

---

**COMP 435 Spring 2010**

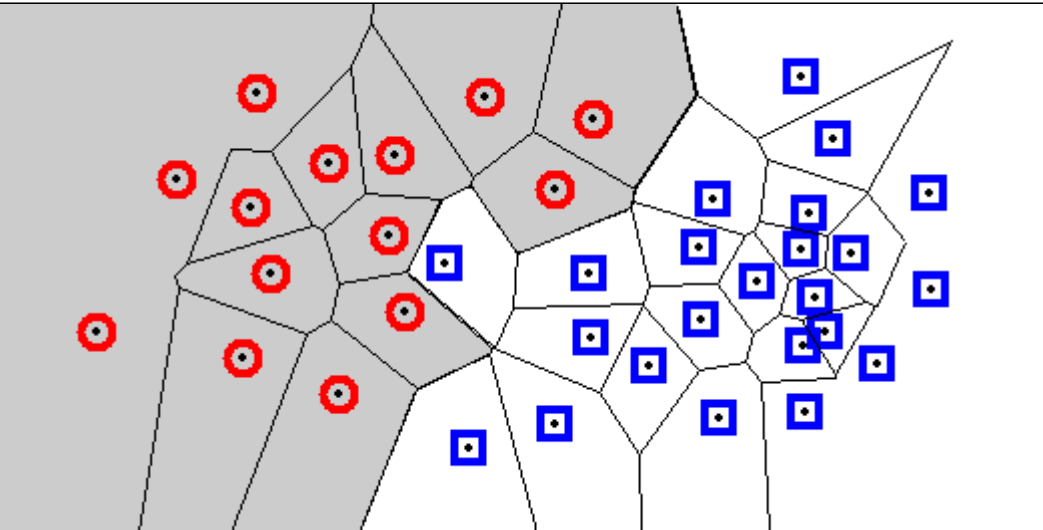
**Lecture 07**

# **Face Detection & Recognition I**

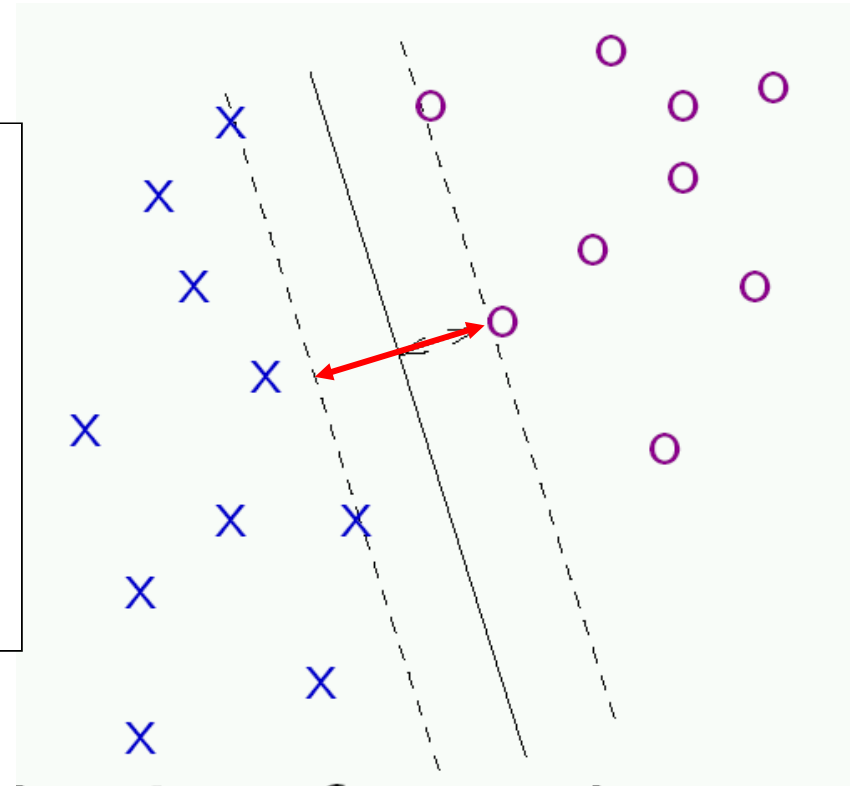
# Re-CAP of Machine Learning

- **3 types of classifiers so far:**
  - **kNN:**
    - » Not assuming any underlying prob distribution
    - » Classify by the nearest neighbour's labels
    - » Simplify computation/storage by editing/condensing/indexing
  - **SVM:**
    - » Minimize the structural risk of the classifier by maximizing the gap
    - » Numeric solution via Lagrangian decomposition and dual problem solution
    - » Kernel trick to fit non-linear boundaries (a bit tricky !)
    - » Works well in a wide variety of problems

# Illustrations



*$k$ NN Decision Boundaries*



*SVM Decision Boundaries*

# Re-CAP of Machine Learning

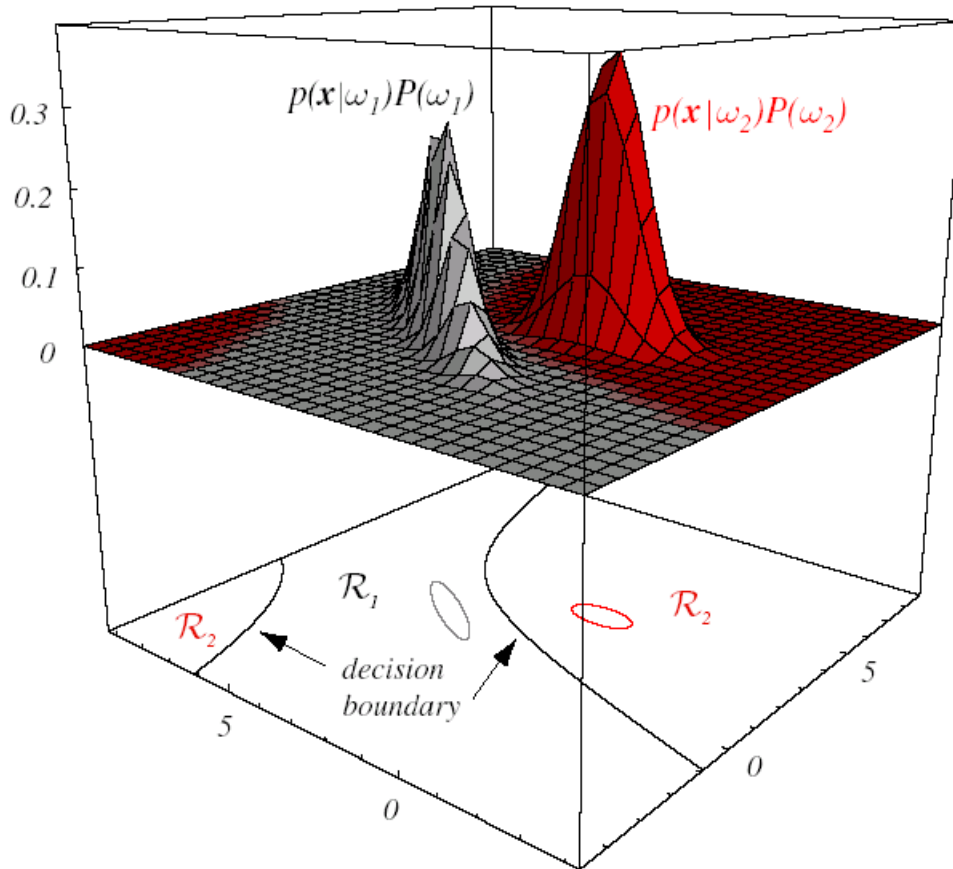
- Bayesian decision fitting a Gaussian distribution
  - Bayesian decision: given the observation, what is the most likely class label (posterior) ?
  - Likelihood: the feature distribution of a given class
  - Prior: the class prob

posterior  $\propto$  likelihood  $\times$  prior

$$p(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{\sum_i p(\mathbf{x}|\omega_i)p(\omega_i)}$$

*const* 

# Illustration of Bayesian Decision Boundaries with Gaussian Distribution Data



---

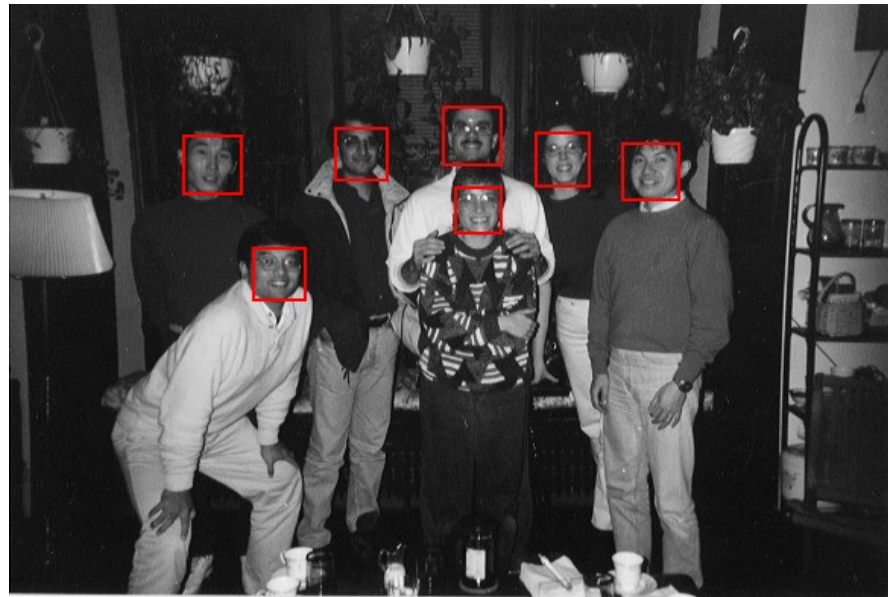
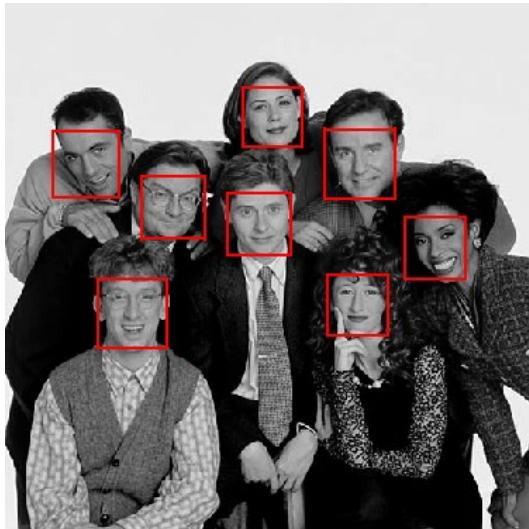
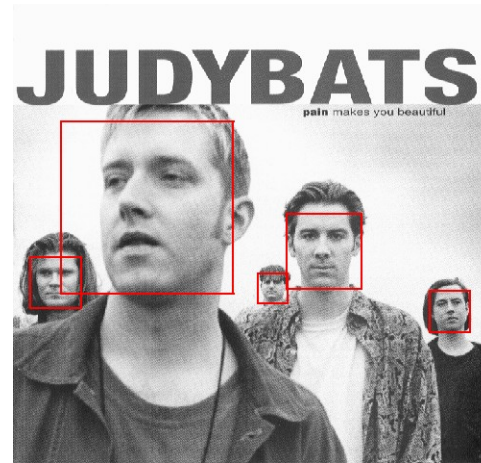
# Face Detection

# Outline

---

- **3 approaches**
  - Face Detection in Color Image: Skin Color based face detection
  - Gray scale Appearance based Face detection:
    - » Boosting based Face Detection (Viola & Jones)
    - » SVM based Face Detection (Stan Z. Li, [ioa.ac.cn](http://ioa.ac.cn))

# Face Detection in Images



# Skin Color Based Face Detection



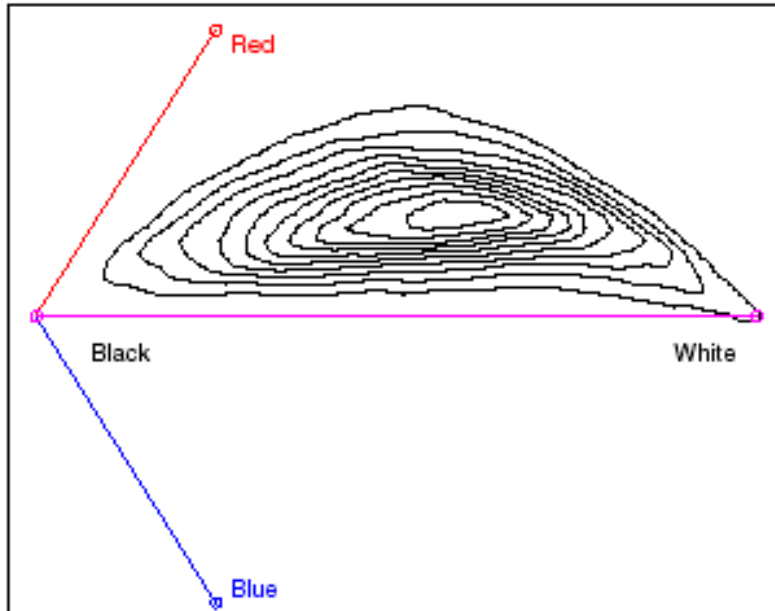
- Human skin color are quite unique in the color space
- Can be used to detect human face area from color images by labelling pixels falling inside human skin color region.
- However, can have false detection as well as shown on the left.
- Need to be further refined by the gray scale "appearance" based face detection method.

# Results from - Rehg & Jones's 99 work

- Used 18,696 images to build a general color model.
- Density is concentrated around the gray line and is more sharply peaked at white than black.
- Most colors fall on or near the gray line.
- Black and white are by far the most frequent colors, with white occurring slightly more frequently.
- There is a marked skew in the distribution toward the red corner of the color cube.
- 77% of the possible 24 bit RGB colors are never encountered (i.e. the histogram is mostly empty).
- 52% of web images have people in them.

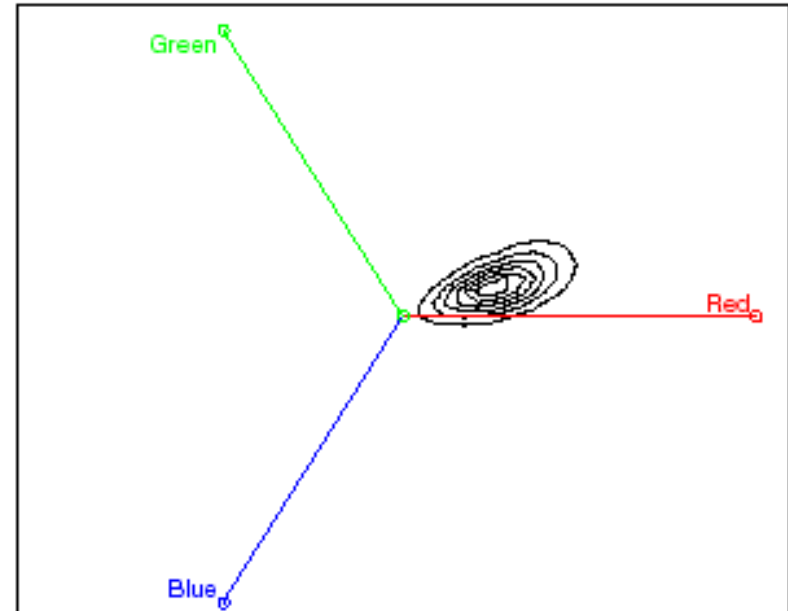
# Skin model in RGB space

Skin Color Model, Green–Magenta Axis Marginal



(a) Contour plot for skin model, marginalized along the green-magenta axis.

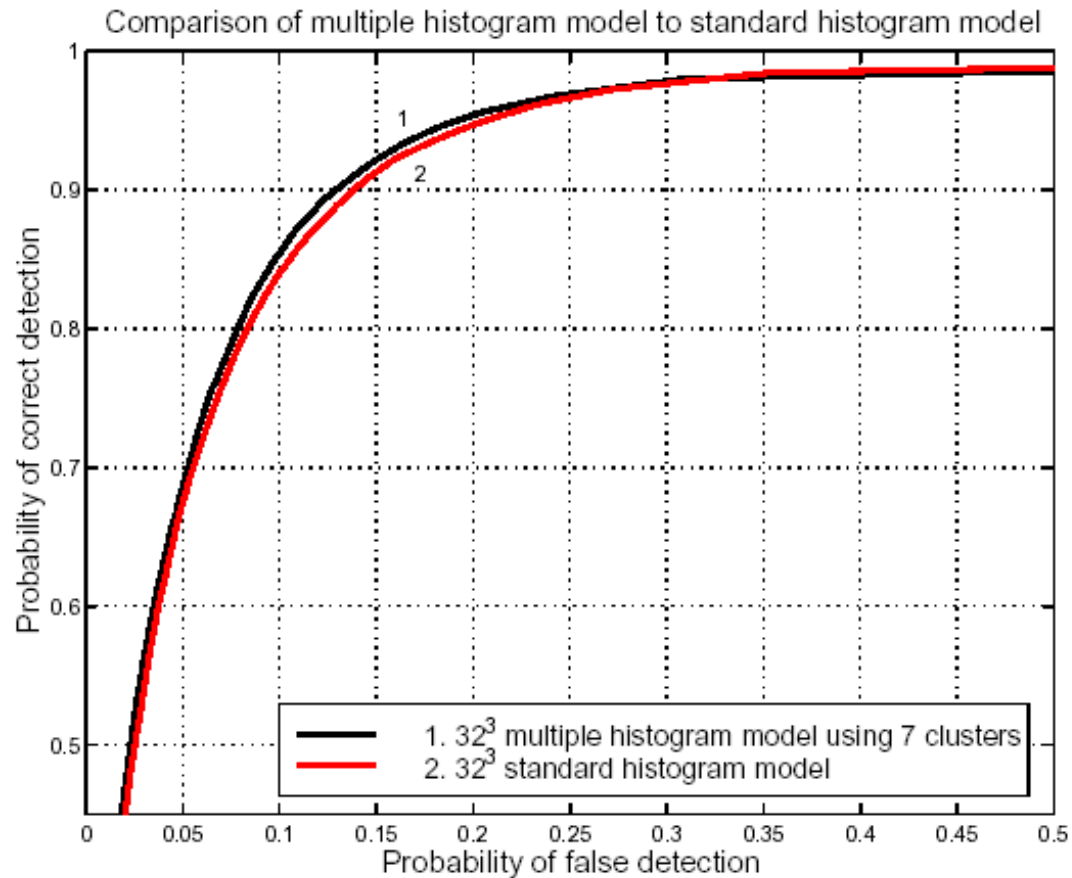
Skin Color Model, Gray Axis Marginal



(b) Contour plot for skin model, marginalized along the gray axis.

*Notice that this is a pixel based operation in detection !*

# Performance

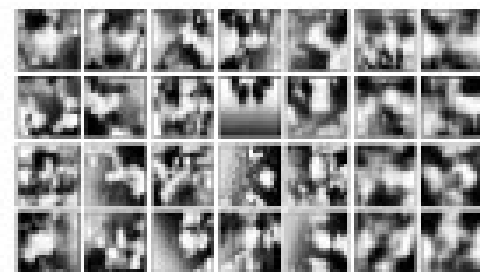


# Skin Color Based Summary

- Skin color is a surprisingly simple yet effective face area detection solution
- Very fast in operation due to pixel based simple operation
- But still not good enough, so we need appearance based detections, which we will discuss
  - SVM based face vs non-face image detection
  - Boosting based face detection

# Appearance

- Integral image feature + boosting method can be viewed as a local feature based face detection
- Face vs Non-face image appearances:
  - Face images from training set, crop and cut
    - How to generate more ? (by small shifting and in and out of plane rotations)
  - Nonface images
    - What are the important ones ? (close to decision boundary)



# The complexity of appearance

Consider an 20x20 pixel images

It could have  $256^{20 \times 20} = 2^{3200}$  possible images, for 8 bit/ 256 gray scale levels

World population is roughly 6.4 billion, or  $6,400,000,000 = 2^{32}$ .

So this small thumbnail appearance space is huge and decision boundaries between face and non-face quite complex.

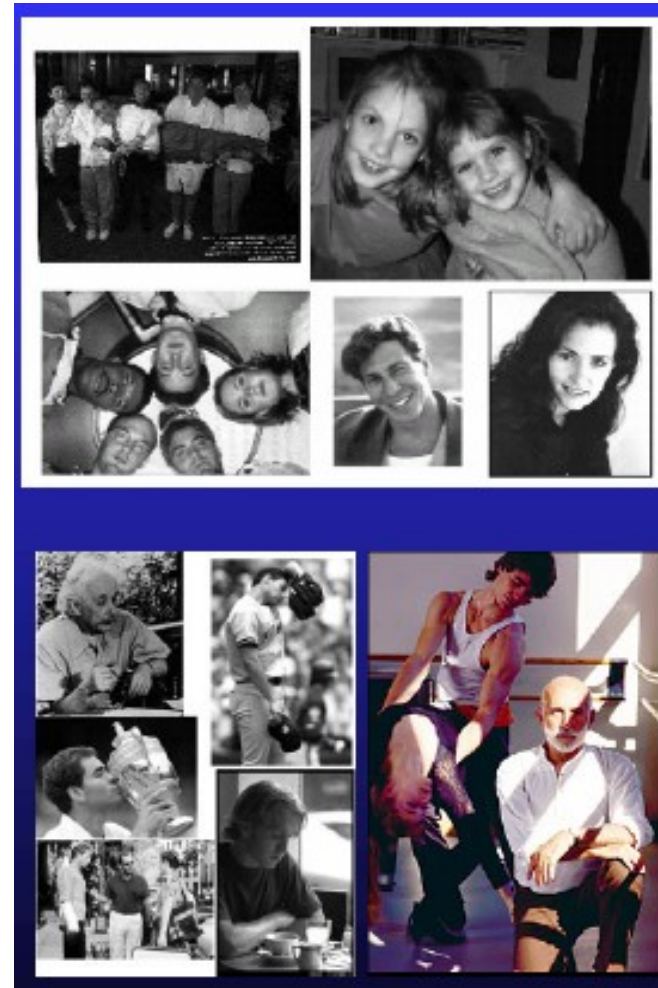
Challenges:

To capture all variations in face images so we won't miss one in detection

To have a very good characterization of the decision boundary between face and nonface, in a very hi-dimensional space ( $\mathbb{R}^{400}$  for 20x20 thumbnails)

# Why face detection is so difficult ?

- Variations in the face appearances due to:
  - Poses and Orientations,
    - Out of plane rotations of faces
    - Picture orientations
  - Illumination and shadows
    - Image formation variations
    - Lighting changes
  - Facial Expressions
    - Sad, happy, neutral, ....etc.
  - Cosmetics and glasses
    - Sun glasses, beard, ..et.c



# Face appearance features

- Pixel based
  - Each  $w \times h$  pixel images is represented as a vector in  $\mathbb{R}^{w \times h}$
  - Or a point in  $\mathbb{R}^w \times \mathbb{R}^h$  tensor space
- Block based
  - Represented as small blocks overlapping or non-overlapping
  - Pre-processed by PCA



# Generate more positive samples

Generate more samples by

Mirror and view morphing

Small random rotate, translate, scale operations

Simple and effective



[Sung & Poggio, 94]

## Find more useful negatives

---

From kNN and SVM classifier, we know that only samples on the decision boundaries are influencing the decision boundaries

How to find “good” negatives ?

Those are mis-classified as faces in the training

This is called “Bootstrapping” [Sung & Poggio 94]

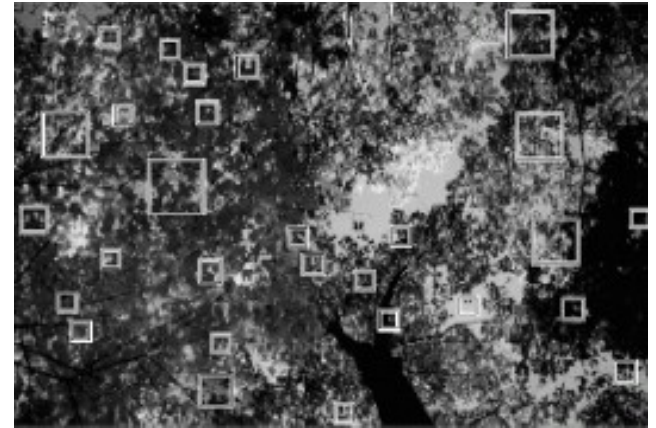
# Bootstrapping

Start with an initial set of training data, train a classifier.

In iteration  $k$ , find those nonface images wrongly classified as faces from test images from test images

Add these non-face images to the classifier training

Repeat the process until satisfactory

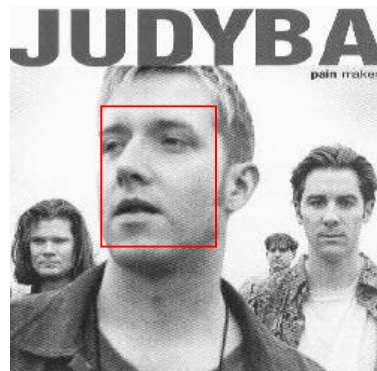


*bootstrapping*

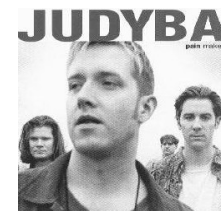


# Search strategy

- Need to find faces in different:
  - locations and Scales
- Search strategy: fix the detection window, say  $25 \times 20$ 
  - For all image scales
    - $\{1.0, 0.9, 0.9^2, 0.9^3 \dots\}$
  - For all pixel locations  $(x, y)$



...



# Face vs Non-Face in feature space

Model face/non-face appearance

as clusters of data points

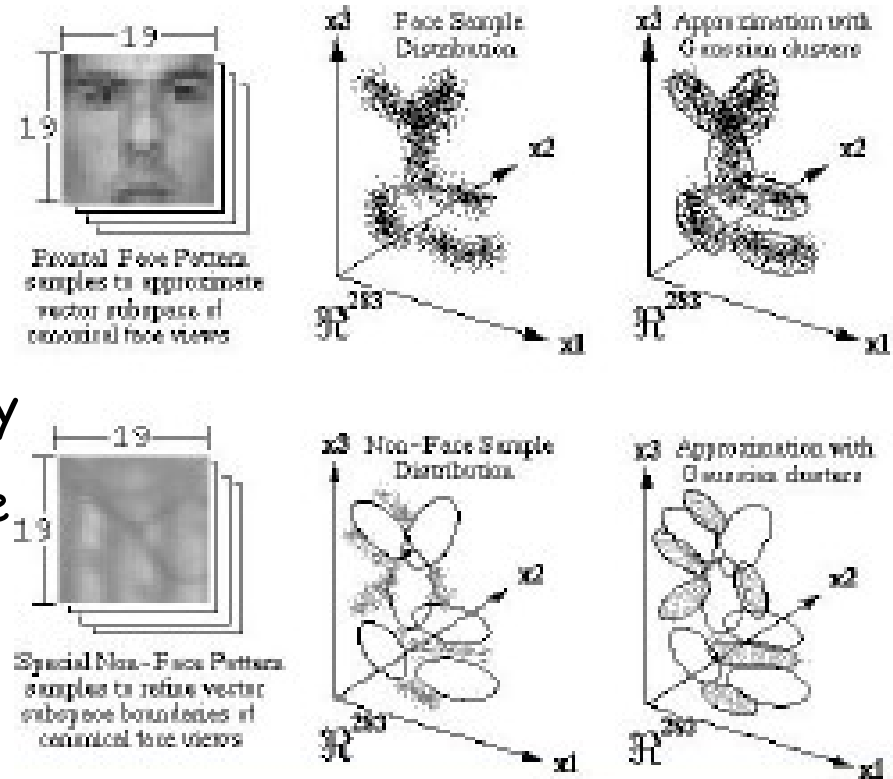
6 clusters via k-means algorithm in

[Moghaddam & Pentland'97]

Then recognition can be achieved by

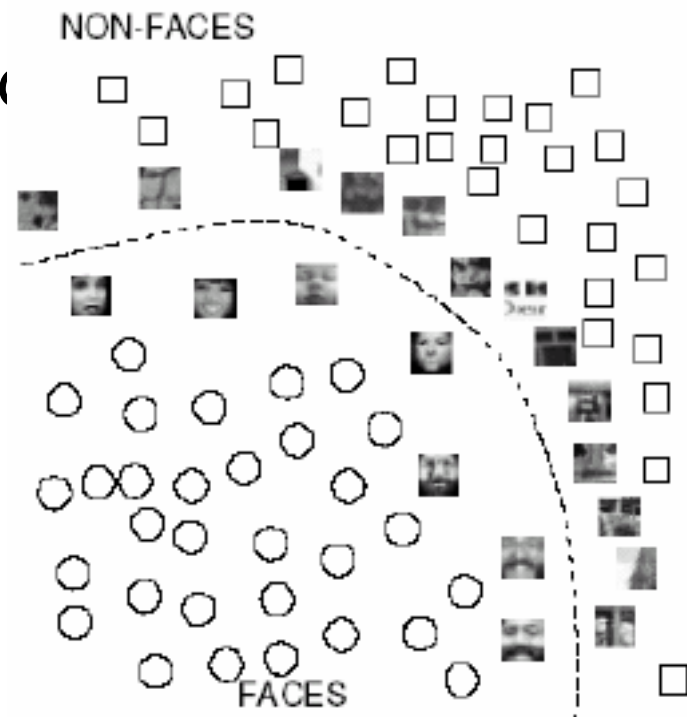
learning a boundary between face

vs non-face data points



# Classifier for Detection

- Given a "good" training set of  $\{x_k, y_k\}$ , with  $\{x_k\}$  in  $\mathbb{R}^d$ , how to find a good decision boundary between face?
  - Distribution and Bayesian approach
  - Neural networks
  - Support Vector Machine



# SVM based solution

## Refresher of SVM

Given the data set  $\{x_k, y_k\}$ ,

The SVM decision boundary is the one that maximizes the separations between face and non-face data

The non-linearity of the decision boundaries is addressed by fitting a kernel

Challenges:

Hard to find a right kernel as it is a heuristic process

Training involves quadratic programming with  $n \times n$  or  $d \times d$  Hessians, the computation can be un-reliable.

# Solutions

- **How to compute a good SVM for face detection ?**
  - Numerical solution improvement via SMO - sequential minimum optimization.
  - Reducing the number of support vectors
  - Component based SVMs
  - NN SVM : compute a NN set from the data, fit multiple SVM models

# Profiled Face Detection

- Non-frontal faces
- No magic solution, have to train models for each pose orientation [Schneiderman & Kanade, 98].
  - Train a different model for frontal, left and right pose faces



# Face Detection Summary

---

**Problem: Find and localize faces in images**

## **Challenges:**

Modeling difficulty with high dimensional face/non face features

Complex inter and intra face image variances

Difficult to characterize the decision boundaries

## **Performance evaluation:**

Accuracy: precision-recall, or FAR/FRR

Speed:

In training, not matter as much

Online detection, critical

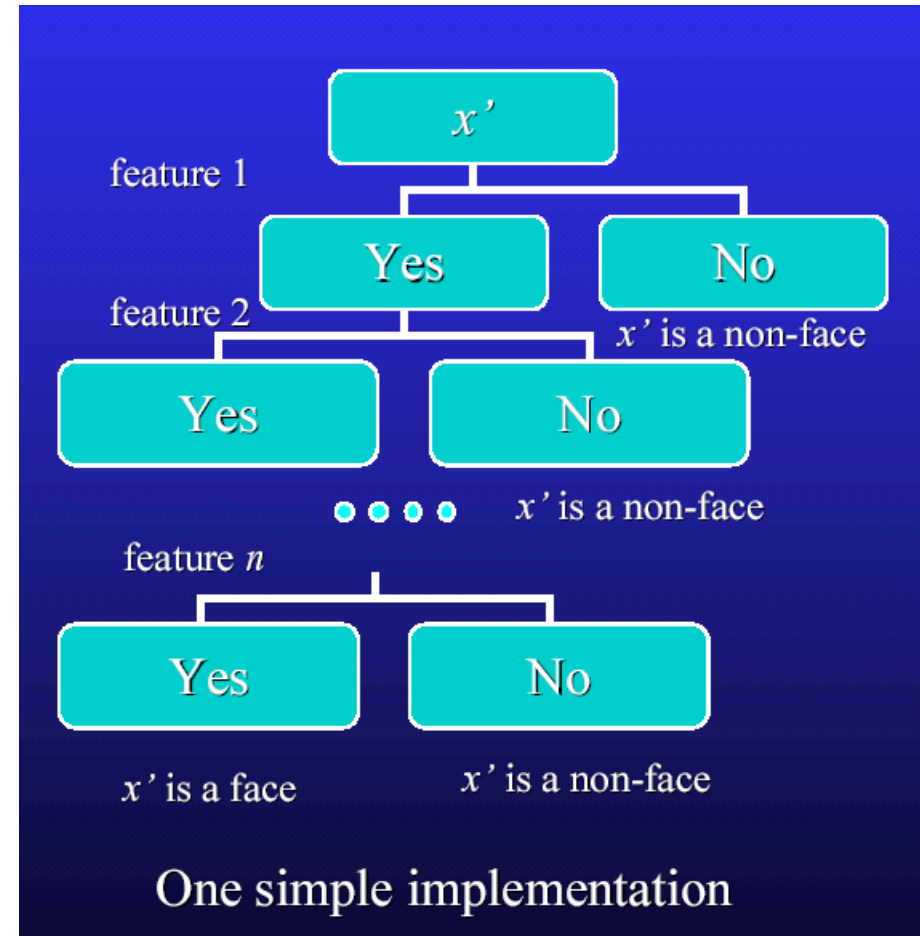
---

## Boosting Based Face Detection

- when you can't have a big horse to do the job, grown a group of smaller horses.

# Viola-Jones Face Detection Algorithm

- Overview :
  - Viola Jones technique overview
  - Features
  - Integral Images
  - Feature Extraction
  - Weak Classifiers
  - Boosting and classifier evaluation
  - Cascade of boosted classifiers
  - Example Results
- See the original CVPR paper for more detail.

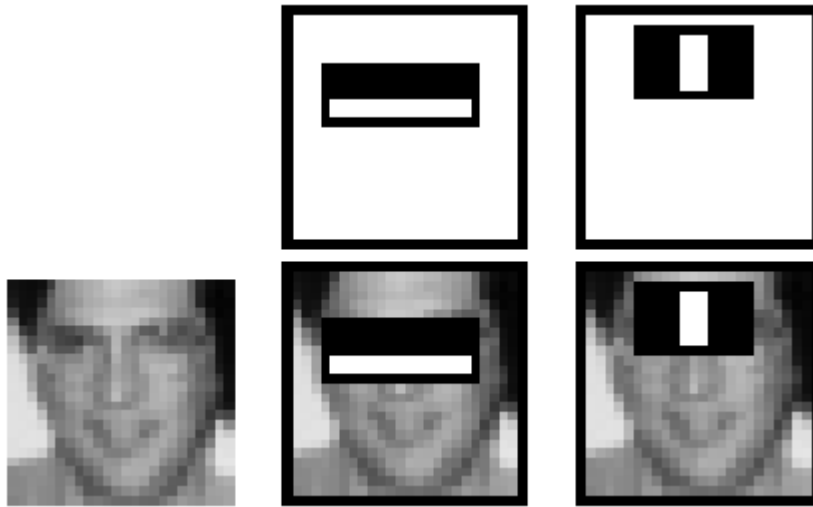


# Viola Jones Technique Overview

- **Three major contributions/phases of the algorithm :**
  - Feature extraction
  - Classification using boosting
  - Multi-scale detection algorithm
- **Feature extraction and feature evaluation.**
  - Rectangular features are used, with a new image representation their calculation is very fast.
- **Classifier training and feature selection using a slight variation of a method called AdaBoost.**
- **A combination of simple classifiers is very effective**

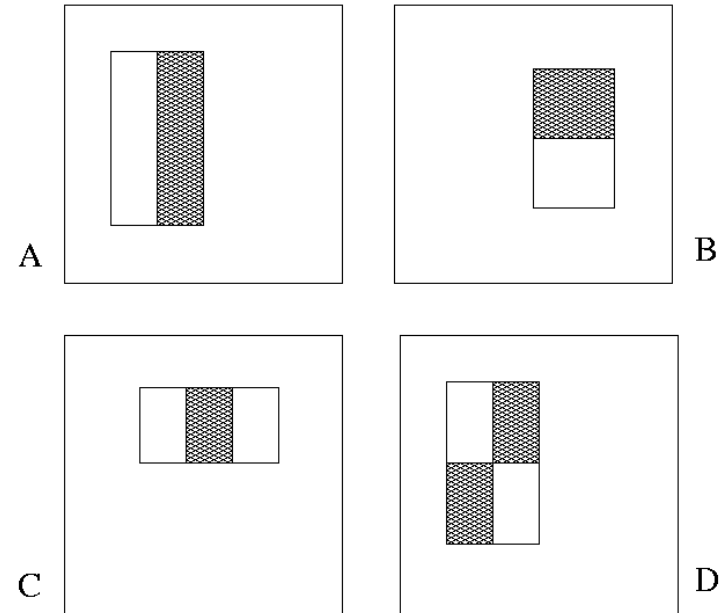
# Image Features

“Rectangle filters”



*Value =*

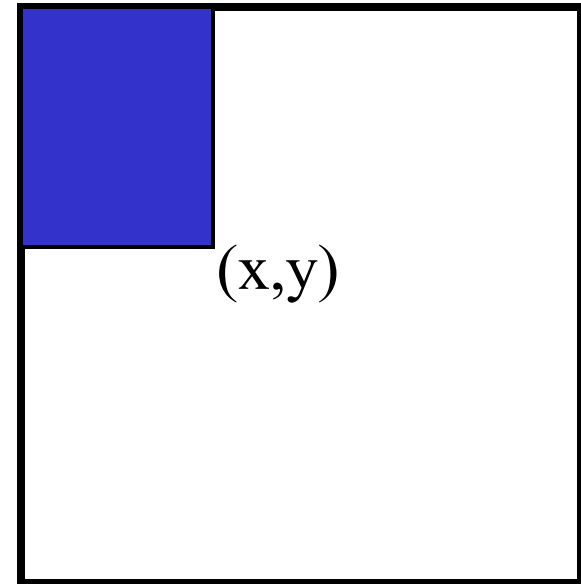
$$\sum (\text{pixels in white area}) - \sum (\text{pixels in black area})$$



4 feature examples,  
each return a scalar  
value

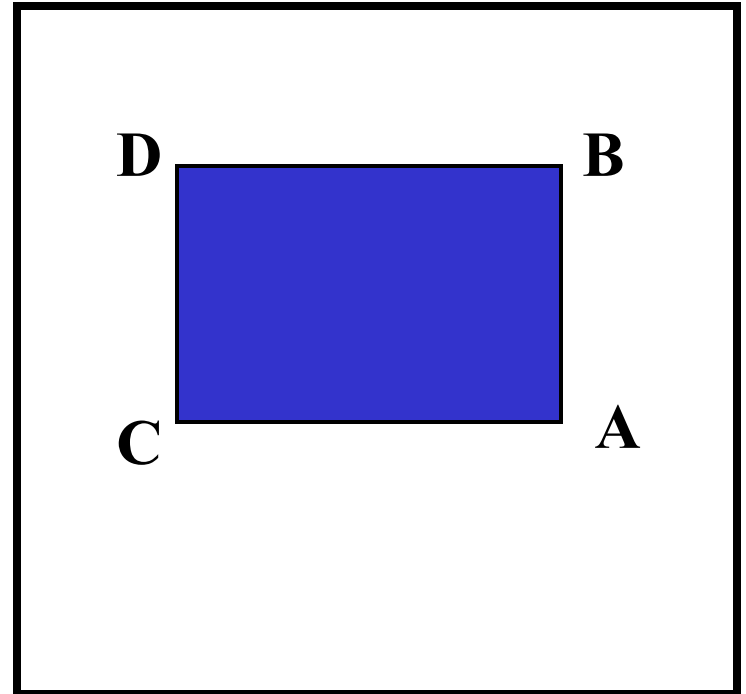
# Fast computation with integral images

- The *integral image* computes a value at each pixel  $(x, y)$  that is the sum of the pixel values above and to the left of  $(x, y)$ , inclusive
- This can quickly be computed in one pass through the image



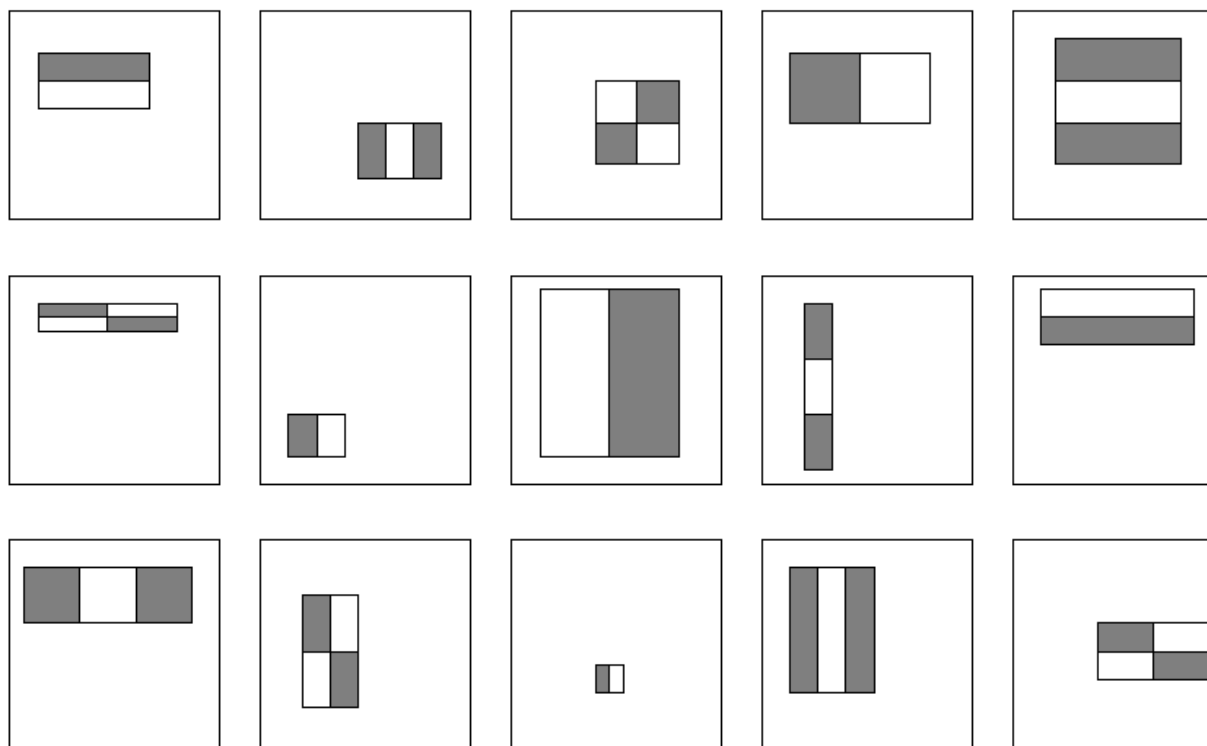
## Computing sum within a rectangle

- 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:  
$$\text{sum} = A - B - C + D$$
- Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision



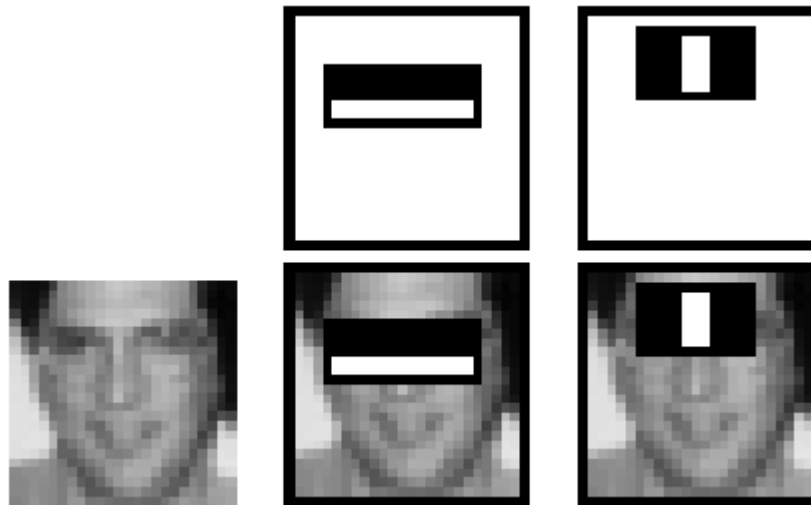
# Feature selection

- For a 24x24 detection region, the number of possible rectangle features is  $\sim 180,000!$



# Training

- Not all those features will generate discriminating info for face detection
- Need to find a set of features (4 types, size, location) that are useful for detecting faces
- The algorithm is called “Boosting”
- The first 2 most useful feature learnt for a given detection window:



# Boosting

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
- *Weak learner*: classifier with accuracy that need be only better than chance
- We can define weak learners based on rectangle features:

# Boosting

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
- *Weak learner*: classifier with accuracy that need be only better than chance
- We can define weak learners based on rectangle features:

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

Diagram annotations for the equation above:

- value of rectangle (points to  $f_t(x)$ )
- feature (points to  $f_t(x)$ )
- parity (points to  $p_t$ )
- threshold (points to  $\theta_t$ )
- window (points to  $h_t(x)$ )

# Boosting outline

- **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)**

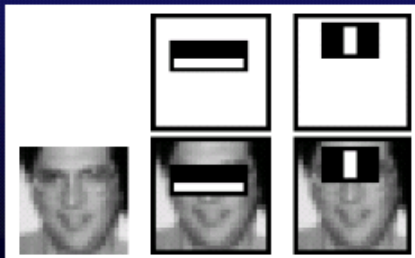
# Boosting for Face Detection

## ■ For $t=1, \dots, T$

- ◆ Construct a weak classifier using one single feature  $h_t$   $h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$  where  $p_j$  is a parity bit and  $\theta_j$  is a threshold
- ◆ For each feature  $j$ , train a classifier  $h_j$ , the error is evaluated with respect to  $w_t$ ,  $\varepsilon_t = \sum_i w_i |h_j(x_i) - y_i|$
- ◆ Choose the classifier  $h_t$ , with the minimum error  $\varepsilon_t$
- ◆ Update the weights:  $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$ ,  $e_i = 0$  if  $x_i$  is correctly classified, and  $e_i = 1$  otherwise.  $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$

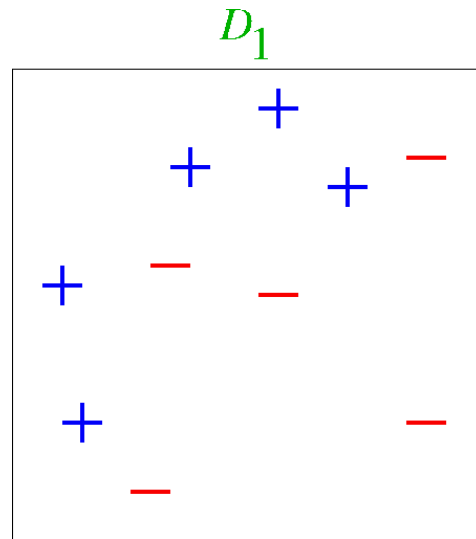
## ■ Final classifier:

$$h_j(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t, \alpha_t = \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}$$

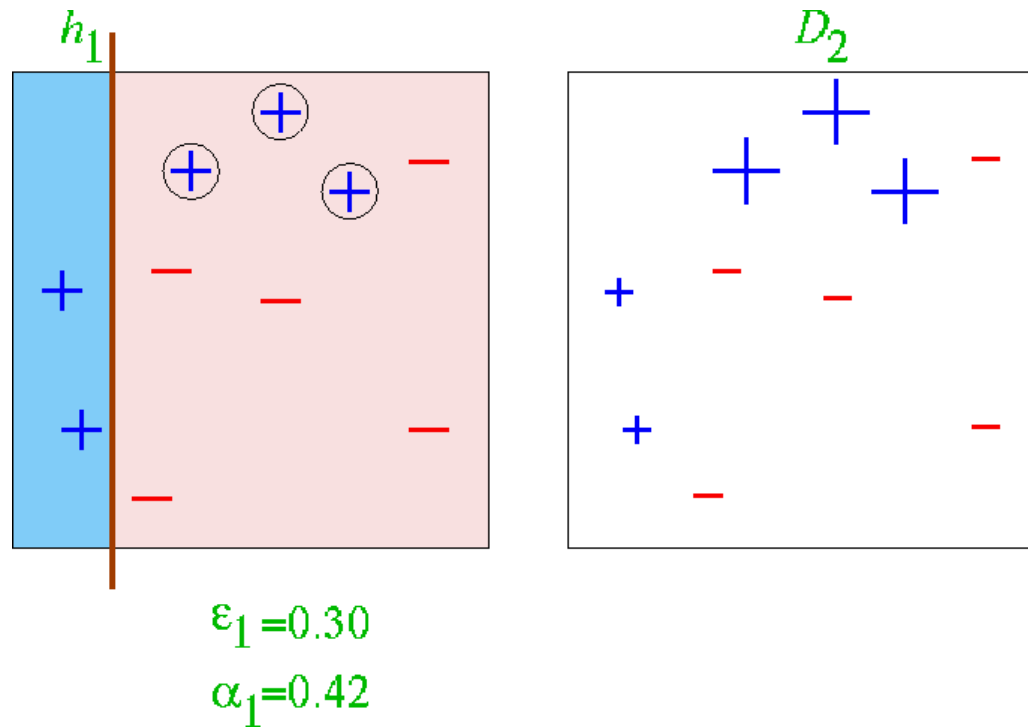


The top two boxlets selected by Adaboost

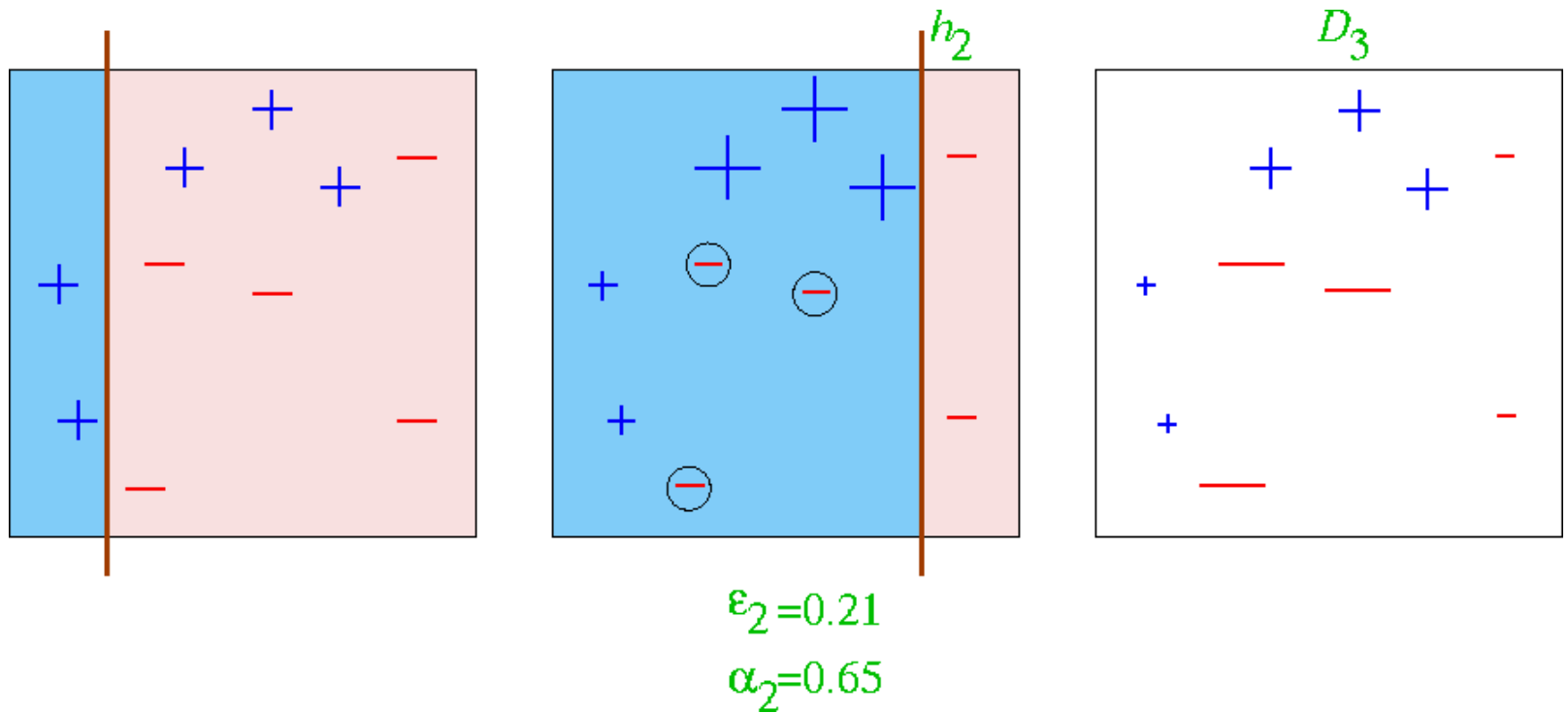
# Boosting Example (adapted from prof. p. smyth's slides)



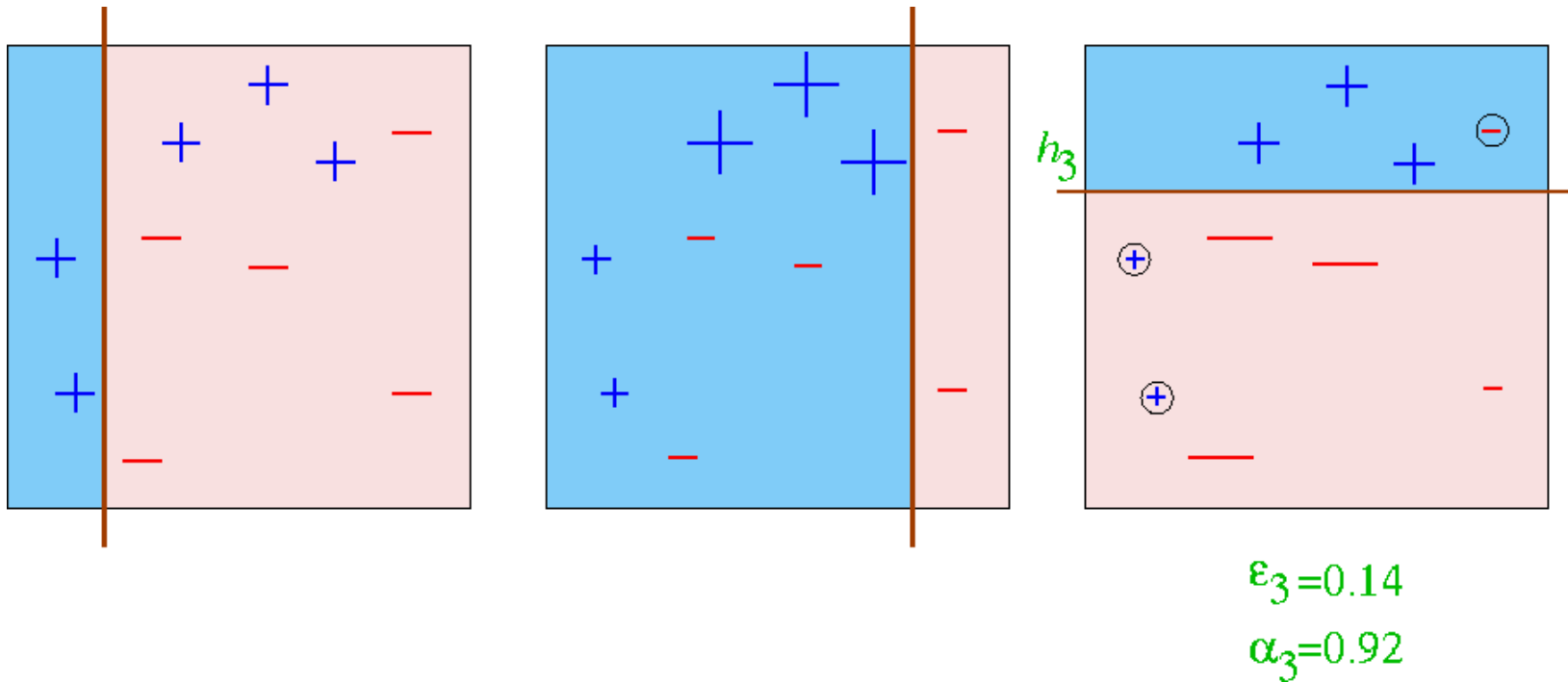
# First classifier



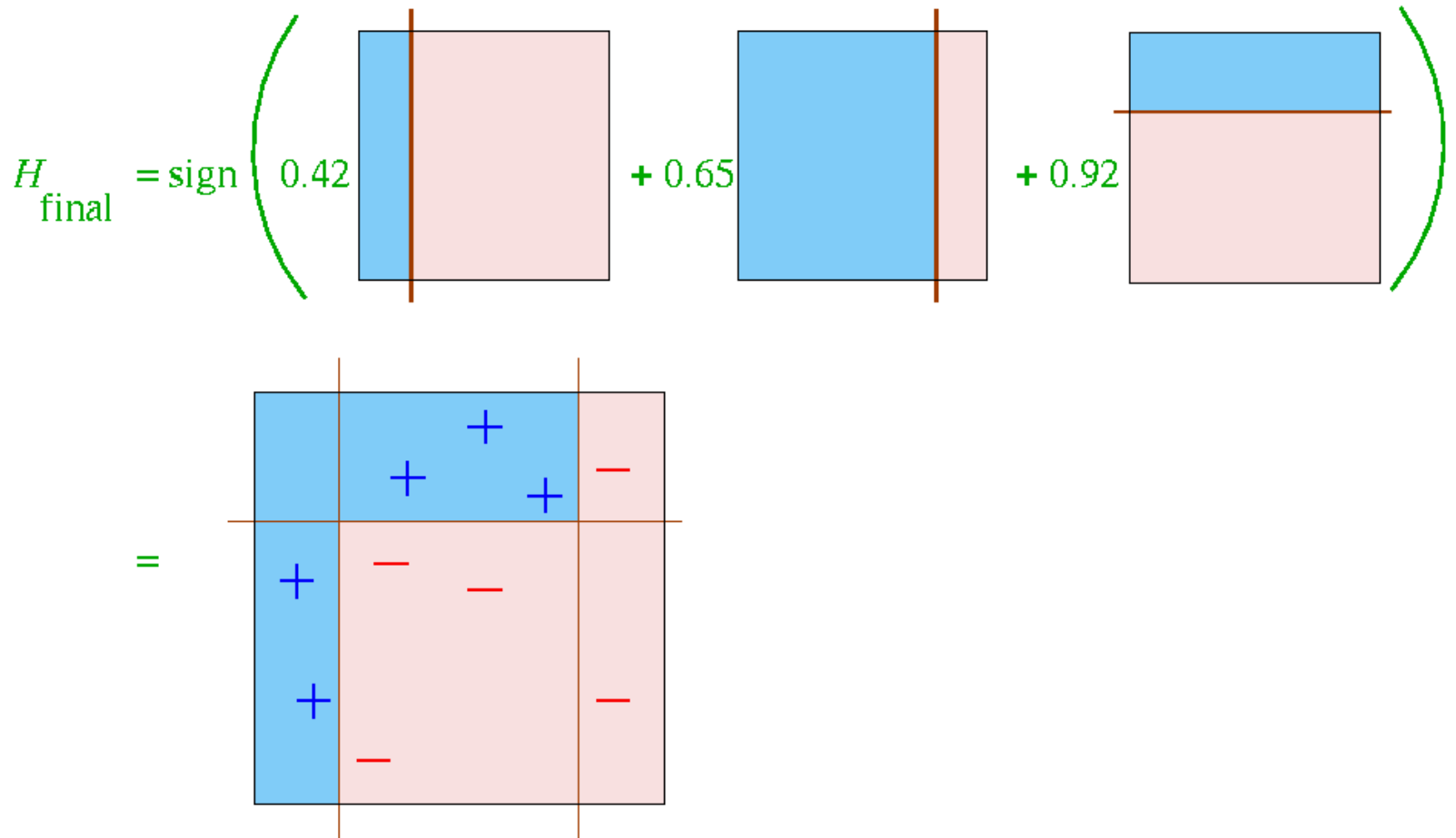
# First 2 classifiers



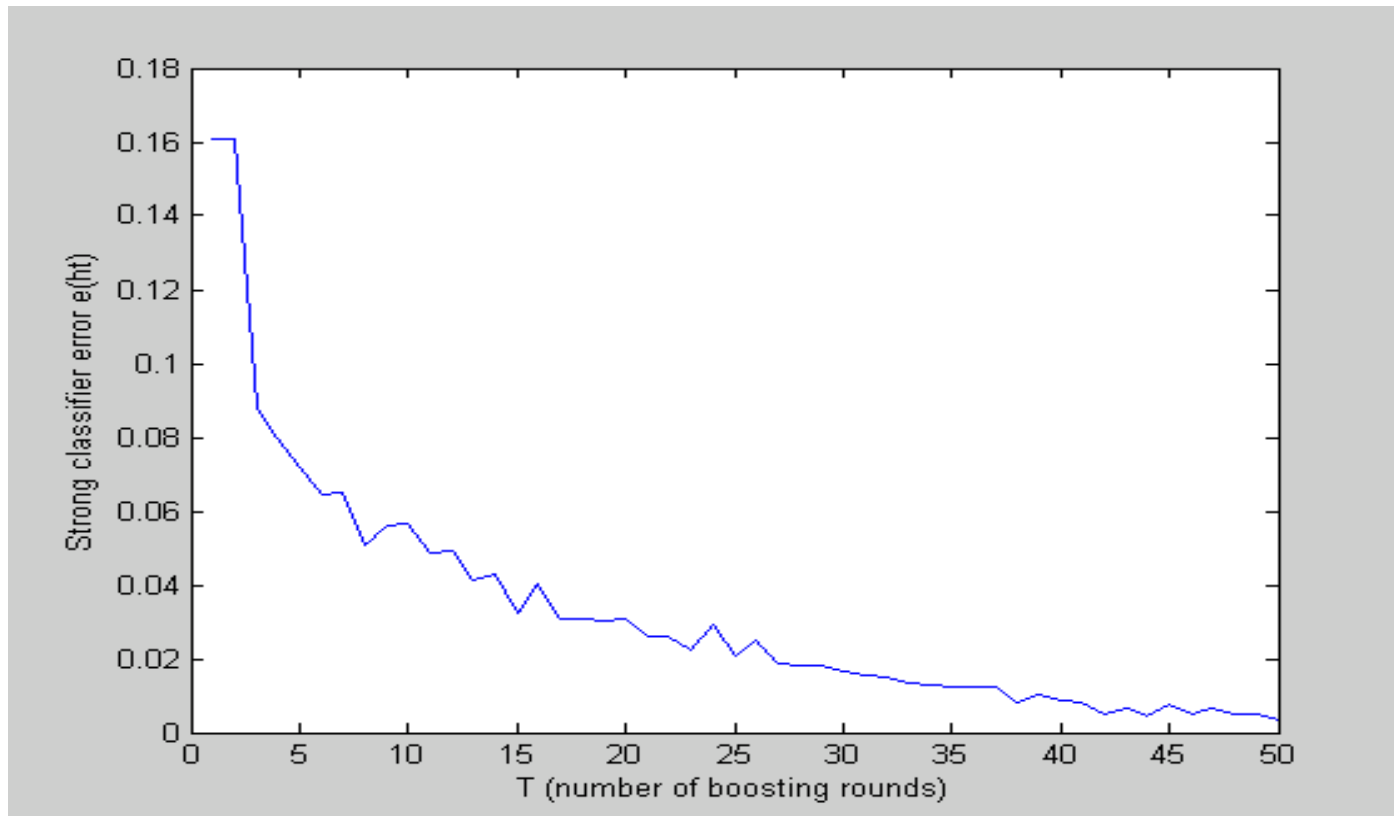
# First 3 classifiers



# Final Classifier learned by Boosting

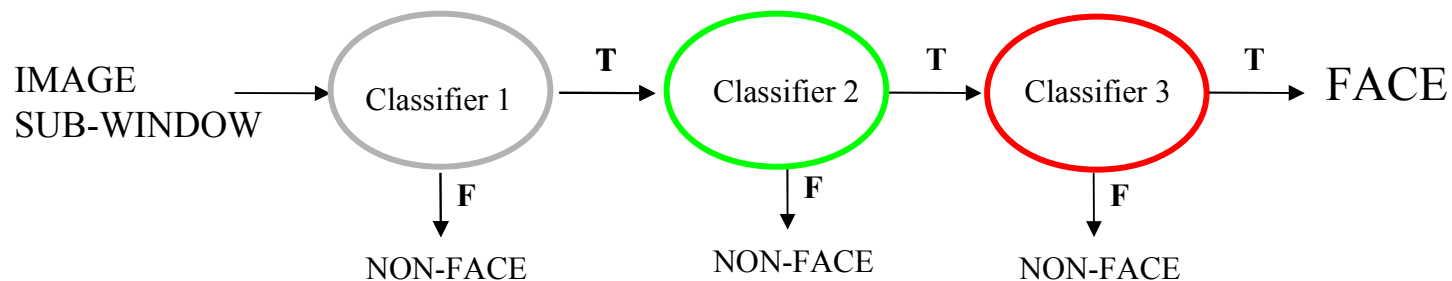


# Reduction in Error as Boosting adds Classifiers



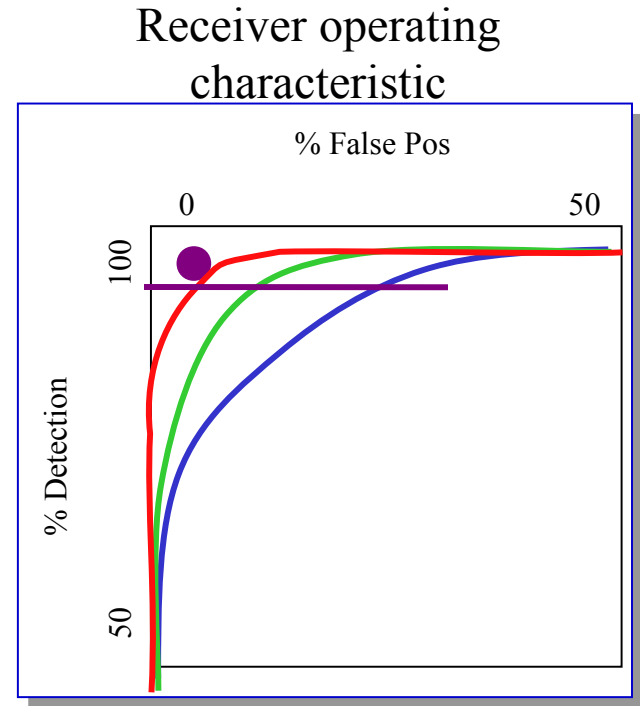
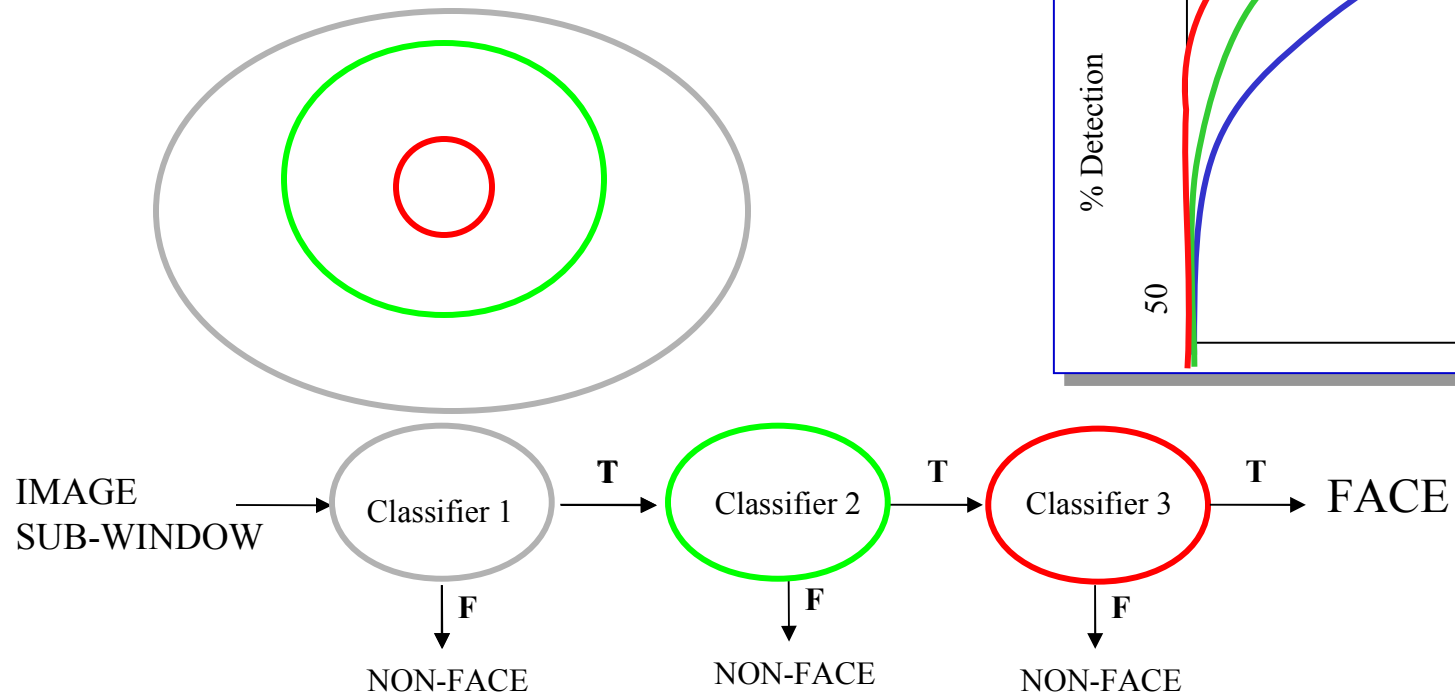
# Cascading classifiers

- 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:



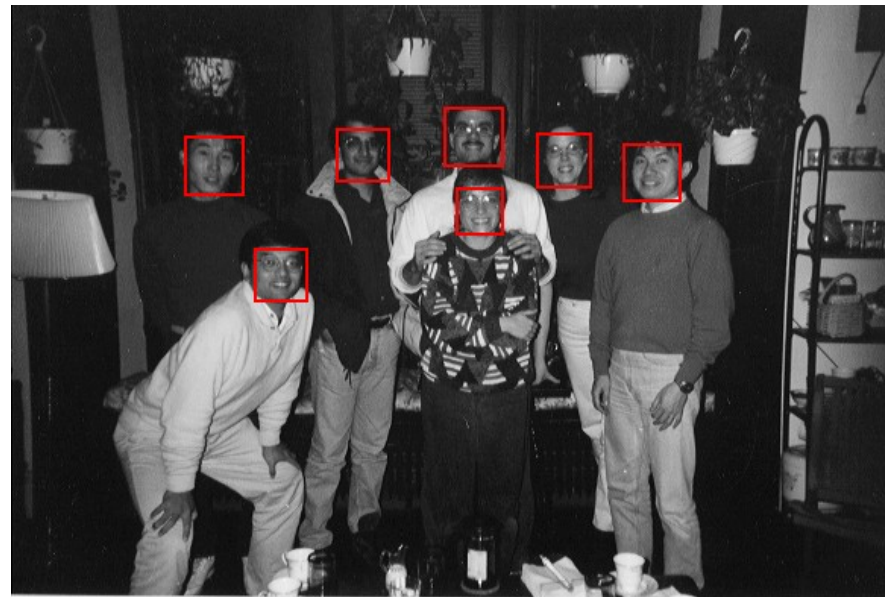
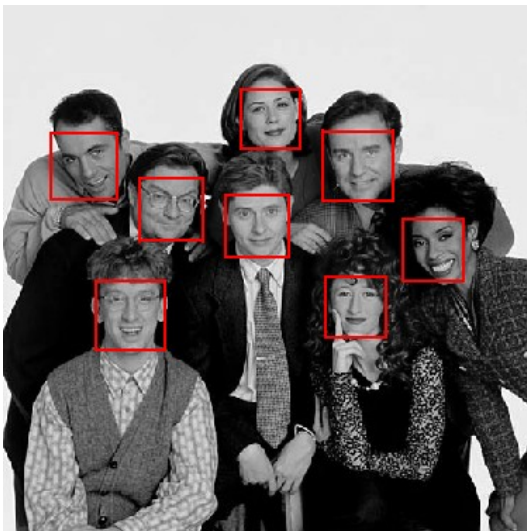
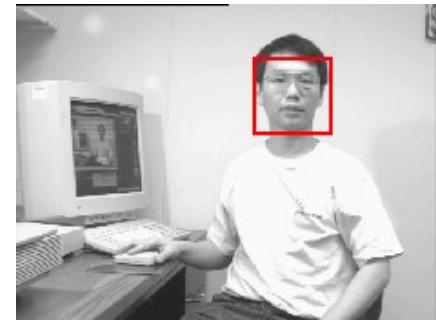
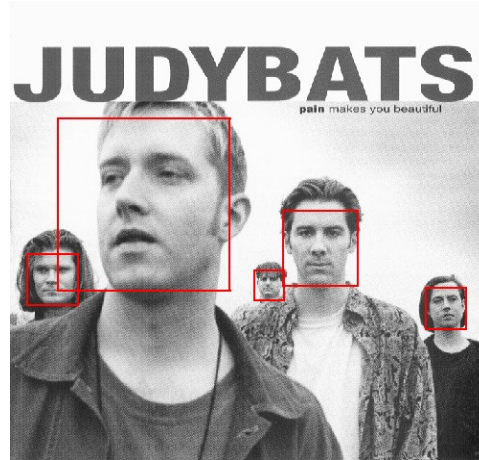
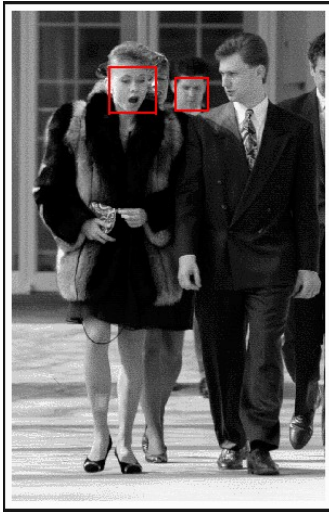
# The implemented system

- **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



(Most slides from Paul Viola)

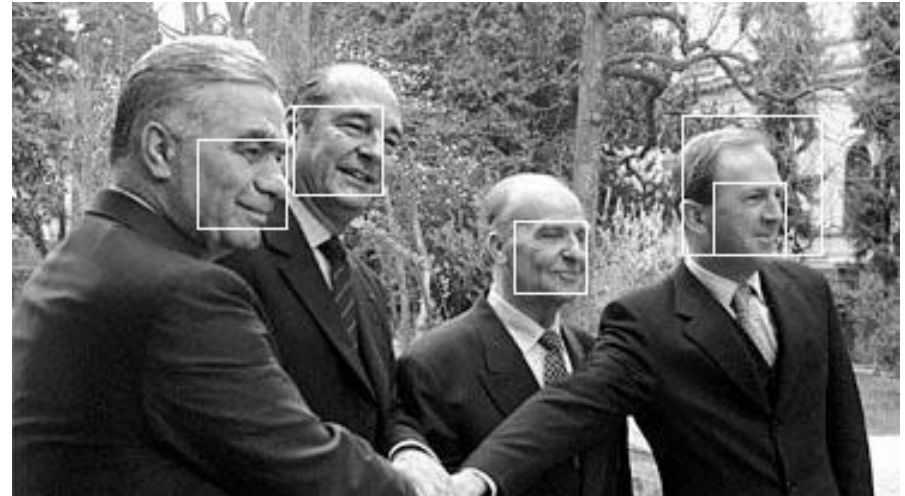
# Output of Face Detector on Test Images



# Other detection tasks

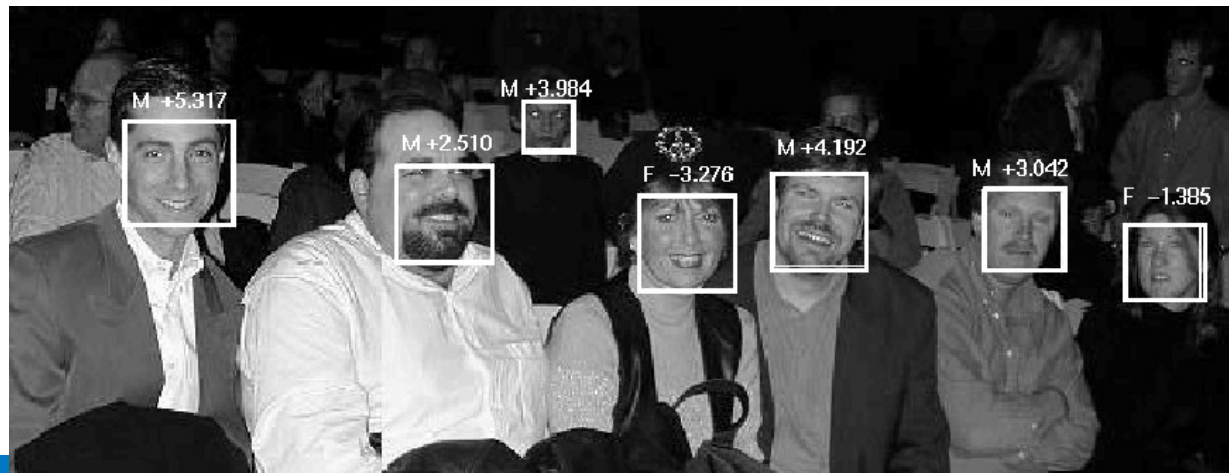


Facial Feature Localization

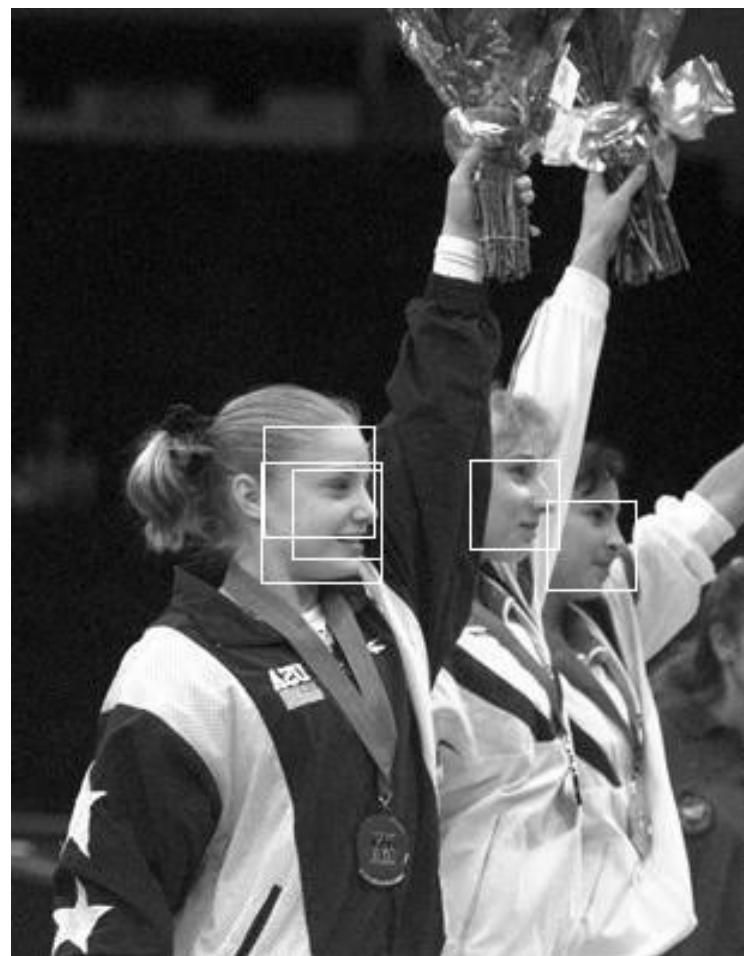


Profile Detection

Male vs.  
female



# Profile Detection



# Face Detection Summary

- The problem of face detection in image/video is to locate the face areas
- Color based approach fast but not accurate enough
- Appearance based solutions
  - SVM + bootstrapping: refine the SVM boundaries with more useful support vectors
  - KNN SVM: fit multiple SVM models
  - Boosting: judiciously using large set of simple classifiers, fast, state of art solution. (demo)