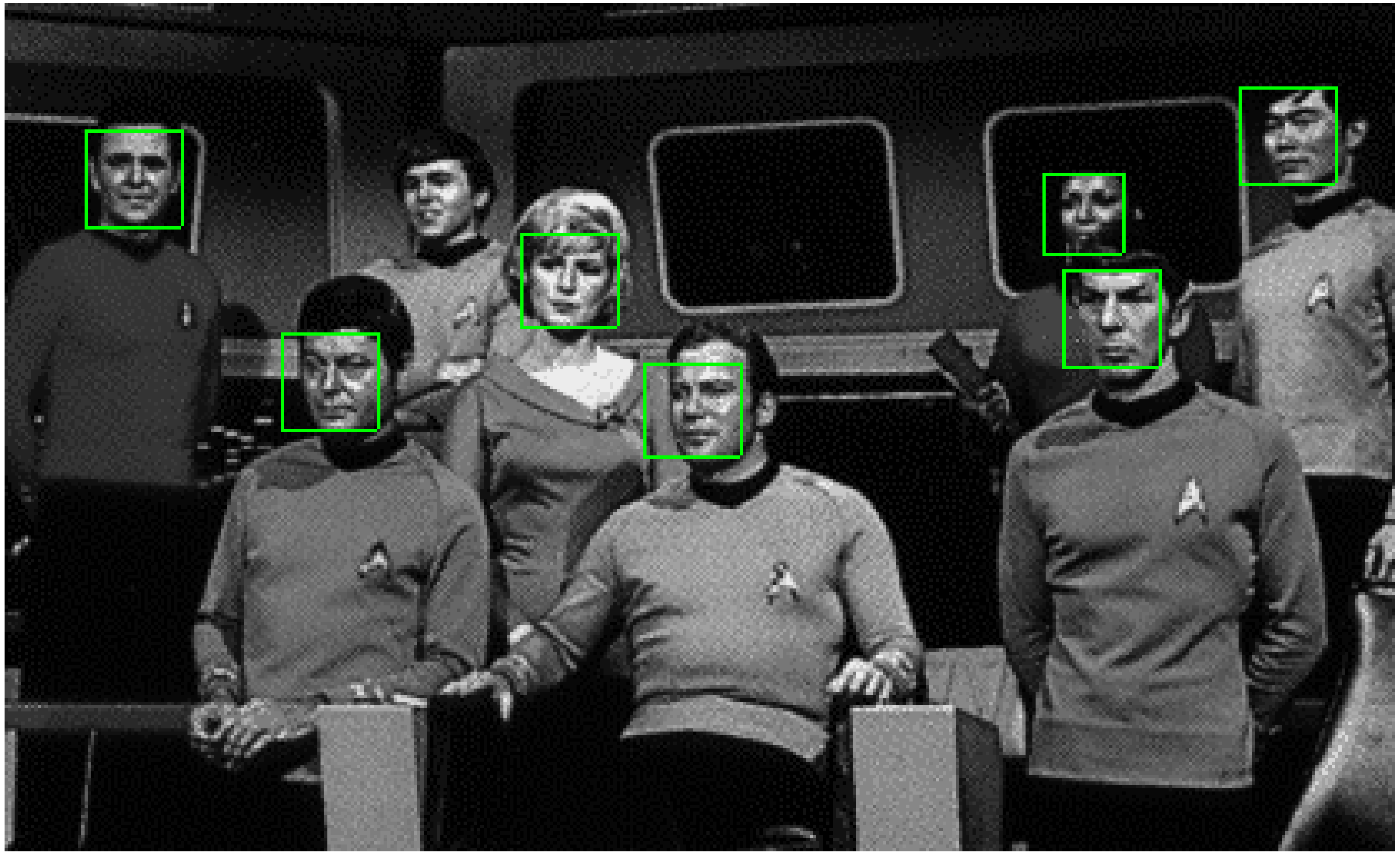

ΣΧ

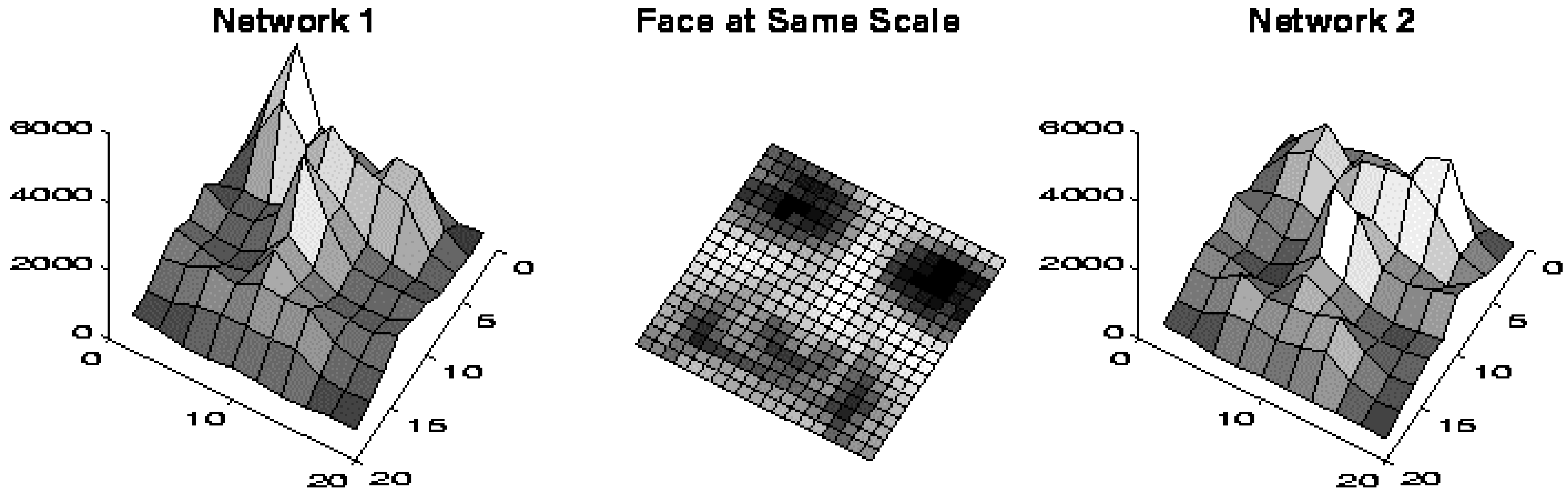












Error rates (vertical axis) on a small test resulting from adding noise to various portions of the input image (horizontal plane), for two networks. Network 1 has two copies of the hidden units shown in Figure 1 (a total of 58 hidden units and 2905 connections), while Network 2 has three copies (a total of 78 hidden units and 4357 connections).

The networks rely most heavily on the eyes, then on the nose, and then on the mouth (Figure 9). Anecdotally, we have seen this behavior on several real test images. Even in cases in which only one eye is visible, detection of a face is possible, though less reliable, than when the entire face is visible. The system is less sensitive to the occlusion of features such as the nose or mouth.

From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/

Support vector machines (SVM’s)

- The 3 good ideas of SVM’s

Good idea #1: Classify rather than model probability distributions.

- Advantages:
  - Focuses the computational resources on the task at hand.
- Disadvantages:
  - Don’t know how probable the classification is
  - Lose the probabilistic model for each object class; can’t draw samples from each object class.

Good idea #2: Wide margin classification

- For better generalization, you want to use the weakest function you can.
  - Remember polynomial fitting.
- There are fewer ways a wide-margin hyperplane classifier can split the data than an ordinary hyperplane classifier.

Too weak

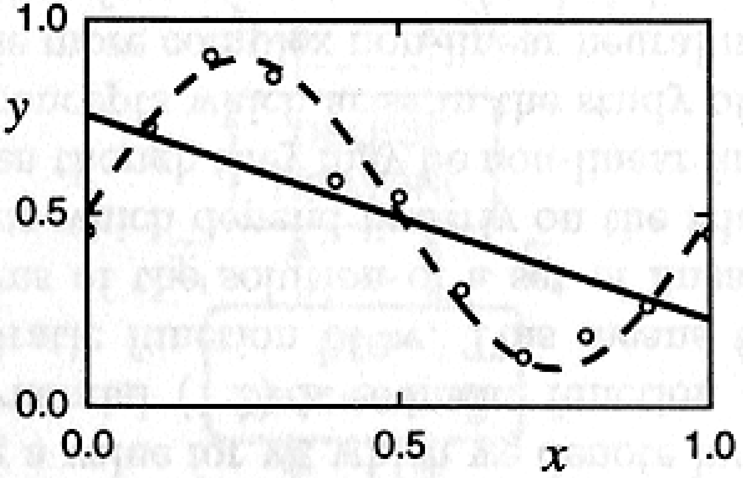


Figure 1.6. An example of a set of 11 data points obtained by sampling the function $h(x)$, defined by (1.4), at equal intervals of x and adding random noise. The dashed curve shows the function $h(x)$, while the solid curve shows the rather poor approximation obtained with a linear polynomial, corresponding to $M = 1$ in (1.2).

Bishop, neural networks for pattern recognition, 1995

Just right

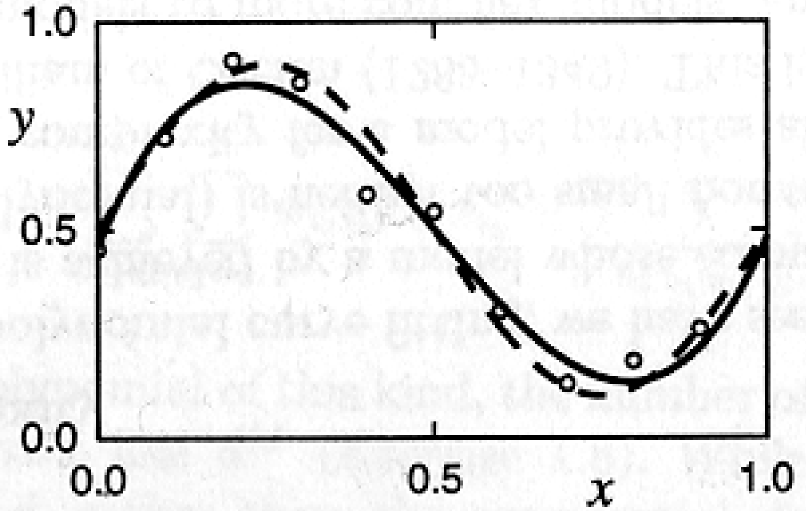


Figure 1.7. This shows the same data set as in Figure 1.6, but this time fitted by a cubic ($M = 3$) polynomial, showing the significantly improved approximation to $h(x)$ achieved by this more flexible function.

Bishop, neural networks for pattern recognition, 1995

Too strong

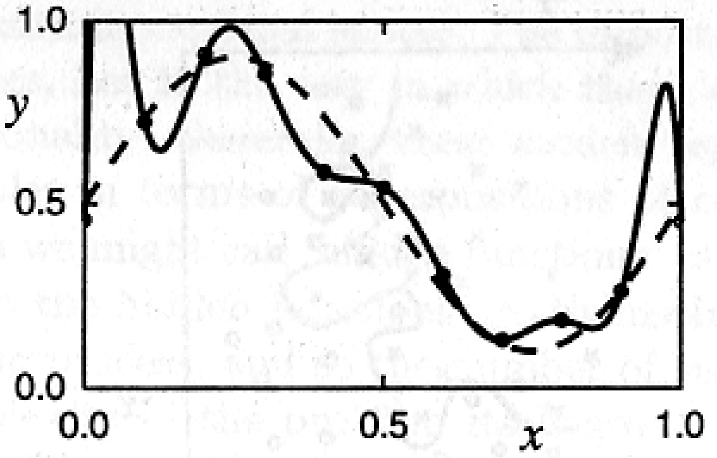
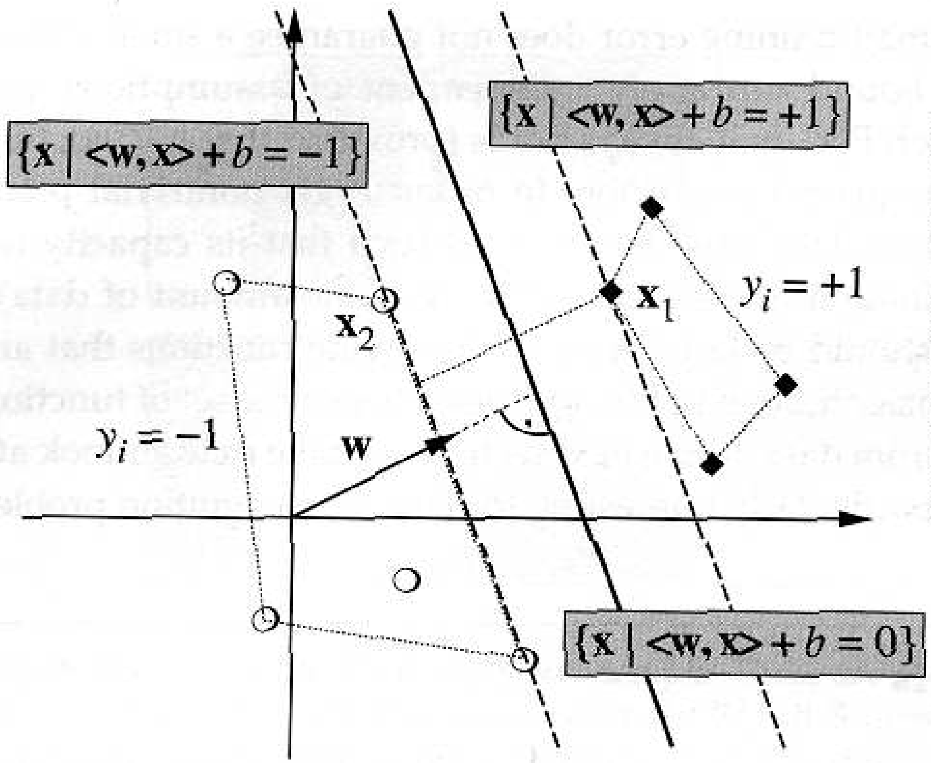


Figure 1.8. The result of fitting the same data set as in Figure 1.6 using a 10th-order ($M = 10$) polynomial. This gives a perfect fit to the training data, but at the expense of a function which has large oscillations, and which therefore gives a poorer representation of the generator function $h(x)$ than did the cubic polynomial of Figure 1.7.

Bishop, neural networks for pattern recognition, 1995



$\{x \mid \langle w, x\rangle + b = -1\}$
$\{x \mid \langle w, x\rangle + b = +1\}$
$\{x \mid \langle w, x\rangle + b = 0\}$
x_1
x_2
$y_i = +1$
$y_i = -1$
w

Note:

$$\langle w, x_1\rangle + b = +1$$
$$\langle w, x_2\rangle + b = -1$$
$$\Rightarrow \quad \langle w, (x_1 - x_2)\rangle = 2$$
$$\Rightarrow \quad \left\langle \frac{w}{\|w\|}, (x_1 - x_2)\right\rangle = \frac{2}{\|w\|}$$

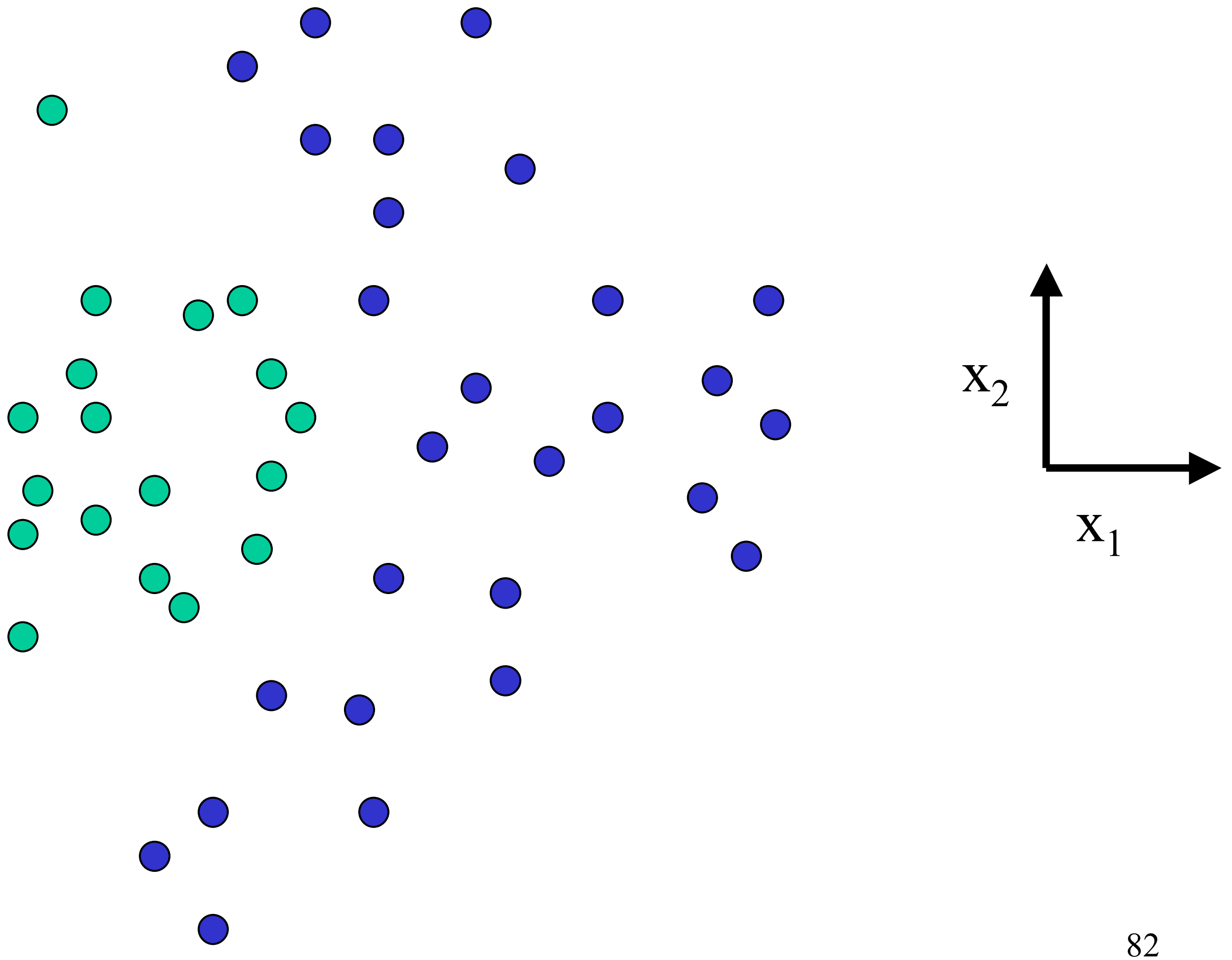
Figure 1.5 A binary classification toy problem: separate balls from diamonds. The *optimal hyperplane* (1.23) is shown as a solid line. The problem being separable, there exists a weight vector $\mathbf{w}$ and a threshold b such that $y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) > 0$ $(i = 1, \ldots, m)$. Rescaling $\mathbf{w}$ and b such that the point(s) closest to the hyperplane satisfy $|\langle \mathbf{w}, \mathbf{x}_i\rangle + b| = 1$, we obtain a *canonical* form $(\mathbf{w}, b)$ of the hyperplane, satisfying $y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1$. Note that in this case, the *margin* (the distance of the closest point to the hyperplane) equals $1/\|\mathbf{w}\|$. This can be seen by considering two points $\mathbf{x}_1, \mathbf{x}_2$ on opposite sides of the margin, that is, $\langle \mathbf{w}, \mathbf{x}_1\rangle + b = 1$, $\langle \mathbf{w}, \mathbf{x}_2\rangle + b = -1$, and projecting them onto the hyperplane normal vector $\mathbf{w}/\|\mathbf{w}\|$.

Learning with Kernels, Scholkopf and Smola, 2002

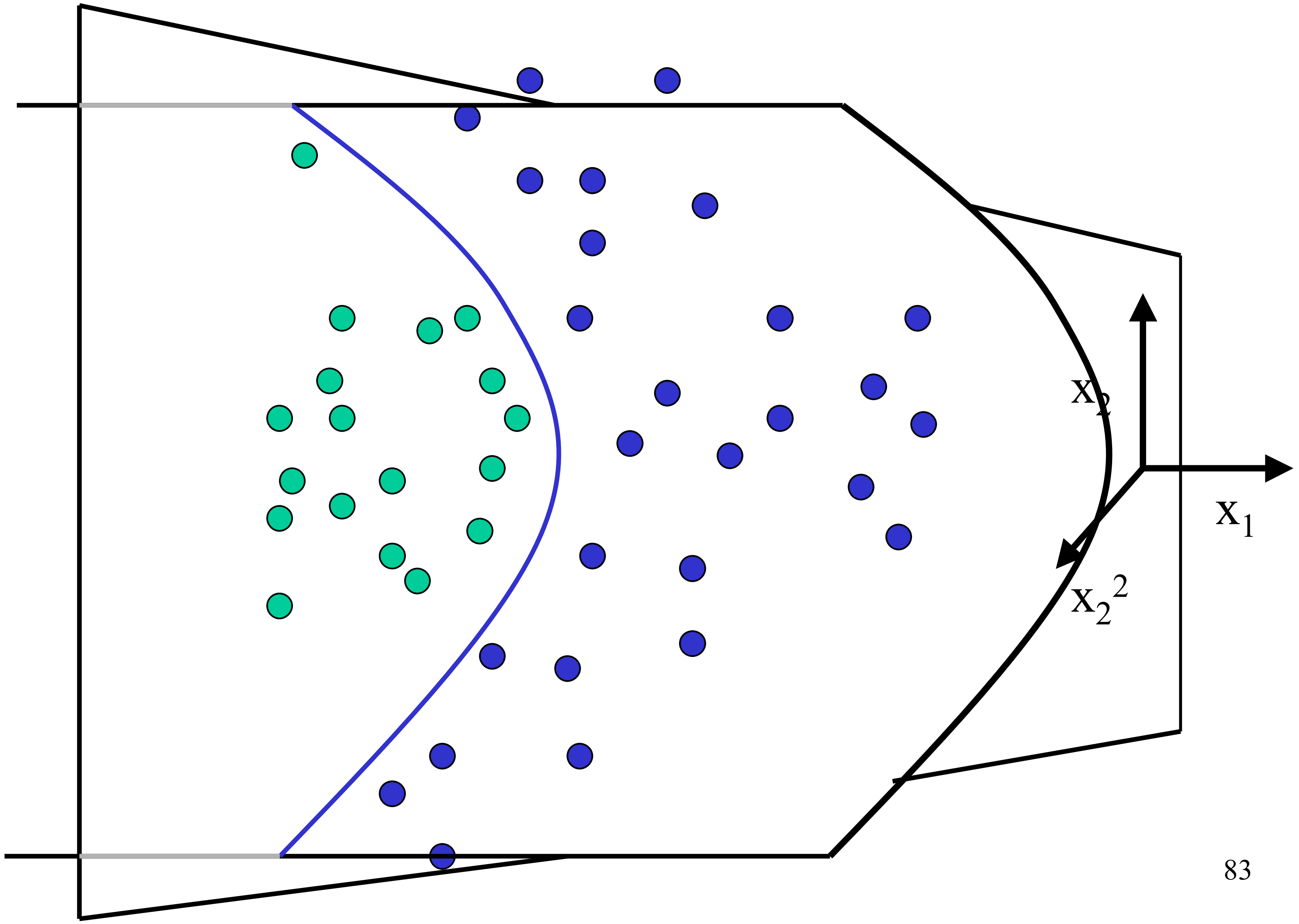
Finding the wide-margin separating hyperplane: a quadratic programming problem, involving inner products of data vectors

Good idea #3: The kernel trick

Non-separable by a hyperplane in 2-d



Separable by a hyperplane in 3-d



Embedding

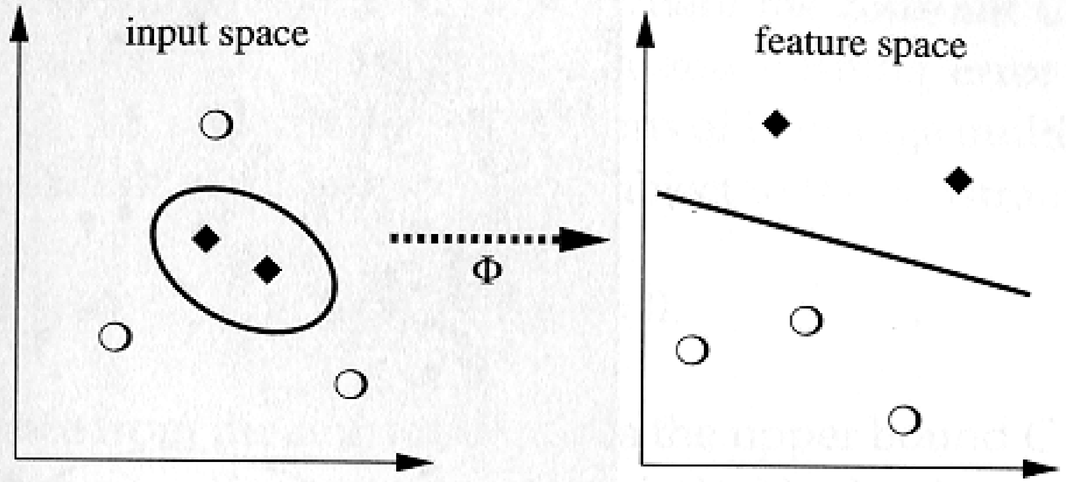


Figure 1.6 The idea of SVMs: map the training data into a higher-dimensional feature space via Φ, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of a kernel function (1.2), it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.

The idea

- There are many embeddings were the dot product in the high dimensional space is just the kernel function applied to the dot product in the low-dimensional space.
- For example:
  - $K(x,x') = (\langle x,x' \rangle + 1)^d$
- Then you "forget" about the high dimensional embedding, and just play with different kernel functions.

Example kernel functions

- Polynomials
- Gaussians
- Sigmoids
- Radial basis functions
- Etc…

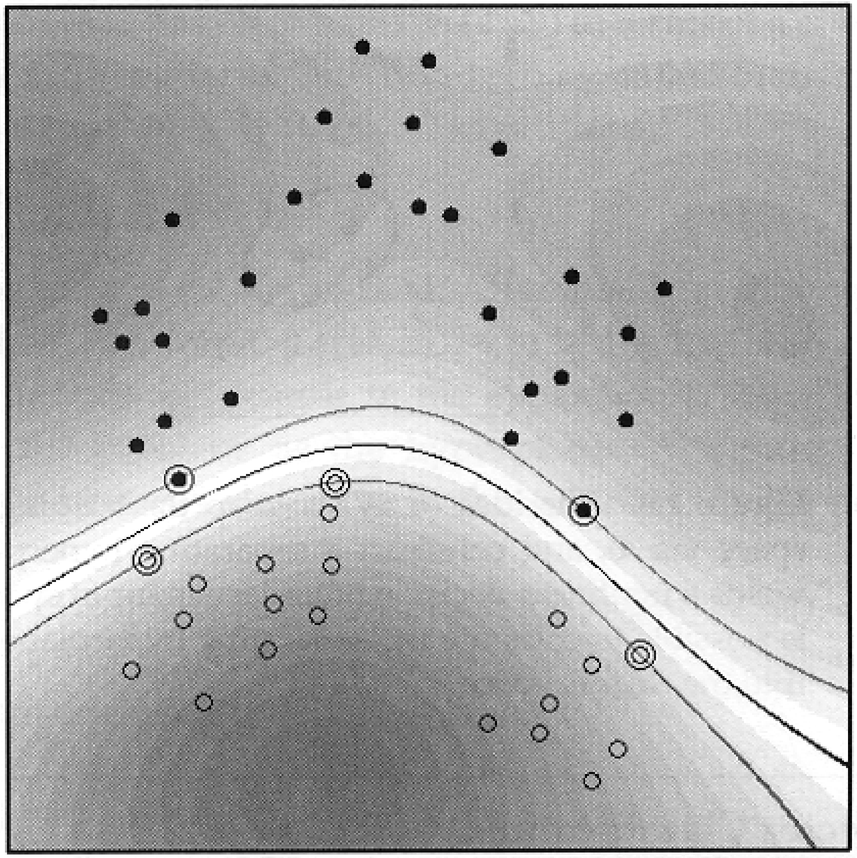
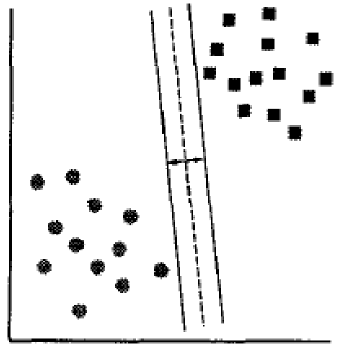
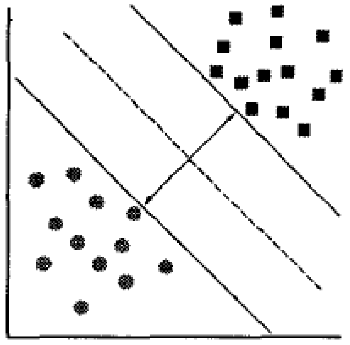


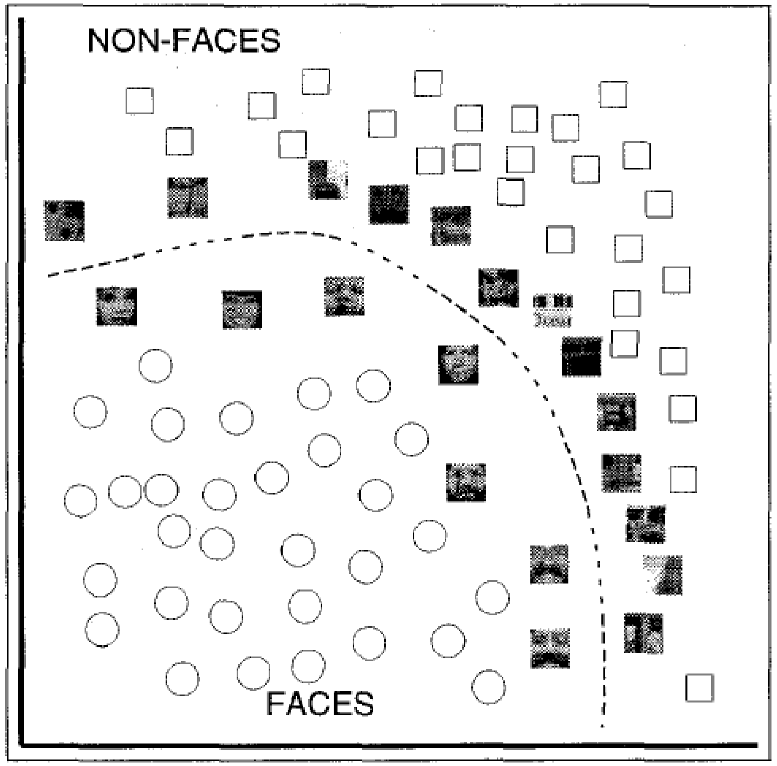
Figure 1.7 Example of an SV classifier found using a radial basis function kernel $k(x, x') = \exp(-\|x - x'\|^2)$ (here, the input space is $\mathcal{X} = [-1, 1]^2$). Circles and disks are two classes of training examples; the middle line is the decision surface; the outer lines precisely meet the constraint (1.25). Note that the SVs found by the algorithm (marked by extra circles) are not centers of clusters, but examples which are critical for the given classification task. Gray values code $|\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b|$, the modulus of the argument of the decision function (1.35). The top and the bottom lines indicate places where it takes the value 1 (from [471]).

Learning with Kernels, Scholkopf and Smola, 2002

Discriminative approaches: e.g., Support Vector Machines


(a)


(b)



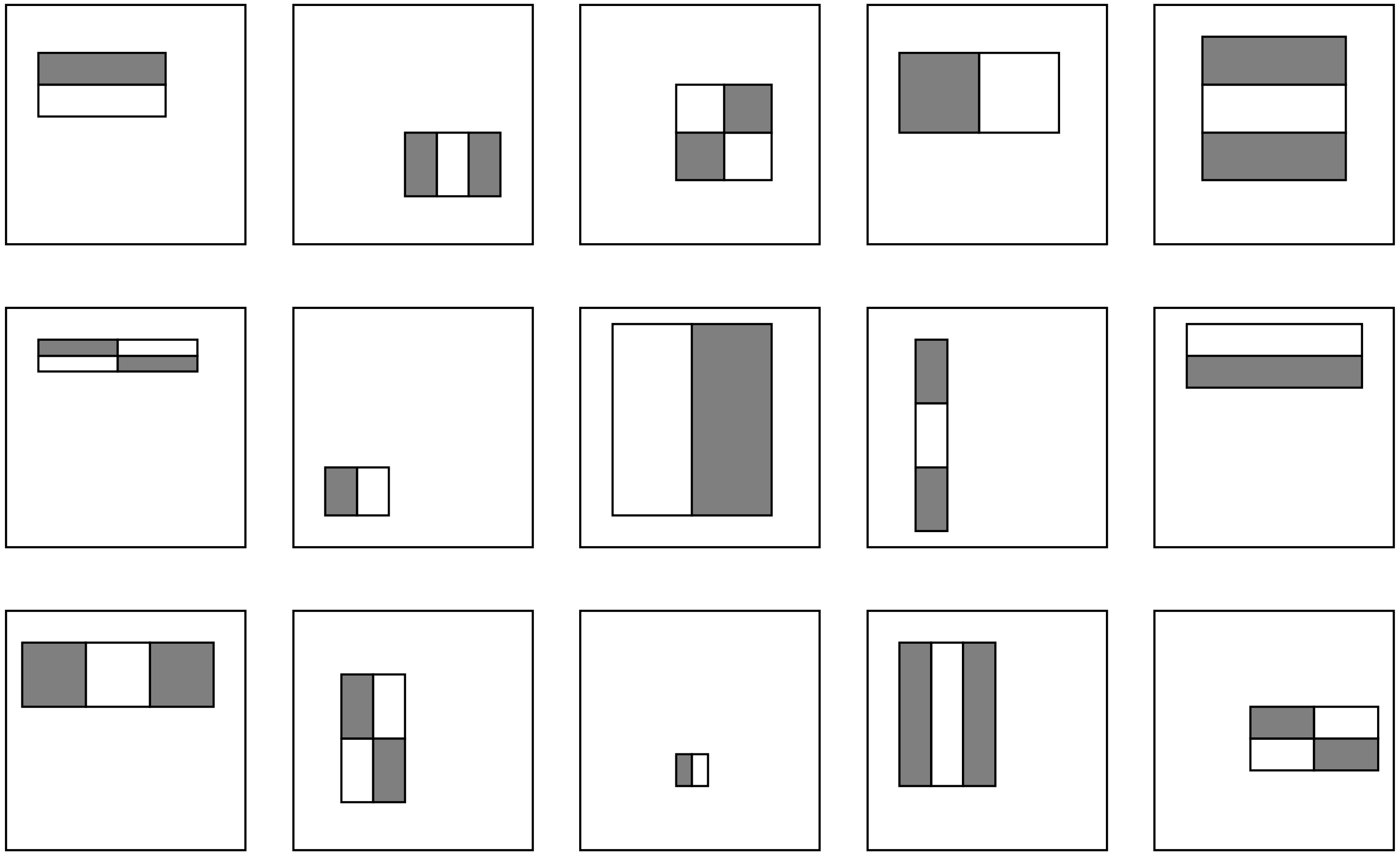
Key Properties of Face Detection

- Each image contains 10 - 50 thousand locs/scales
- Faces are rare 0 - 50 per image
  - 1000 times as many non-faces as faces
- Extremely small # of false positives: 10^{-6}
- Complex operation on each window (e.g., SVM, NN) ==> ***very slow detector!***

Key Properties of Face Detection

- In practice, many ad-hoc prefilter approaches for speed (flesh color, etc)
- Viola-Jones: develop principled approach to fast detection
  - start with large library of local features
  - integral image for efficient computation
  - adaboost to find optimal combination
  - cascade architecture for fast detection

Huge "Library" of Filters



Constructing Classifiers

- Feature set is very large and rich
- Perceptron yields a sufficiently powerful classifier

$$C(x) = \theta\left(\sum_i \alpha_i h_i(x) + b\right)$$

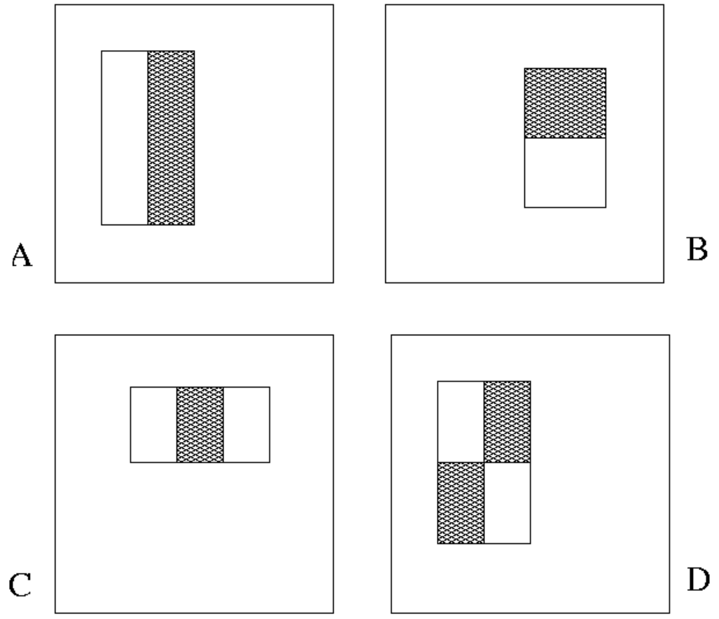
- 6,000,000 Features & 10,000 Examples
  - 60,000,000,000 feature values!
- Classical feature selection is infeasible
  - Wrapper methods
  - Exponential Gradient (Winnow - Roth, et al.)

Image Features

"Rectangle filters"

Similar to Haar wavelets

Differences between sums of pixels in adjacent rectangles



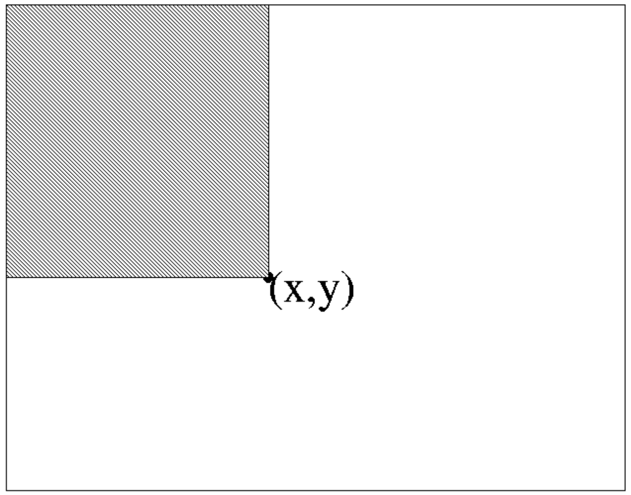
$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

$$160{,}000 \times 100 = 16{,}000{,}000$$
Unique Features

Integral Image

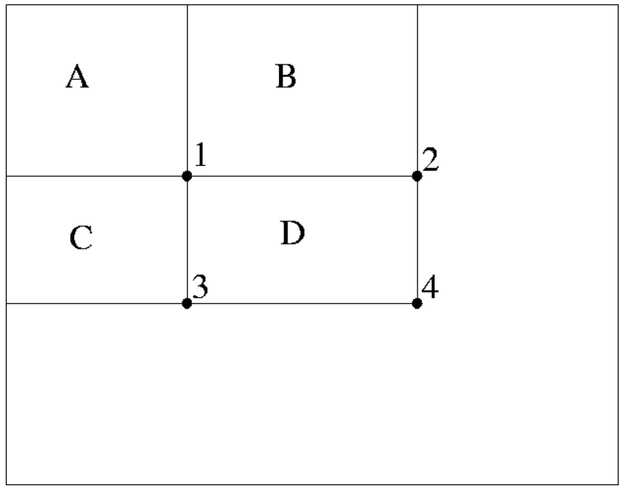
- Define the Integral Image

$$I'(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} I(x',y')$$



- Any rectangular sum can be computed in constant time:

$$\begin{aligned} D &= 1 + 4 - (2 + 3) \\ &= A + (A + B + C + D) - (A + C + A + B) \\ &= D \end{aligned}$$



- Rectangle features can be computed as differences between rectangles

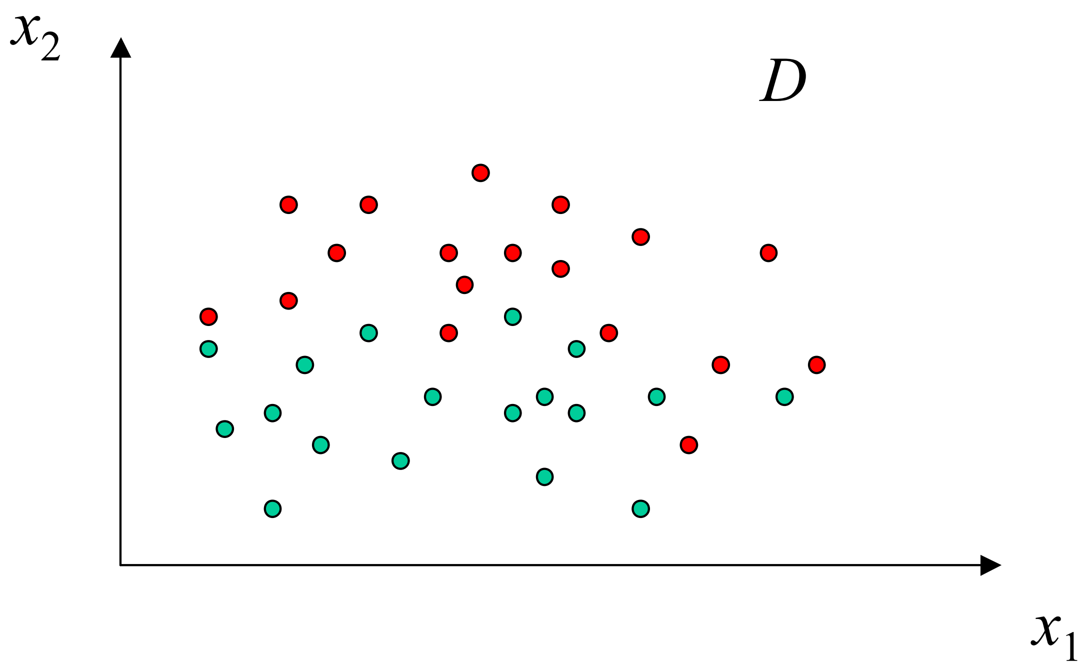
Boosting

A *weak learner* is a classifier with accuracy only slightly better than chance.

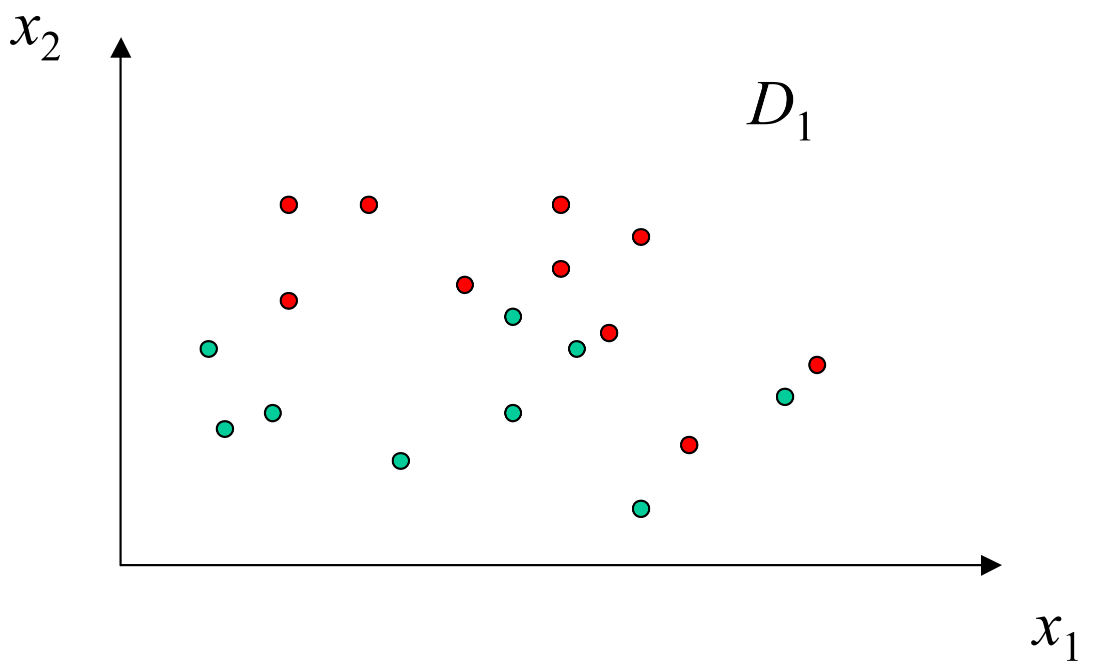
Boosting: combine a number of simple classifiers so that the ensemble is arbitrarily accurate.

Allows the use of simple (fast) classifiers without sacrificing accuracy.

Example

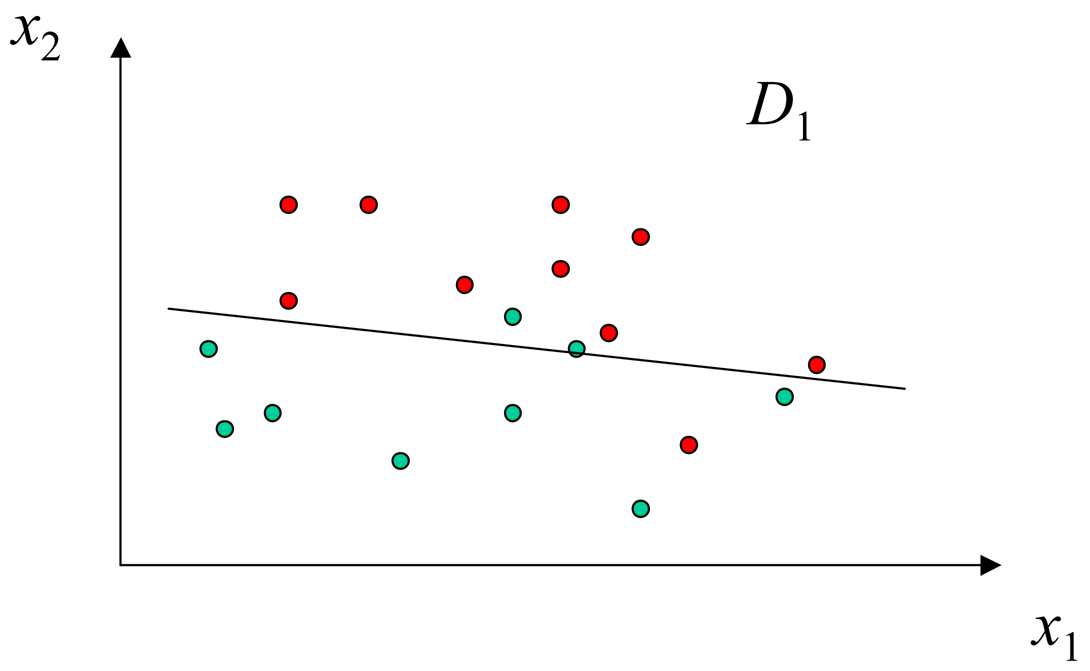


Example



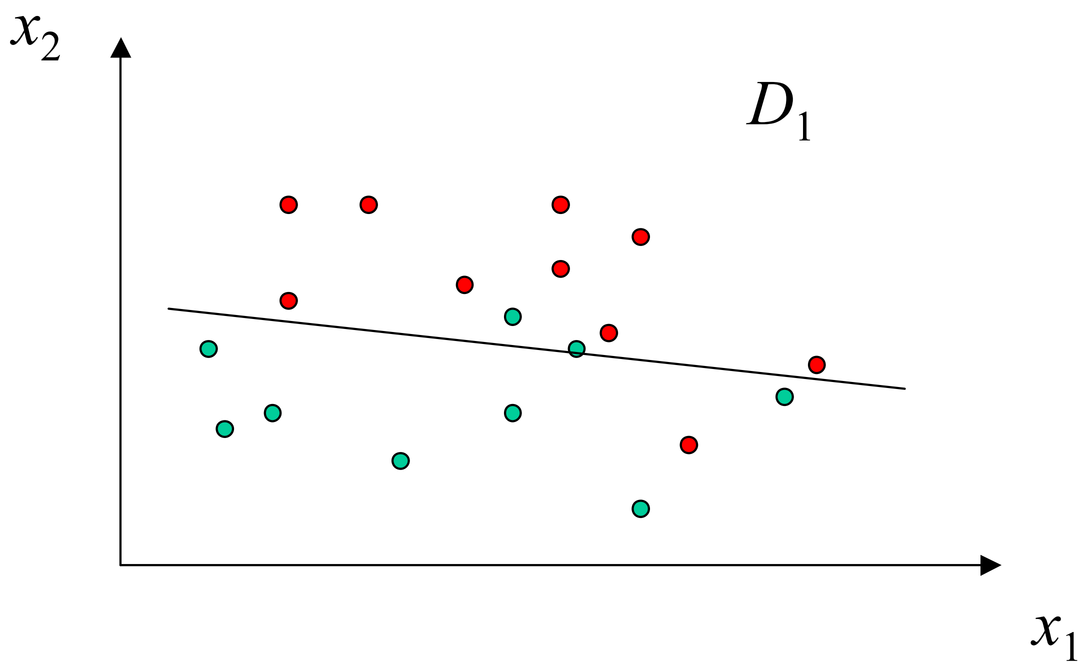
Select a subset of the data from *D* without replacement.

Example



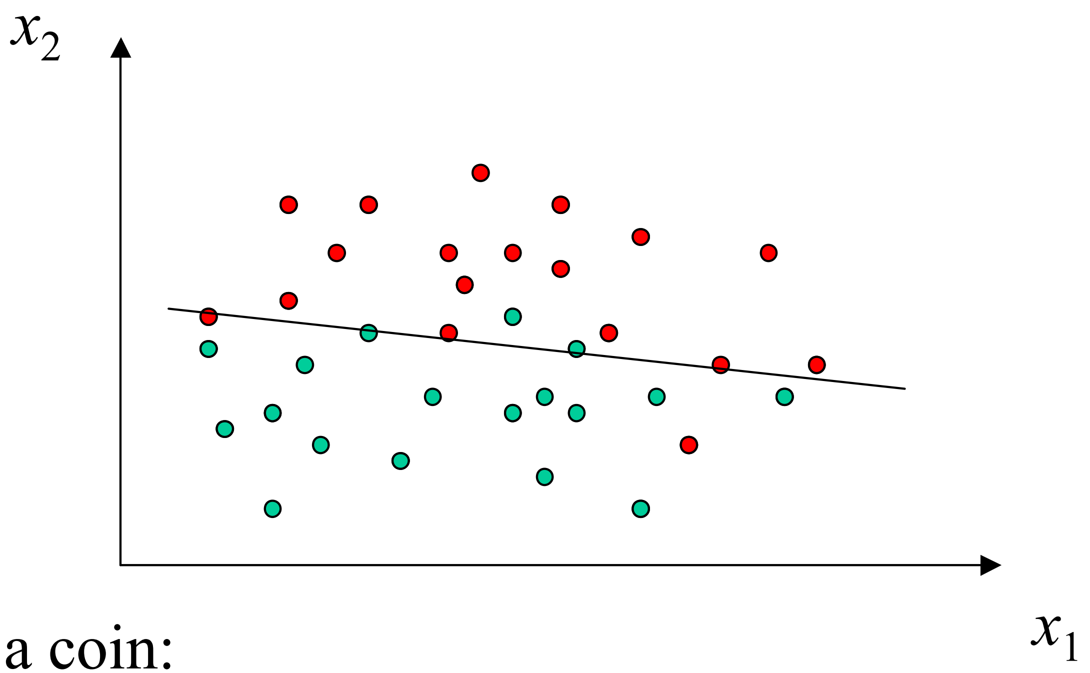
Imagine we have a simple linear classifier, C_1.
It need only perform better than chance.

Example



Chose a new set D2 that is “most informative”.

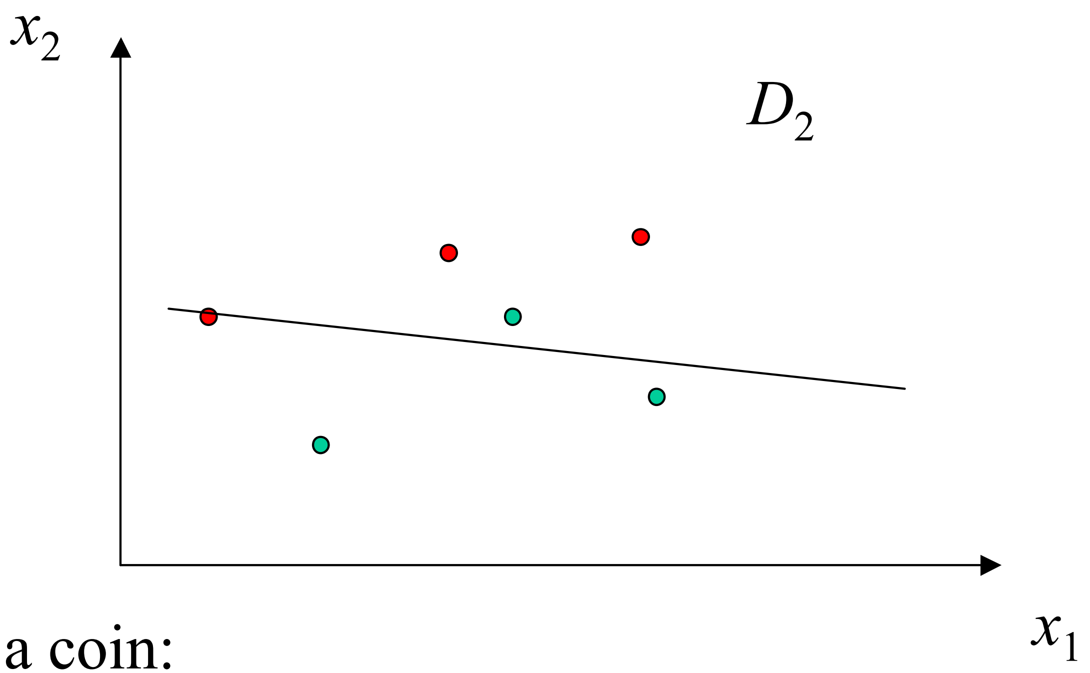
Example



Flip a coin:

Heads: sample from D and present them to C_1 until it fails, then add that sample to $D_{2.}$

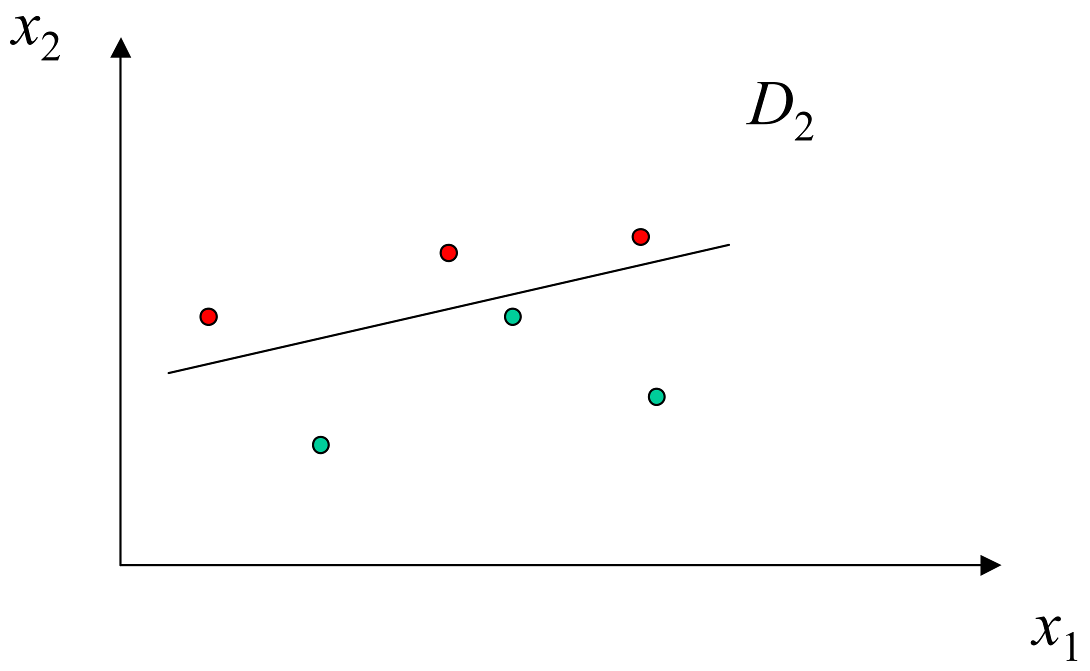
Example



Flip a coin:

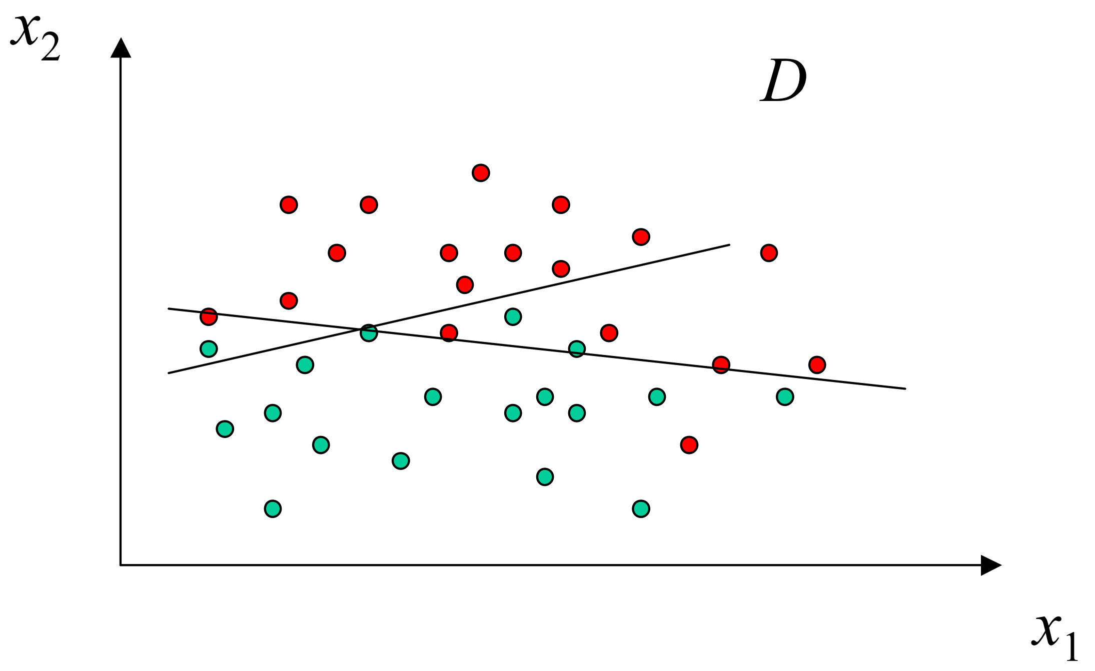
Tails: sample from D and present them to C_1 until it classifies correctly, then add that sample to $D_{2.}$

Example



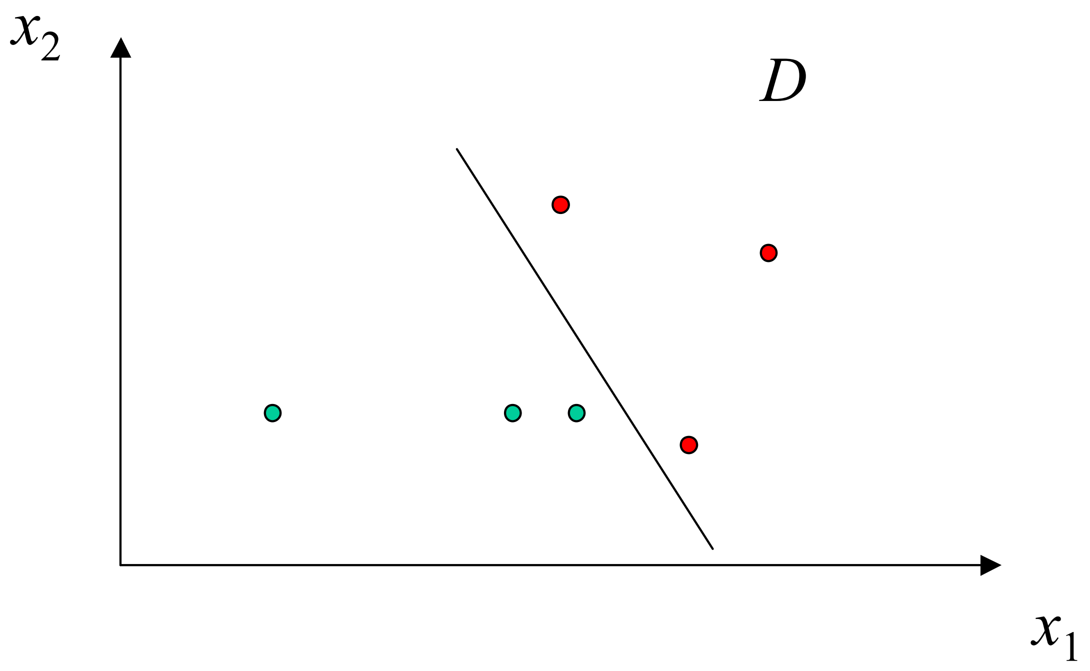
Train a new classifier C_2.

Example



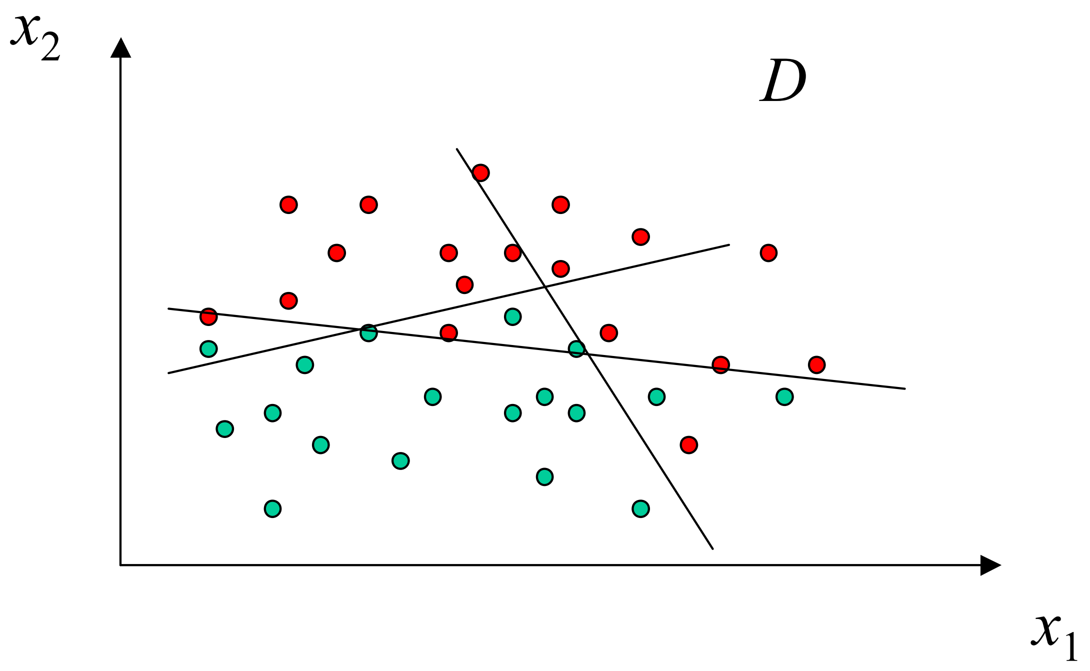
Repeat this now using both classifiers in a cascade.
Train a new classifier on data where the other two disagree.

Example



Repeat this now using both classifiers in a cascade.
Train a new classifier on data where the other two disagree.

Example



Repeat this now using both classifiers in a cascade.

Adaboost

- Same basic idea but give each data element a weight that determines its probability of being selected.
- If the element is accurately classified then it has a low probability of being selected again.
- Focuses resources on the difficult data.
- Classification based on the weighted sum of the output of the component classifiers. Weight of each classifier is related to its training error.

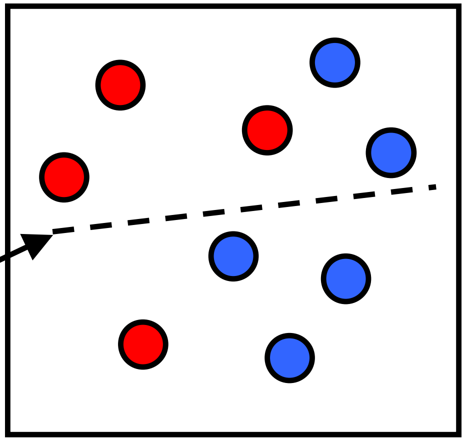
AdaBoost

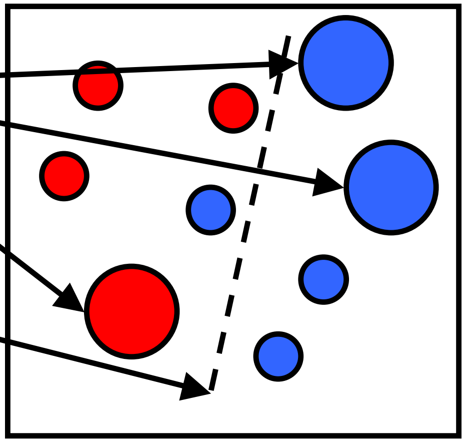
(Freund & Shapire '95)

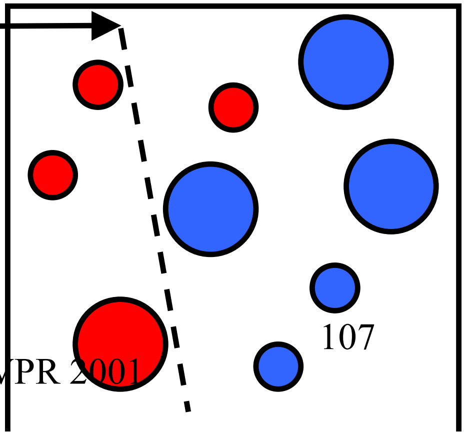
$$f(x) = \theta\left(\sum_t \alpha_t h_t(x)\right)$$

$$\alpha_t = 0.5\log\left(\frac{error_t}{1-error_t}\right)$$

$$w_t^i = \frac{w_{t-1}^i e^{-y_i \alpha_t h_t(x_i)}}{\sum_i w_{t-1}^i e^{-y_i \alpha_t h_t(x_i)}}$$







Final classifier is weighted combination of weak classifiers

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

AdaBoost for Efficient Feature Selection

- Image Features = Weak Classifiers
- 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)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples
  - (There are many tricks to make this more efficient.)

Building a Classifier from Features

Use a single rectangle feature as weak learner

A weak learner consists of a feature f_t, a threshold θ_t, and a parity $p_t = \{-1,1\}$:

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) < p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

Picking a weak learner amounts to finding the rectangle feature with lowest weighted error

Final Classifier is a Perceptron

The classifier learned by AdaBoost is a perceptron:

$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) > 0.5 \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) < p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

Each feature f(x) can be represented as a list of coordinates and a weight: $(x_1, y_1, w_1), (x_2, y_2, w_2), \ldots$

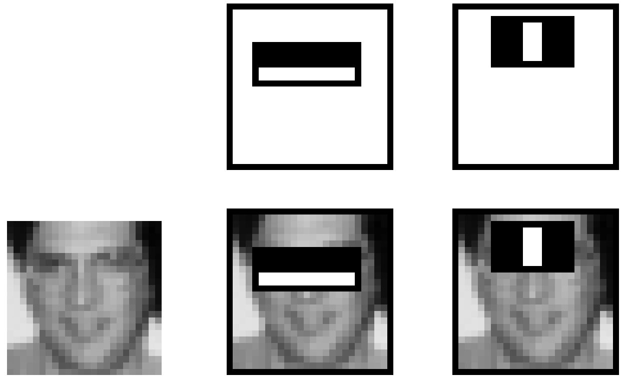
To apply the classifier to larger image sub-windows, we simply scale up each feature.

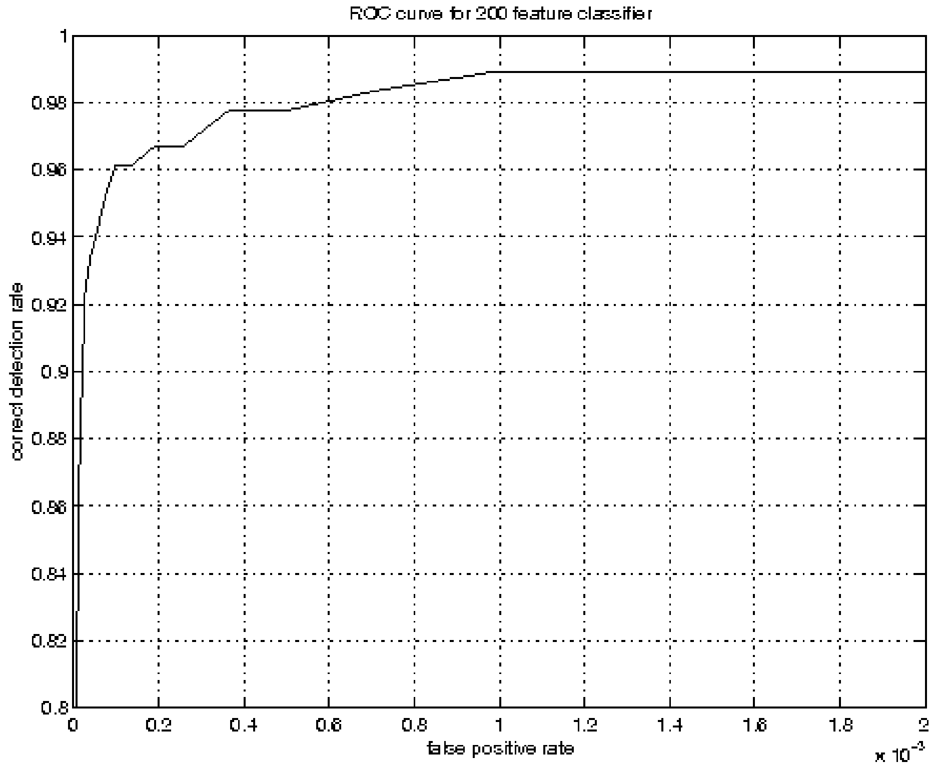
Example Classifier for Face Detection

A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084
false positives.

Not quite competitive...



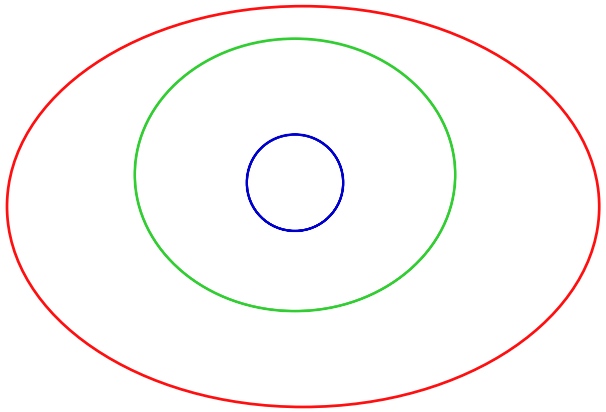


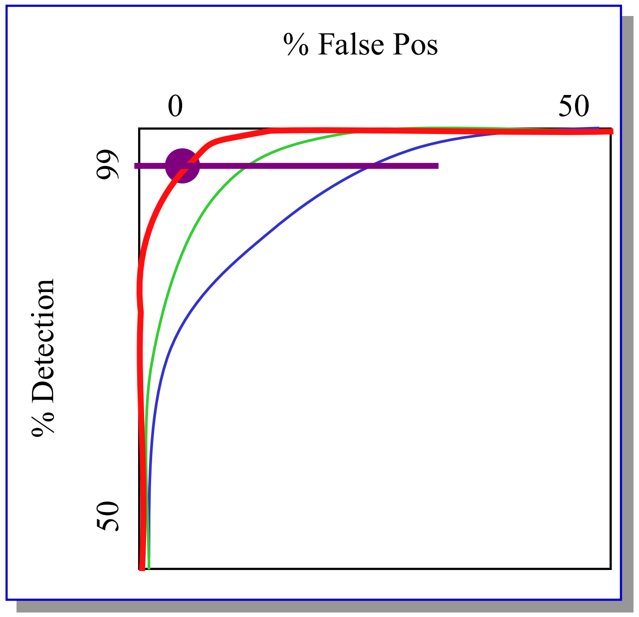
ROC curve for 200 feature classifier

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

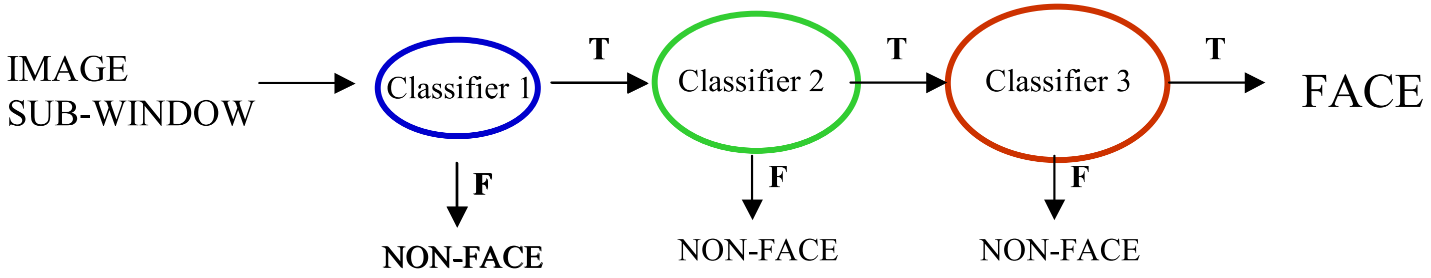
Trading Speed for Accuracy

- Given a nested set of classifier hypothesis classes

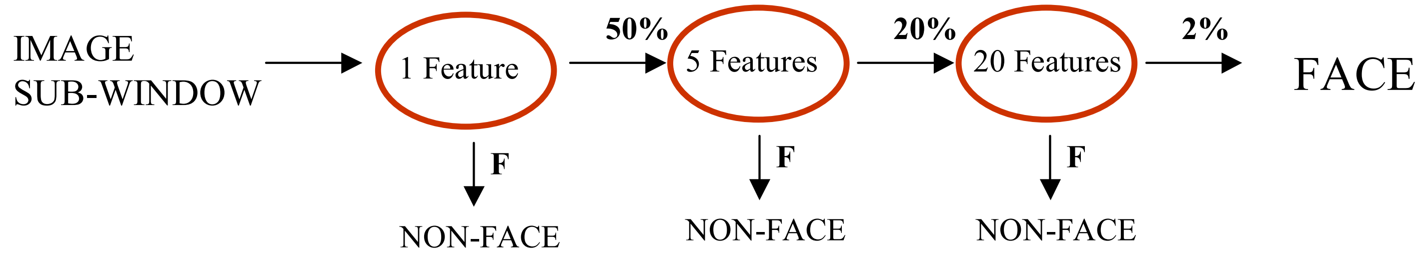




- Computational Risk Minimization



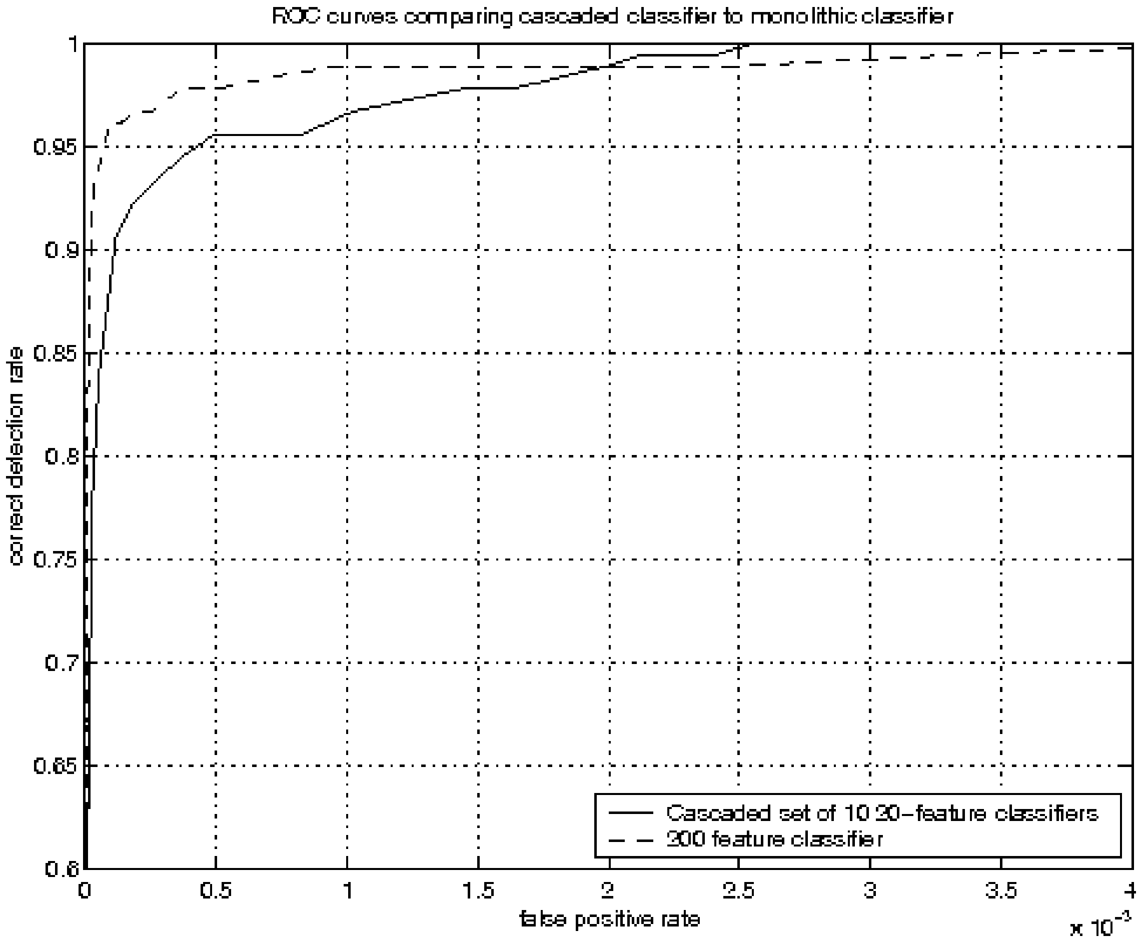
Cascaded Classifier



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

Experiment: Simple Cascaded Classifier



A Real-time Face Detection System

Training faces: 4916 face images (24 x 24 pixels) plus vertical flips for a total of 9832 faces

Training non-faces: 350 million sub-windows from 9500 non-face images

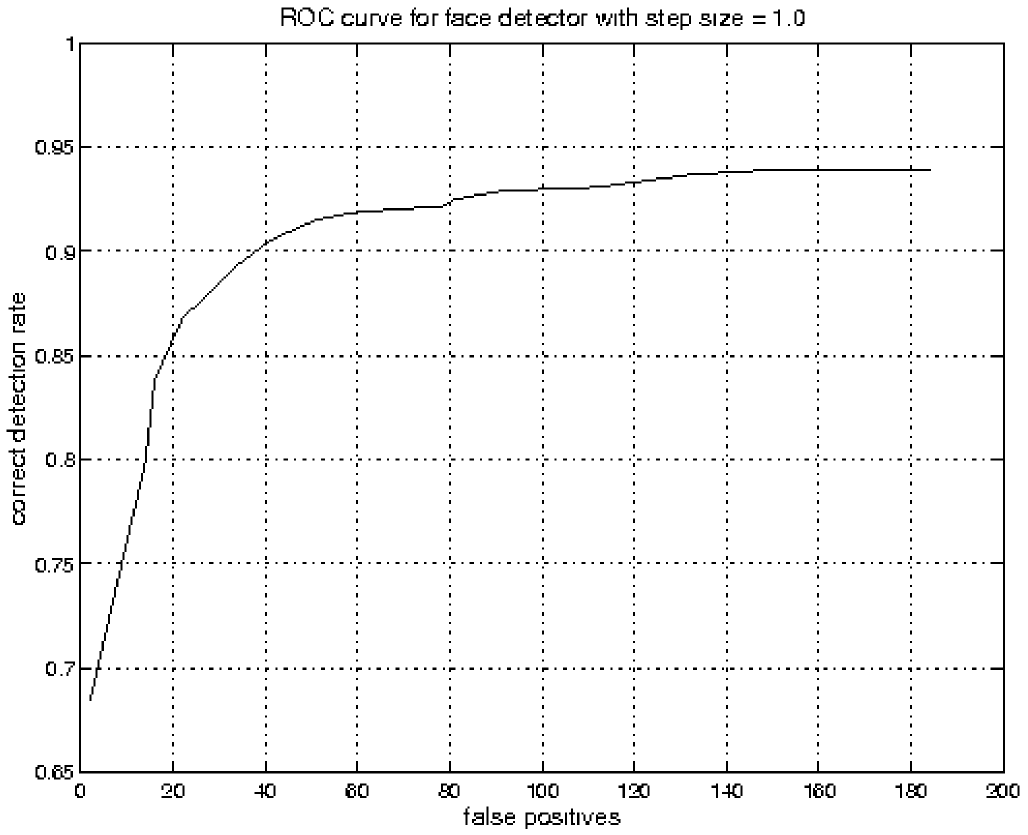
Final detector: 38 layer cascaded classifier
The number of features per layer was 1, 10, 25, 25, 50, 50, 50, 75, 100, …, 200, …

Final classifier contains 6061 features.

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

Accuracy of Face Detector

Performance on MIT+CMU test set containing 130 images with 507 faces and about 75 million sub-windows.



Comparison to Other Systems

Detector \ False Detections	10	31	50	65	78	95	110	167
Viola-Jones	76.1	88.4	91.4	92.0	92.1	92.9	93.1	93.9
Viola-Jones (voting)	81.1	89.7	92.1	93.1	93.1	93.2	93.7	93.7
Rowley-Baluja-Kanade	83.2	86.0				89.2		90.1
Schneiderman-Kanade				94.4				

Speed of Face Detector

Speed is proportional to the average number of features computed per sub-window.

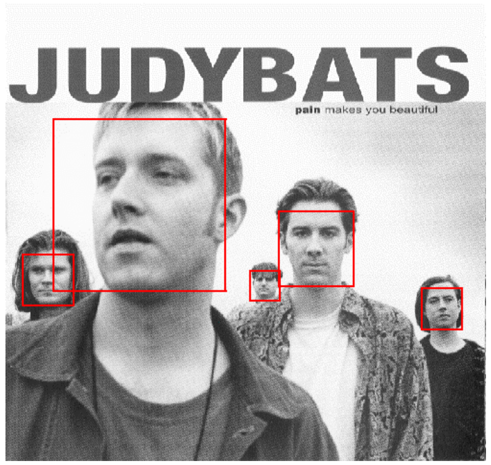
On the MIT+CMU test set, an average of 9 features out of a total of 6061 are computed per sub-window.

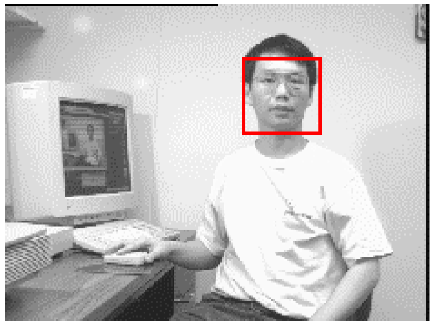
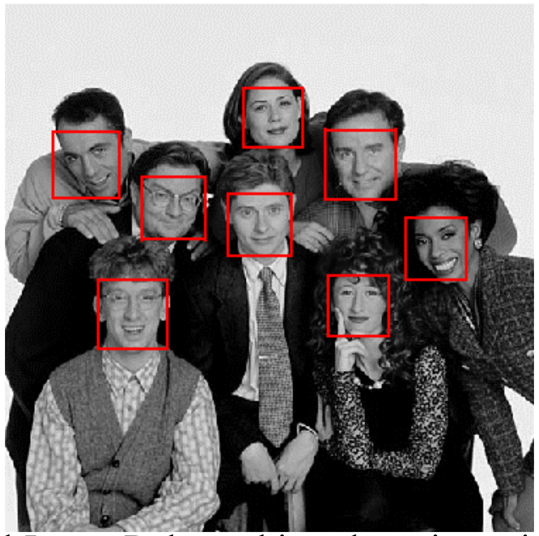
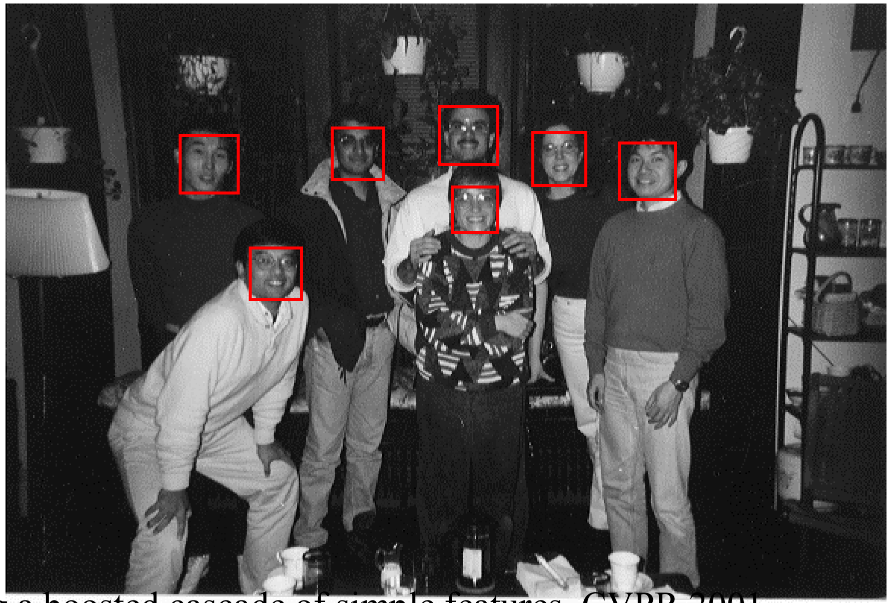
On a 700 Mhz Pentium III, a 384x288 pixel image takes about 0.067 seconds to process (15 fps).

Roughly 15 times faster than Rowley-Baluja-Kanade and 600 times faster than Schneiderman-Kanade.

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

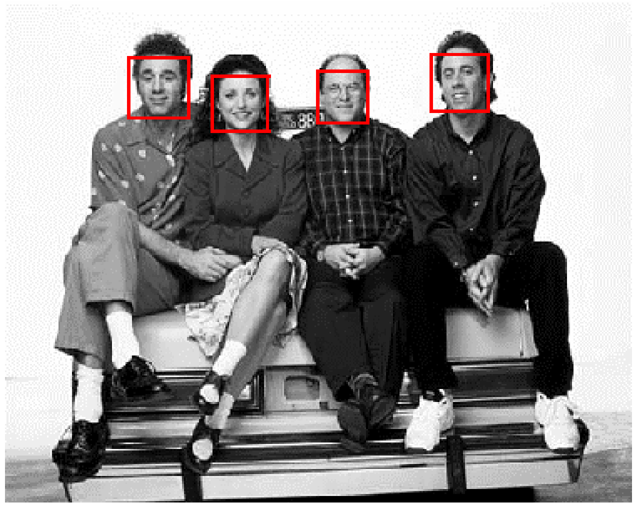
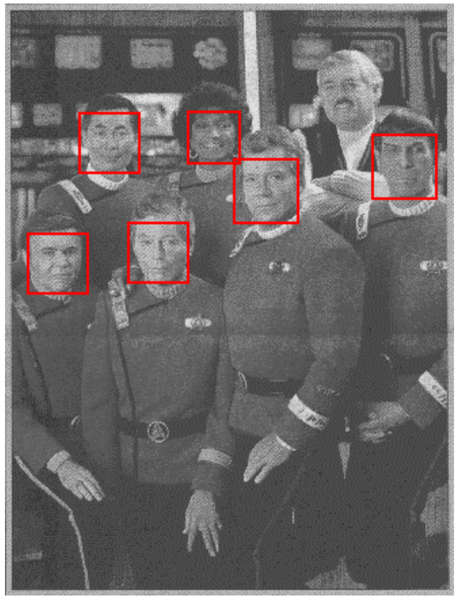
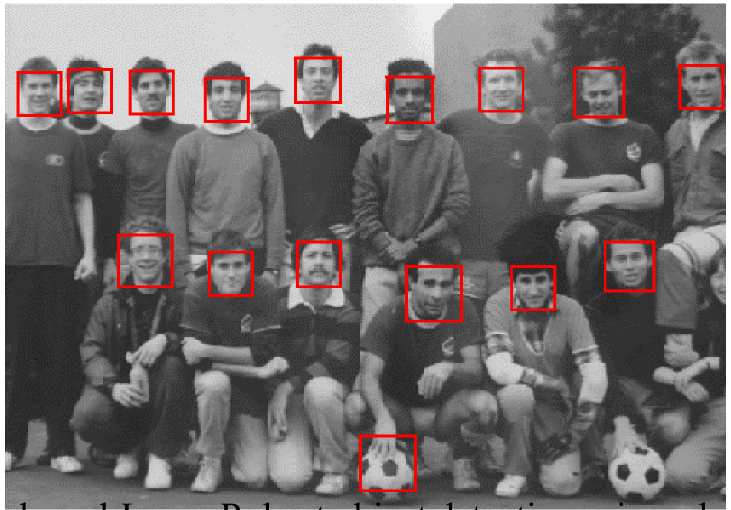
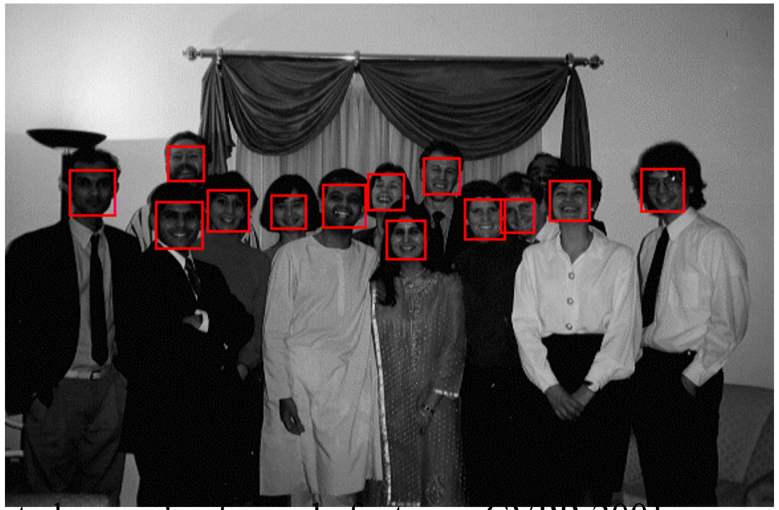
Output of Face Detector on Test Images



Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

More Examples

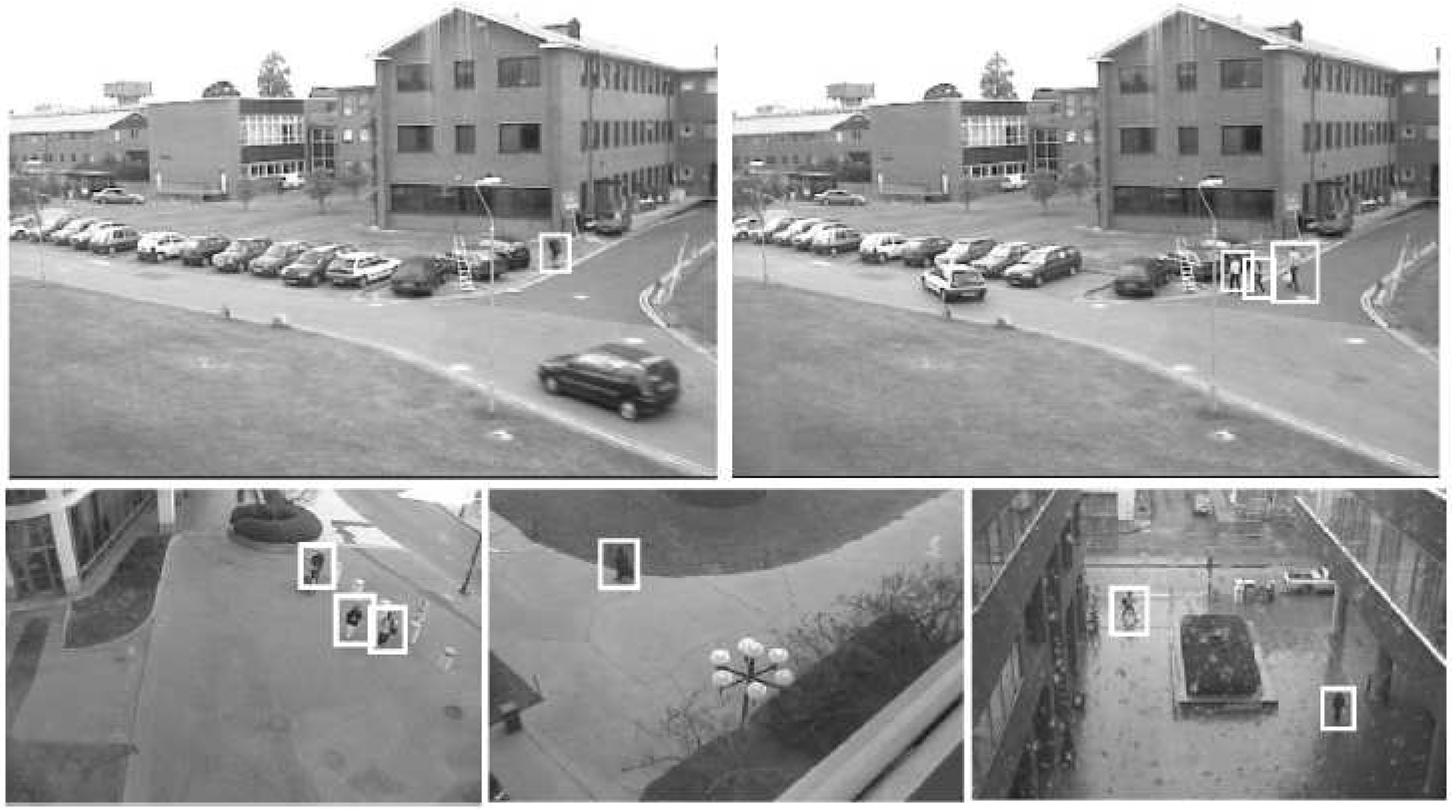





Viola/Jones

Three contributions with broad applicability

- Cascaded classifier yields rapid classification
- AdaBoost as an extremely efficient feature selector
- Rectangle Features + Integral Image can be used for rapid image analysis

Goal: Detect Pedestrians.



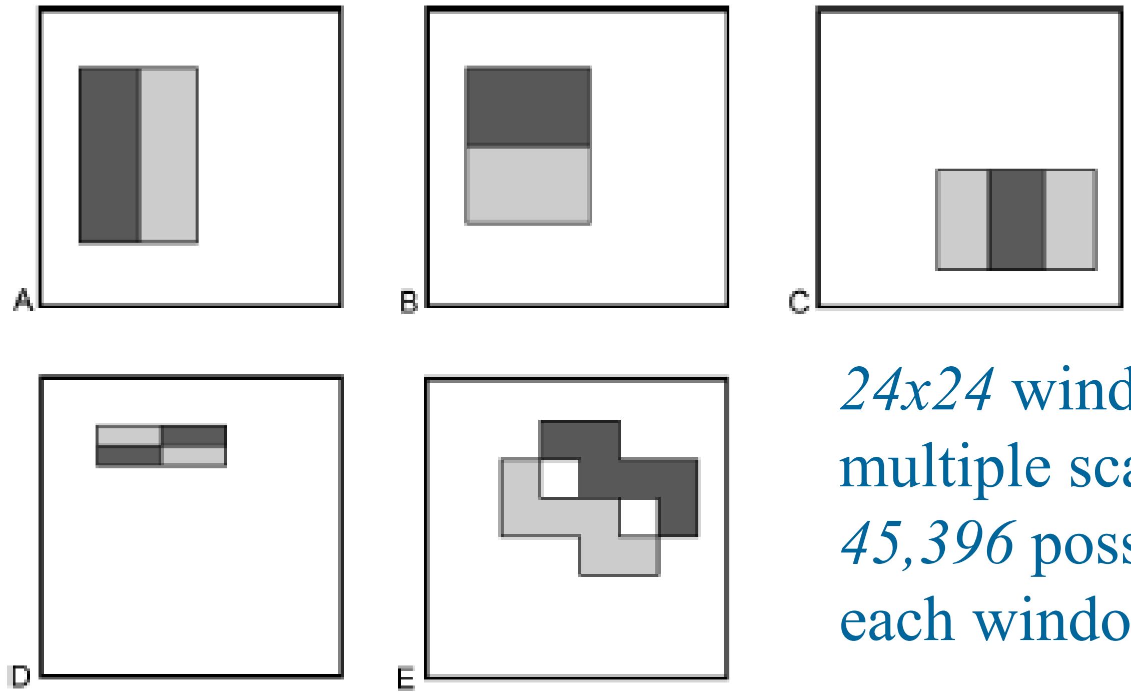
Viola, Jones and Snow, ICCV’03

Training Data

Some positive training examples.

Viola, Jones and Snow, ICCV'03

Simple Features

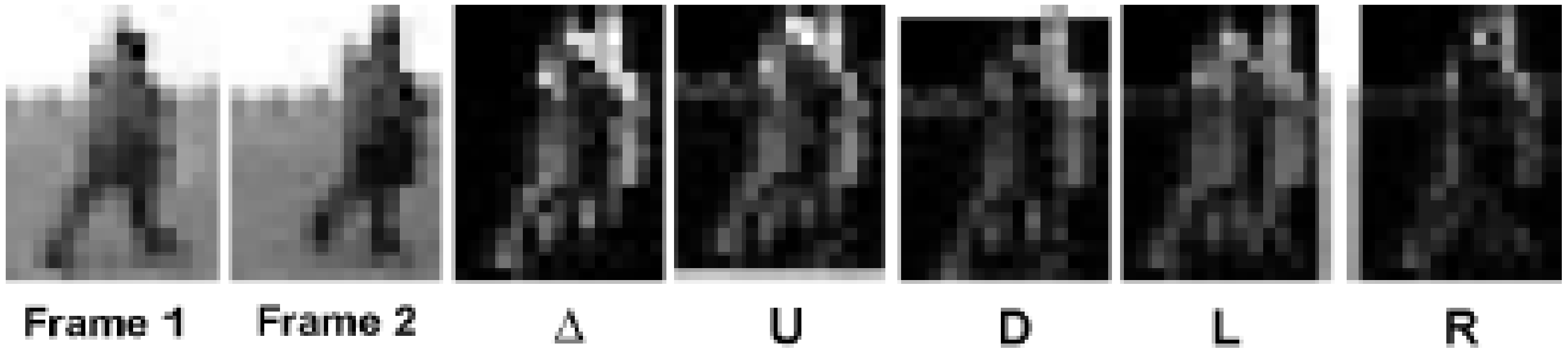


24x24 windows applied at multiple scales.
45,396 possible features in each window.

Examples of simple linear filters.
Many many different possible filters of this type.

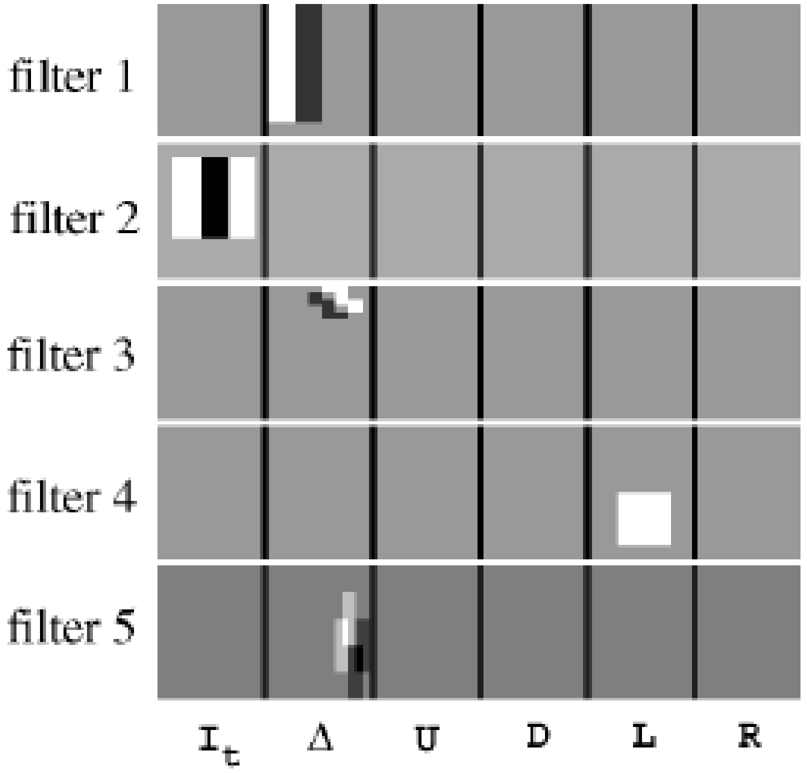
Viola, Jones and Snow, ICCV'03

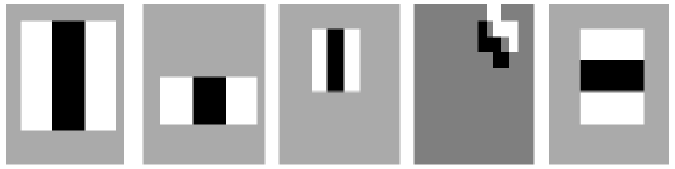
Using Motion Information



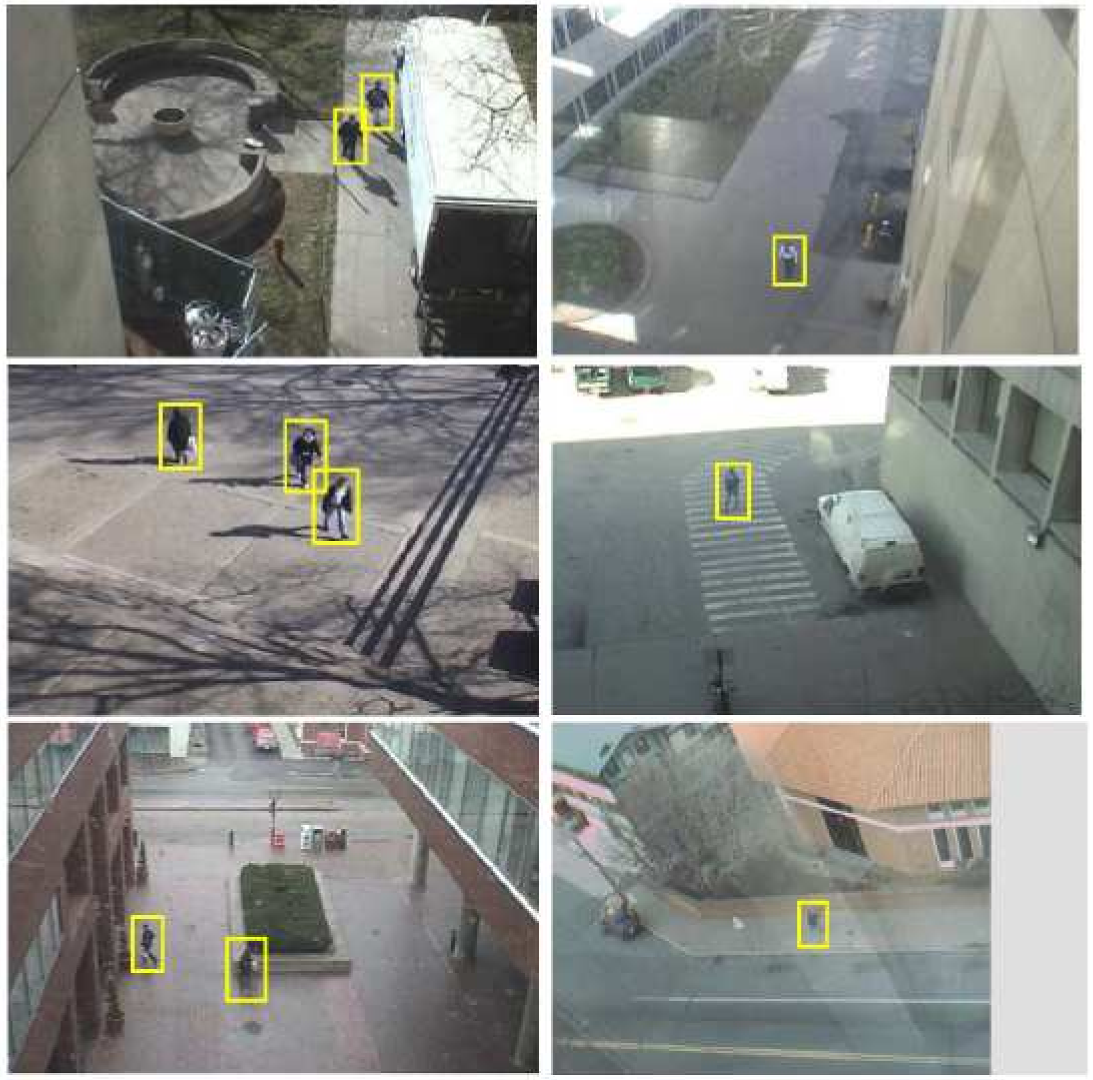
Viola, Jones and Snow, ICCV’03

Pedestrian Filters

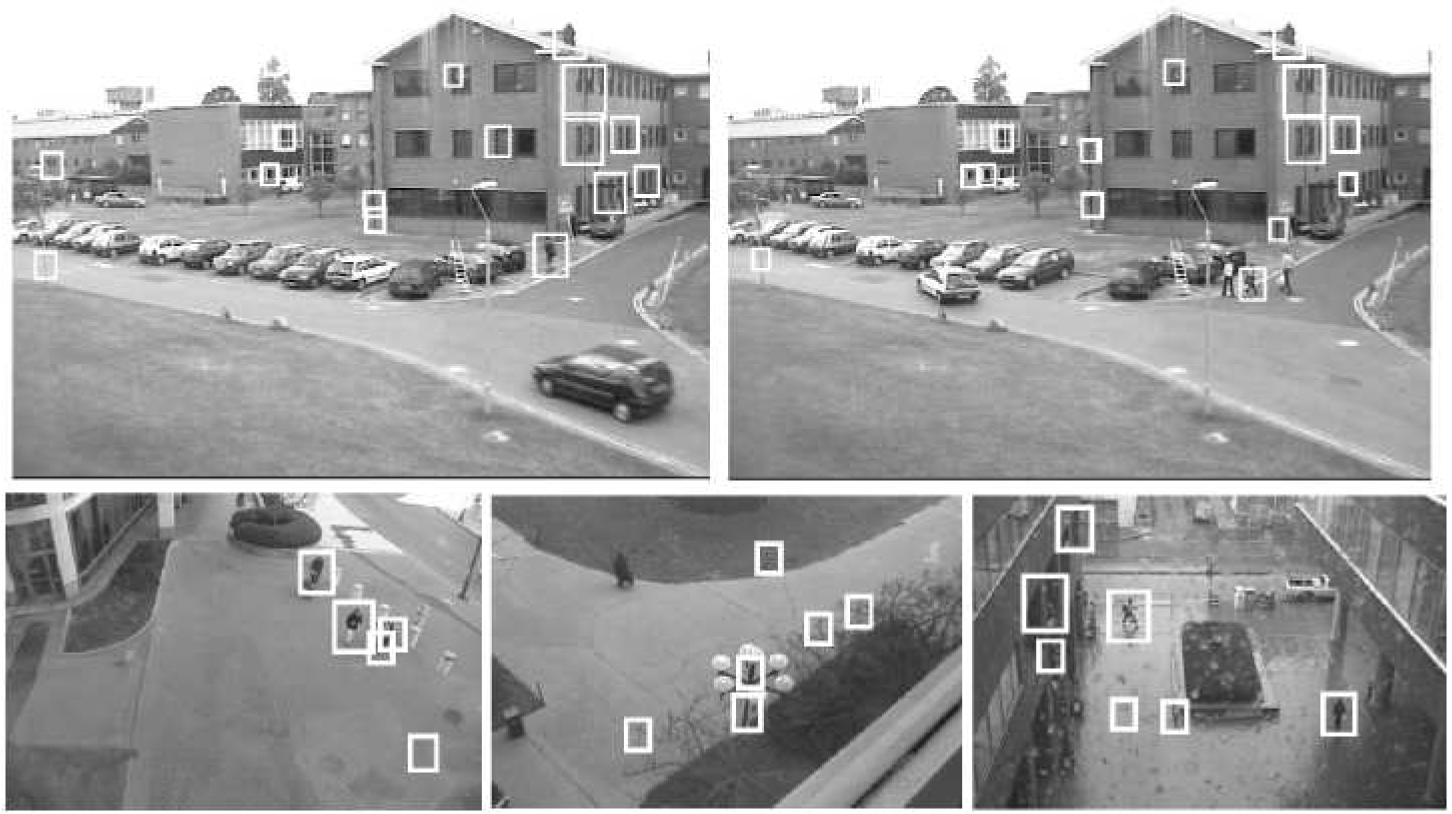




Viola, Jones and Snow, ICCV'03



Viola, Jones and Snow, ICCV'03



Viola, Jones and Snow, ICCV'03