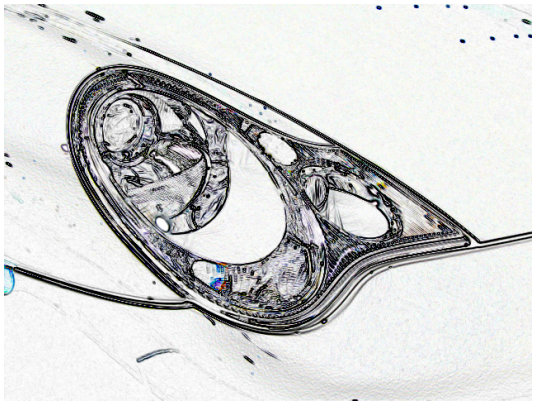
CSE578: Computer Vision

Spring 2016:

Object Detection and Recognition: Face Detection


Anoop M. Namboodiri

Center for Visual Information Technology

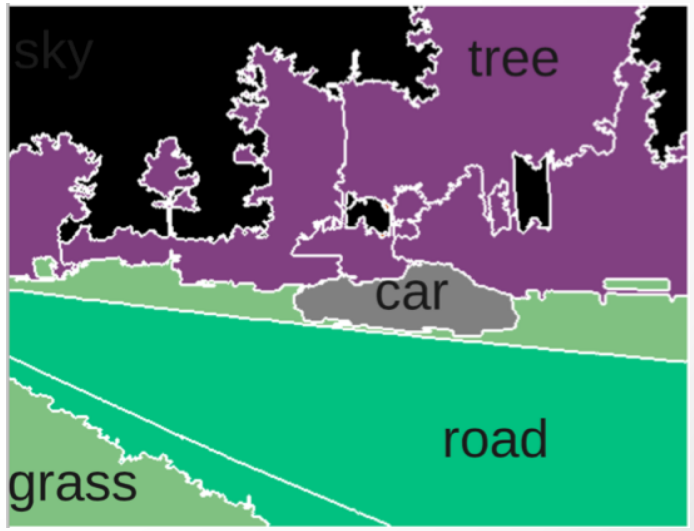
IIIT Hyderabad, INDIA

Labeling Problems in Vision

- Labeling Pixels
  - Segmentation (Semantic)
  - Stereo
  - Normal Estimation
  - Restoration/Enhancement
- Labeling Windows
  - Object Detection
  - Object Recognition
- Other Problems
  - Matching: Registration, Tracking

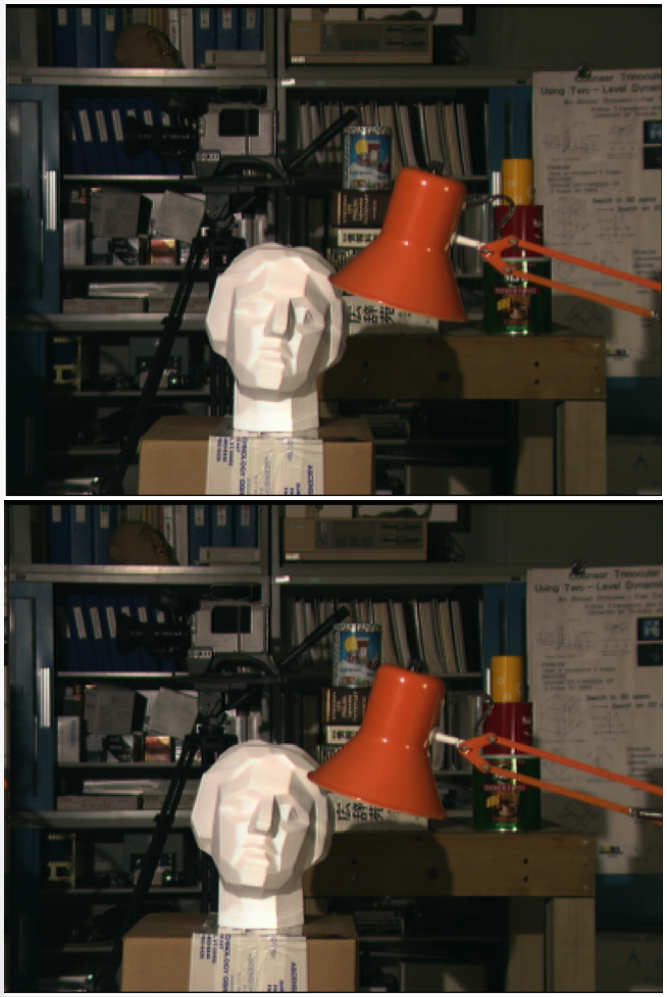
- Posed as Optimization or Energy Minimization
  - MLE/MAP
  - MRF
  - DTW
  - Convex
  - Linear Programming
  - MCMC
  - Simulated Annealing

Segmentation as Labeling

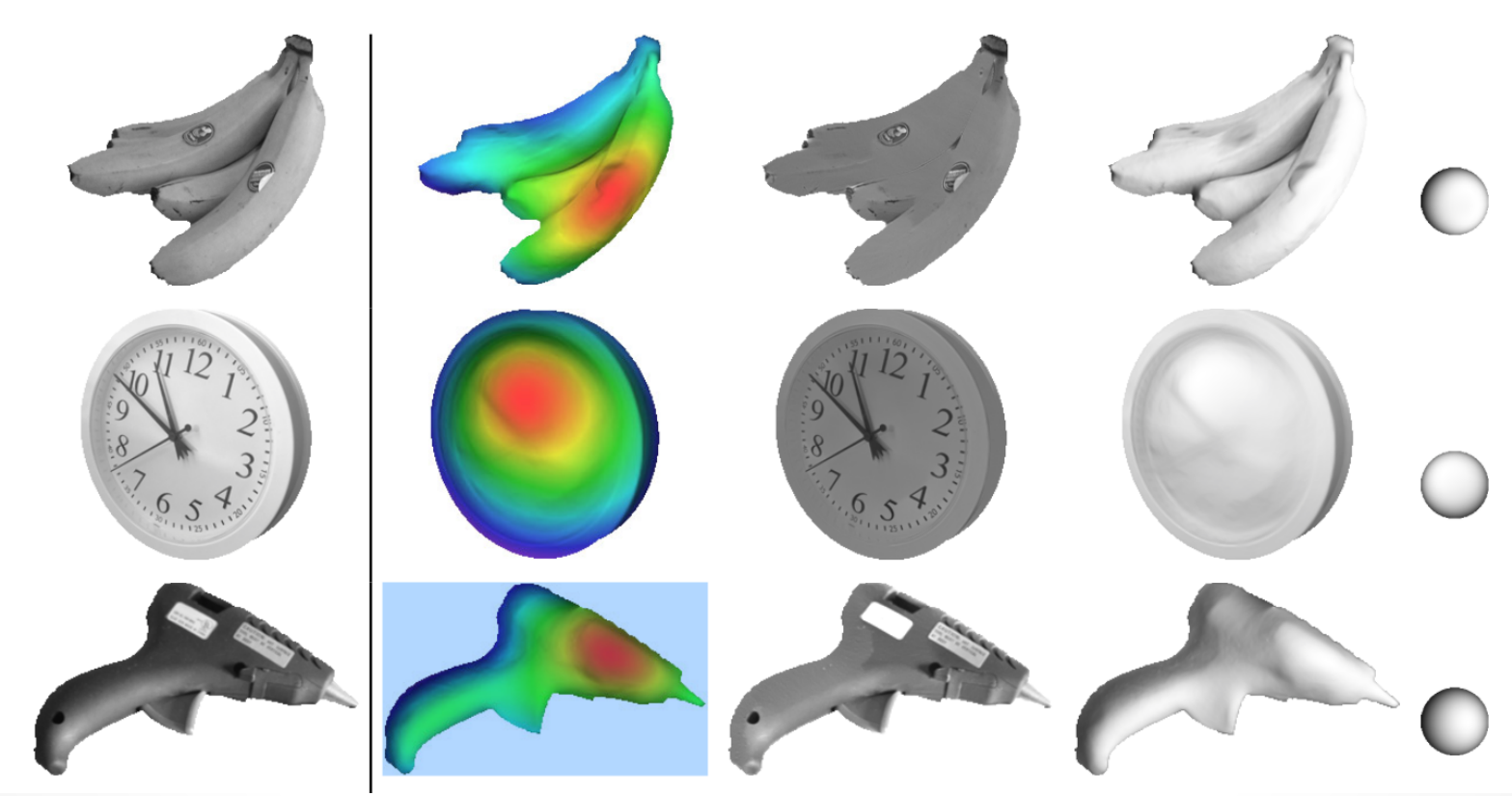


Stereo as Labeling

Each pixel is assigned a disparity (depth). DTW or MRF for optimization



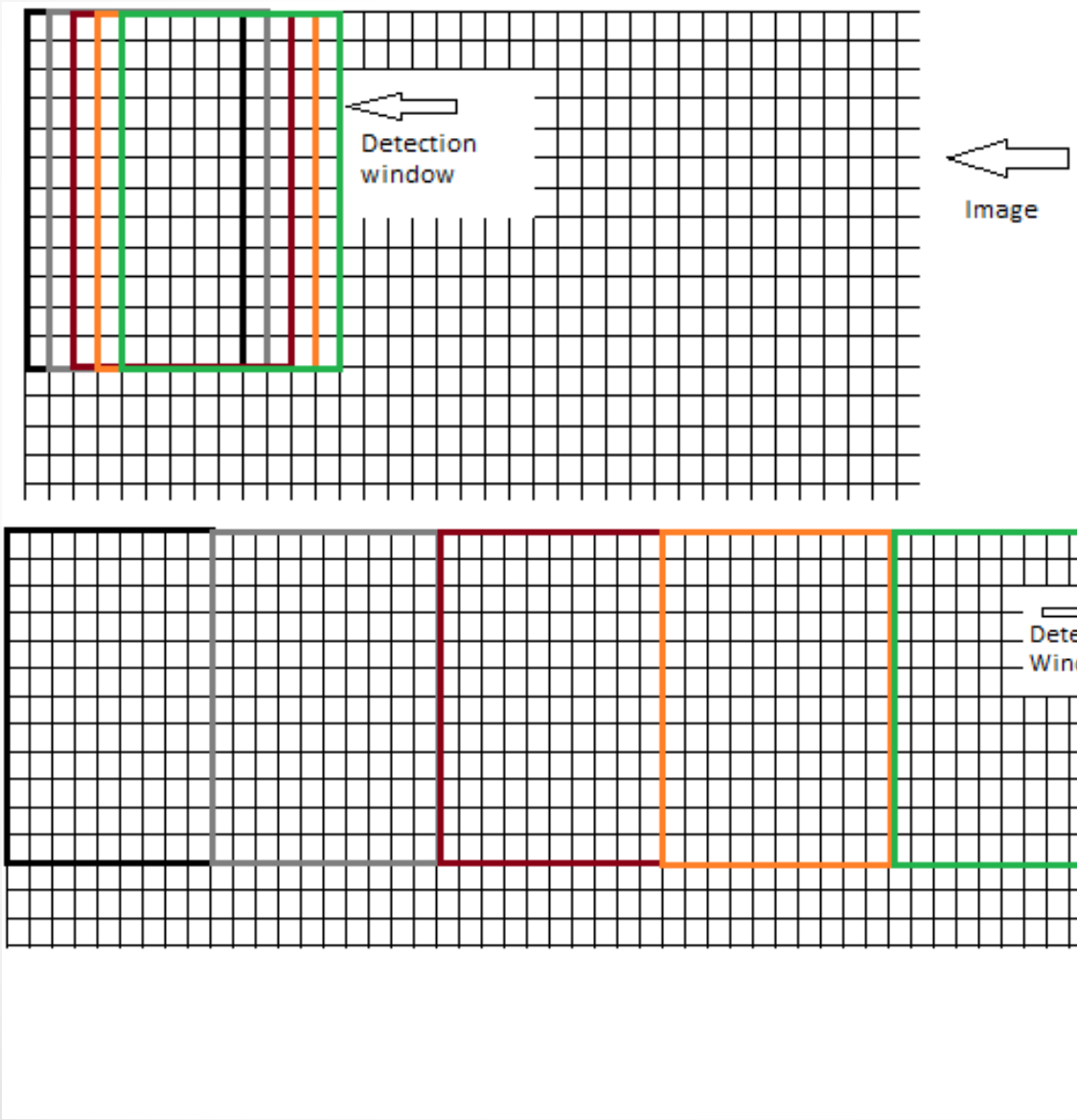
Normal/Shape Estimation



Estimate normal and albedo at every pixel: What should we optimize?

Image Restoration/Enhancement

Detection and Recognition



Classify each window

Face Detection using Eigenfaces

- Given an unknown image Γ

Step 1: compute $\Phi = \Gamma - \Psi$

Step 2: compute $\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \Phi) \quad (where \| u_i \| = 1)$

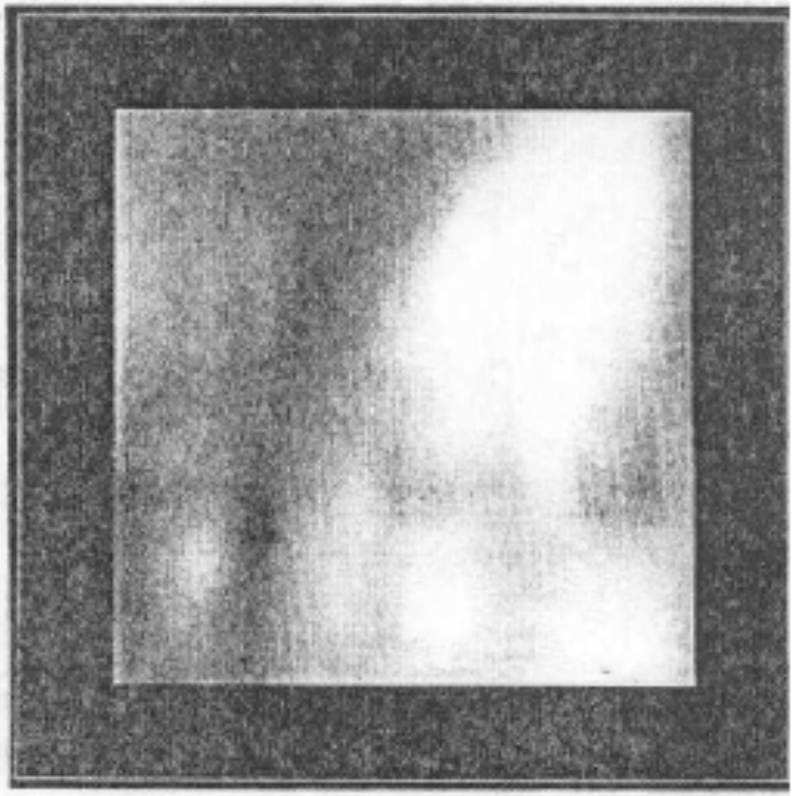
Step 3: compute $e_d = \|\Phi - \hat{\Phi}\|$

Step 4: if $e_d < T_d$, then Γ is a face.

The distance e_d is called distance from face space (dffs)

Distance from Face Space

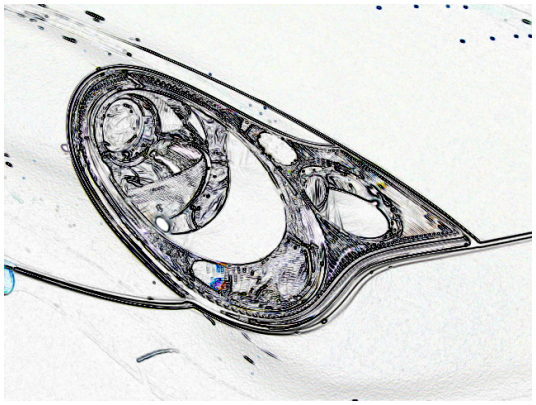
Visualize dffs: $e_d = \|\Phi - \hat{\Phi}\|$



CSE578: Computer Vision

Spring 2016:

Real-time Face Detection: Viola and Jones

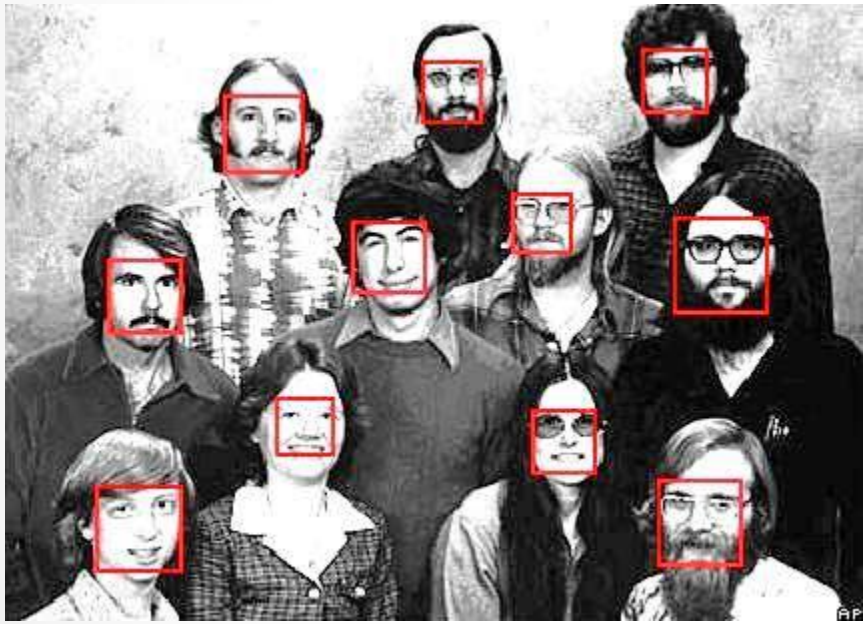



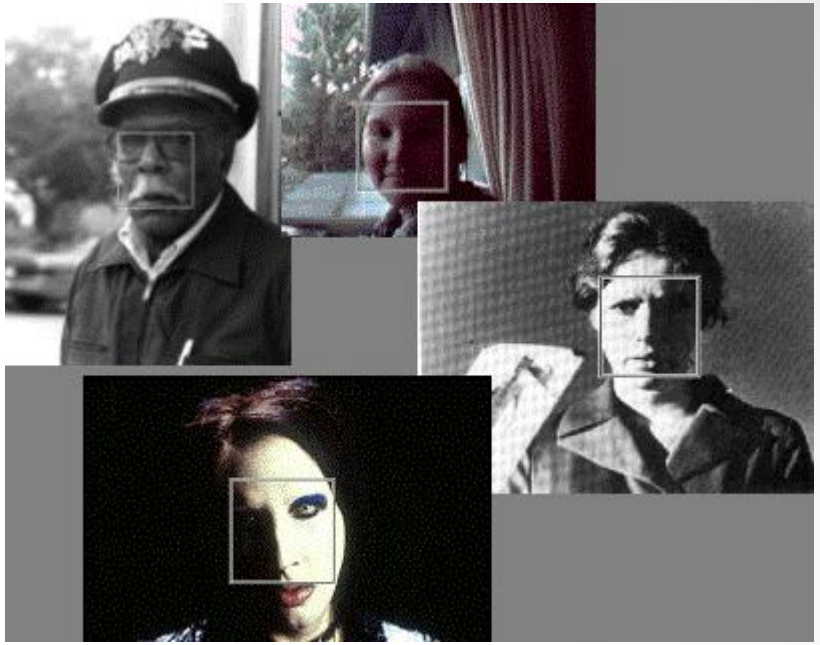

Anoop M. Namboodiri

Center for Visual Information Technology

IIIT Hyderabad, INDIA

The Task of Face Detection



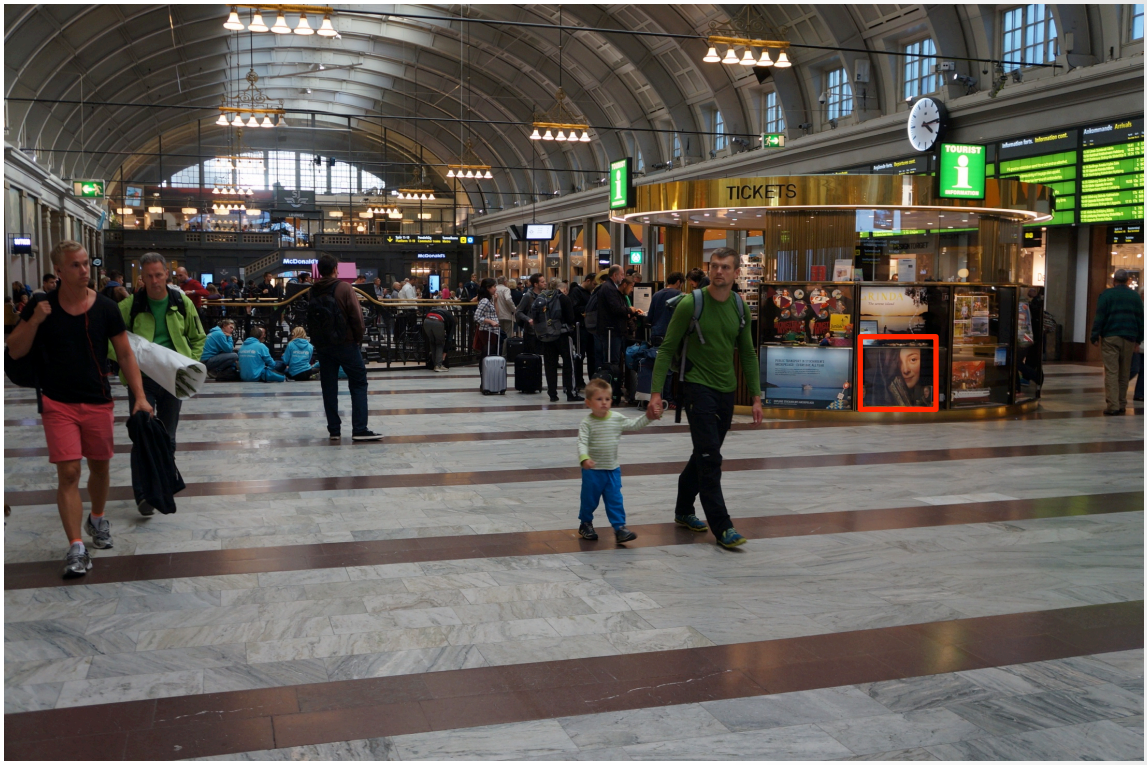


Basic Idea

Slide a window across image and evaluate a face model at every location

How Many Windows (Speed)

- 1280×1024 image; 24×24 to 1024×1024 windows



- # of Windows?
  - ~ 1 million locations
  - 18 scales/window sizes at 1.25 multiples
  - 14.5 million potential face candidates
  - Features to be extracted for each candidate window

Accuracy (FPR)

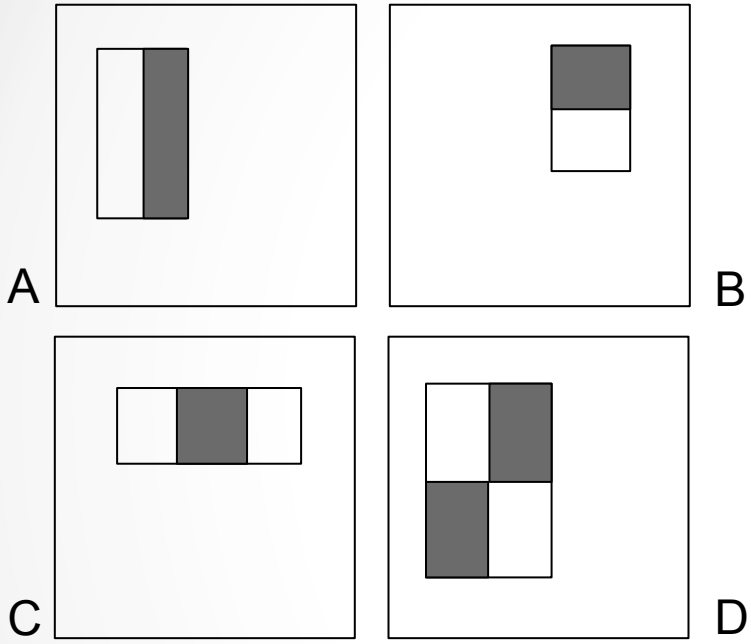
- Classify each as face or non-face
  - What should be the accuracy (False Positive Rate)?
- Faces are rare: 0–10 per image
  - For computational efficiency, we should try to spend as little time as possible on the non-face windows
  - To avoid having a false positive in every image, the false positive rate has to be less than 10^{-6}

The Viola/Jones Face Detector

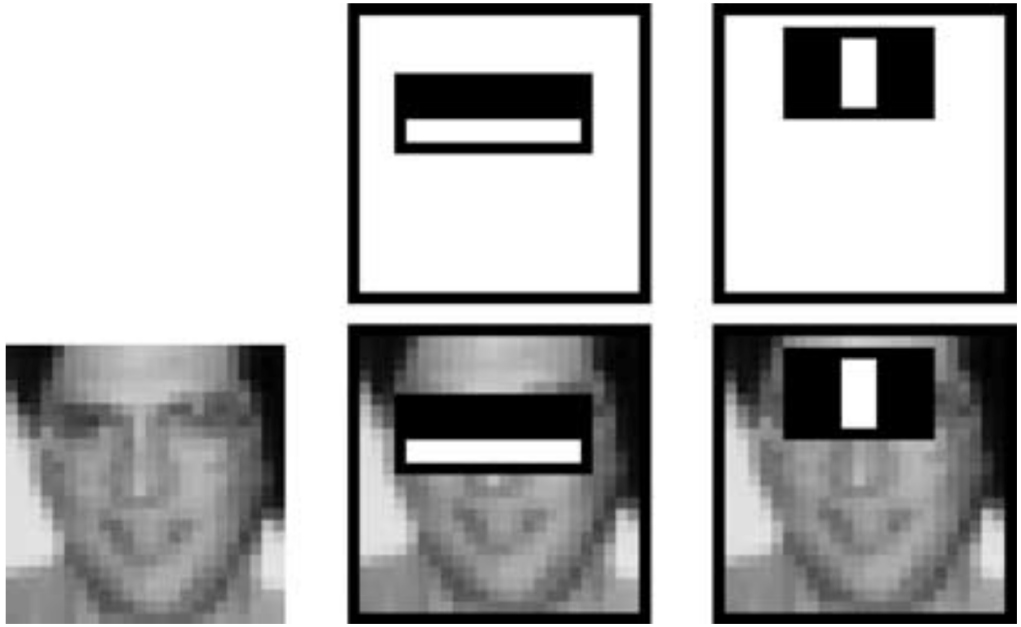
- A seminal approach to real-time object detection
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

- P. Viola and M. Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features.* CVPR 2001.
- P. Viola and M. Jones. *Robust Real-Time Face Detection.* IJCV 57(2), 2004.

Image Features



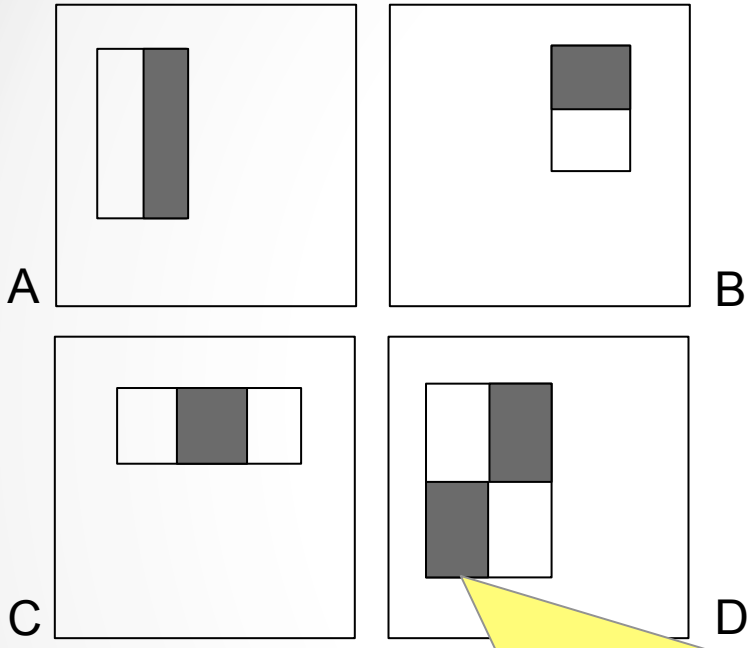
Rectangular filters



$$f(x,y) = \sum_i p_b(i) - \sum_i p_w(i)$$

Local features: Subtract sum of pixels in white area from the sum of pixels in black area; 2-rectangle features (A and B), 3-rectangle feature (C) and 4-rectangle feature (D)

Image Features



Rectangular fil

Too many features

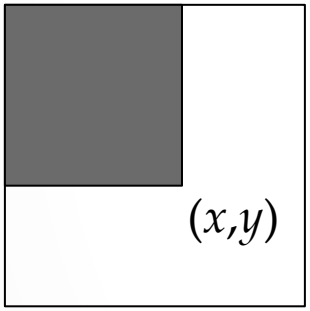
In a 24 x 24 patch with 4x4 detector, there are over 160,000 locations for rectangles

$$\sum_i p_w(i)$$

Local features: …om the sum of pixels in bl… …rectangle feature (C) …

Image Features

- Integral Image: An intermediate representation of the image for rapid calculation of rectangle features
- s(x,y) is the cumulative row sum, s(x,-1) =0 and ii(-1, y) = 0; the integral image can be computed in one pass over the original image

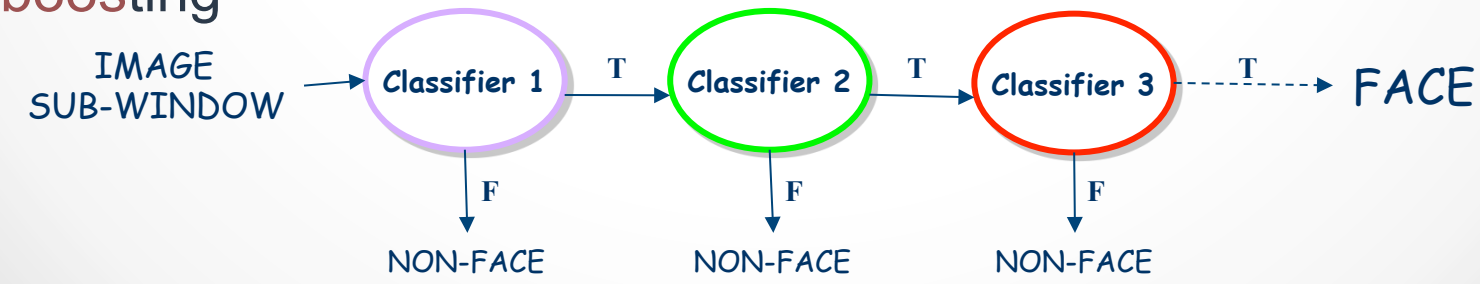


$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x', y'),$$

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

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

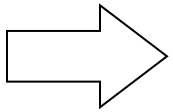
- Reject non-face windows through a cascade of classifiers and boosting



Example

IMAGE

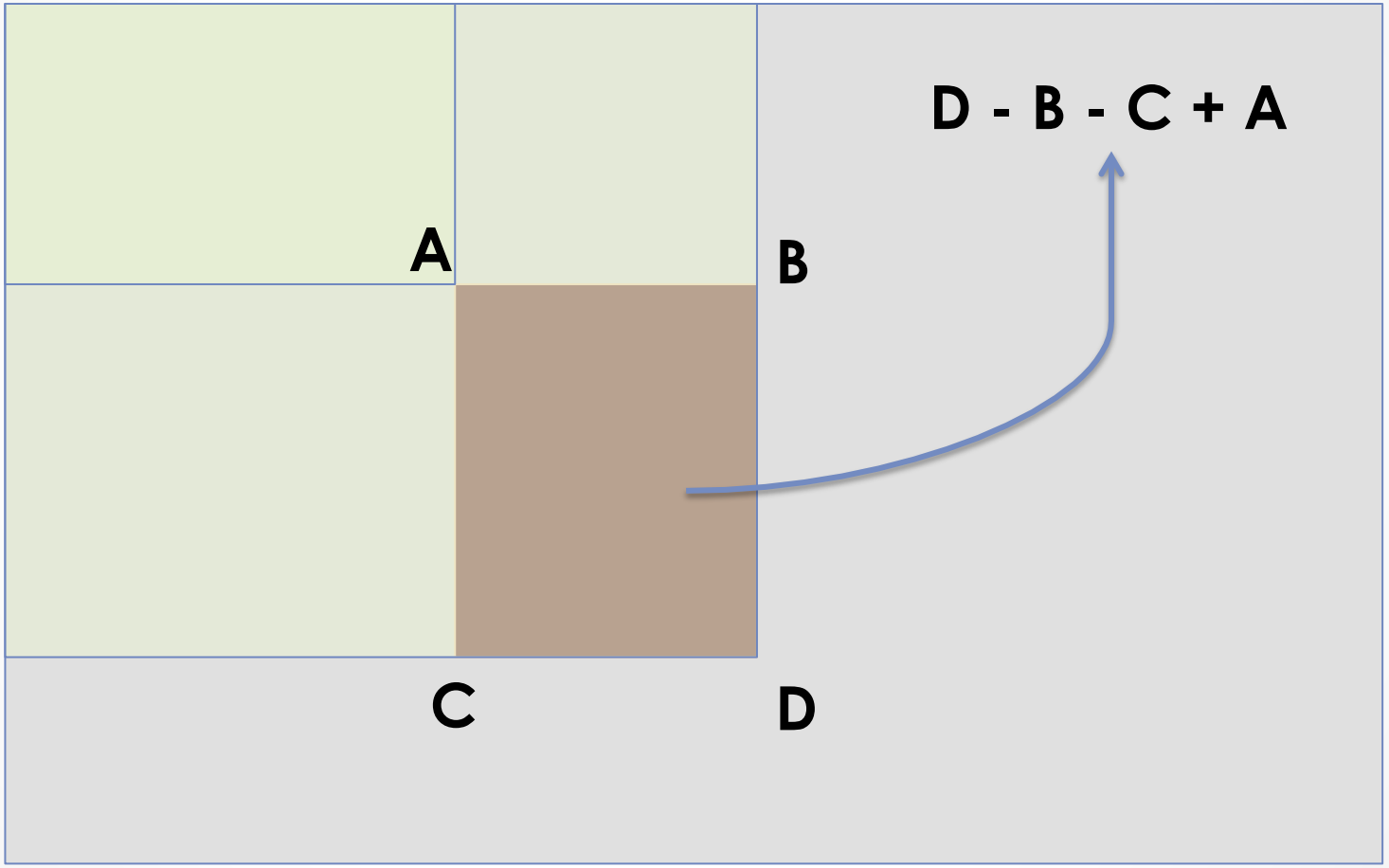
0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0



INTEGRAL IMAGE

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

Computing Rectangular Features



Rectangle Features from Integral Image

- Rectangle features are somewhat primitive and coarse, they are sensitive to presence of edges, bars, and other simple structure
- Generating a large no. of these features and their computational efficiency compensates for this.

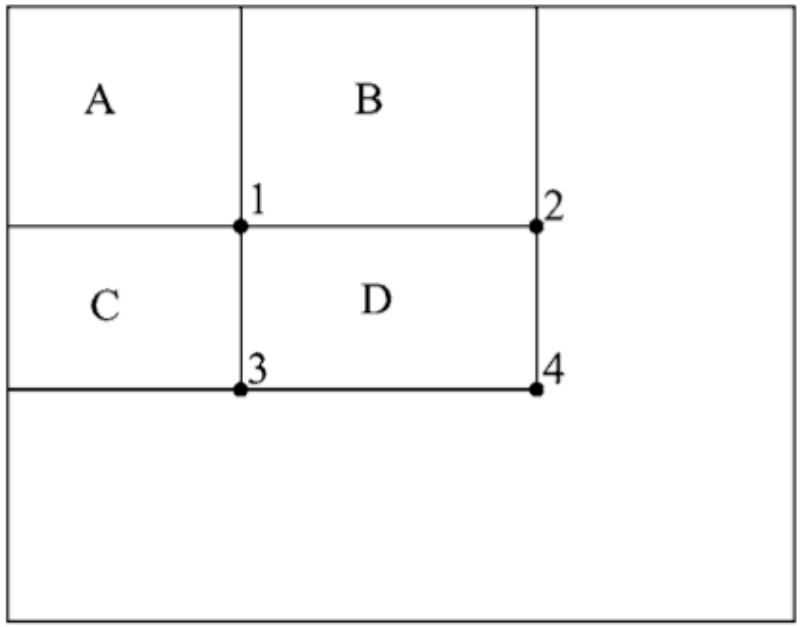


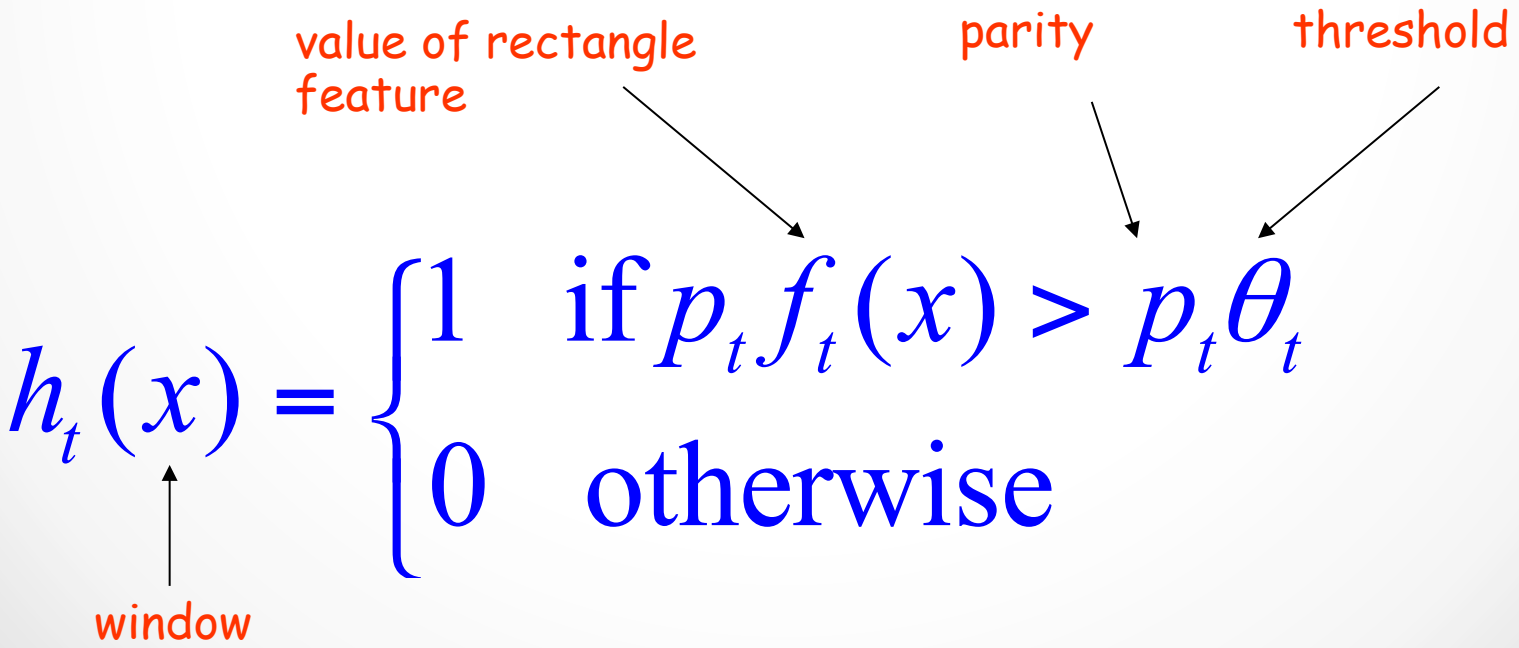
Figure 3. The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

Face Detector

- Scan the input at many scales
- Starting at the base scale in which faces are detected at 24x24 pixels, a 384x288 pixel image is scanned at 12 scales each a factor 1.25 larger than the last
- Any rectangle feature can be evaluated at any scale and location in a few operations
- Face detection @15 fps for the entire image

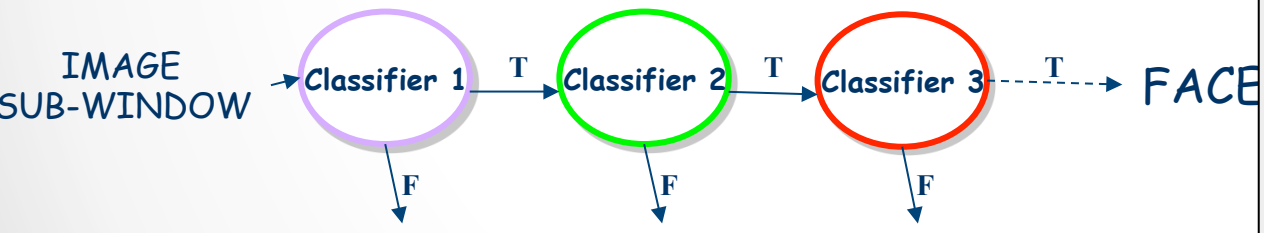
Weak Learners for Face Detection

- Hypothesis: A very small no. of 160K (x4) features for each image sub-window can be formed to find an effective classifier (face vs. non-face)
- AdaBoost is used both to select the features and train the classifier
- Weak learner: a single rectangle feature that best separates positive and negative examples; so weak classifier is a thresholded single feature (can be viewed as a single node decision tree)



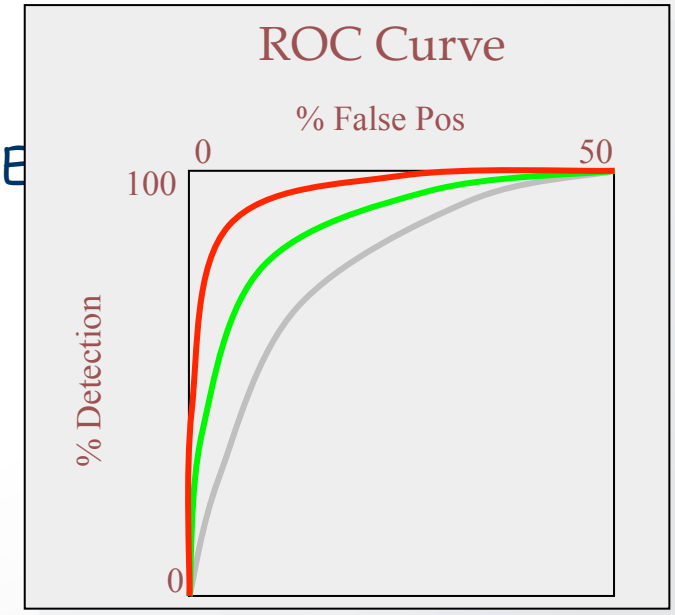
Cascade of classifiers

- Train each classifier with increasing number of features until the target false positive and detection rates are achieved
- Use false positives from current stage as the negative training examples for the next stage
- Classifiers are progressively more complex and have lower false positive rates



False positive rate of cascade $F = \prod_{i=1}^{K} f_i$

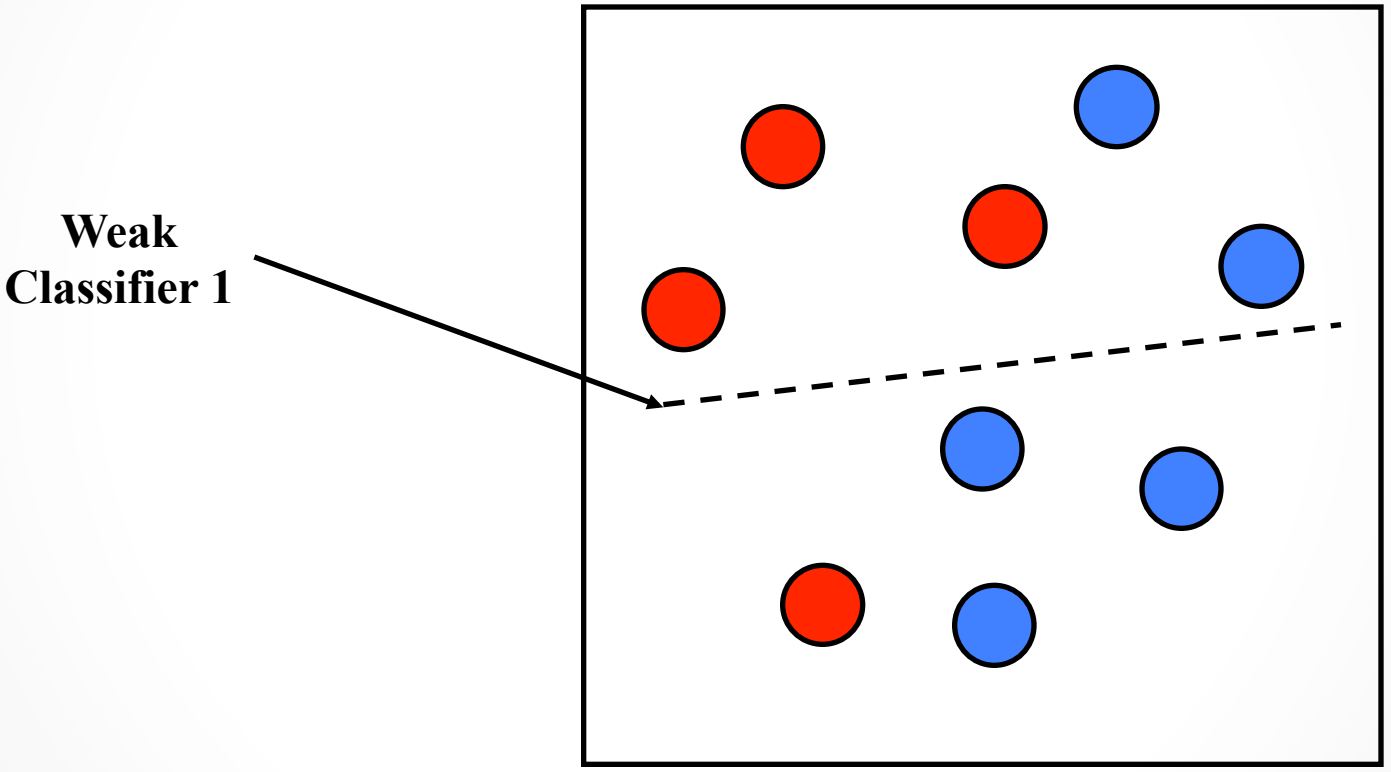
Detection rate of cascade $D = \prod_{i=1}^{K} d_i$



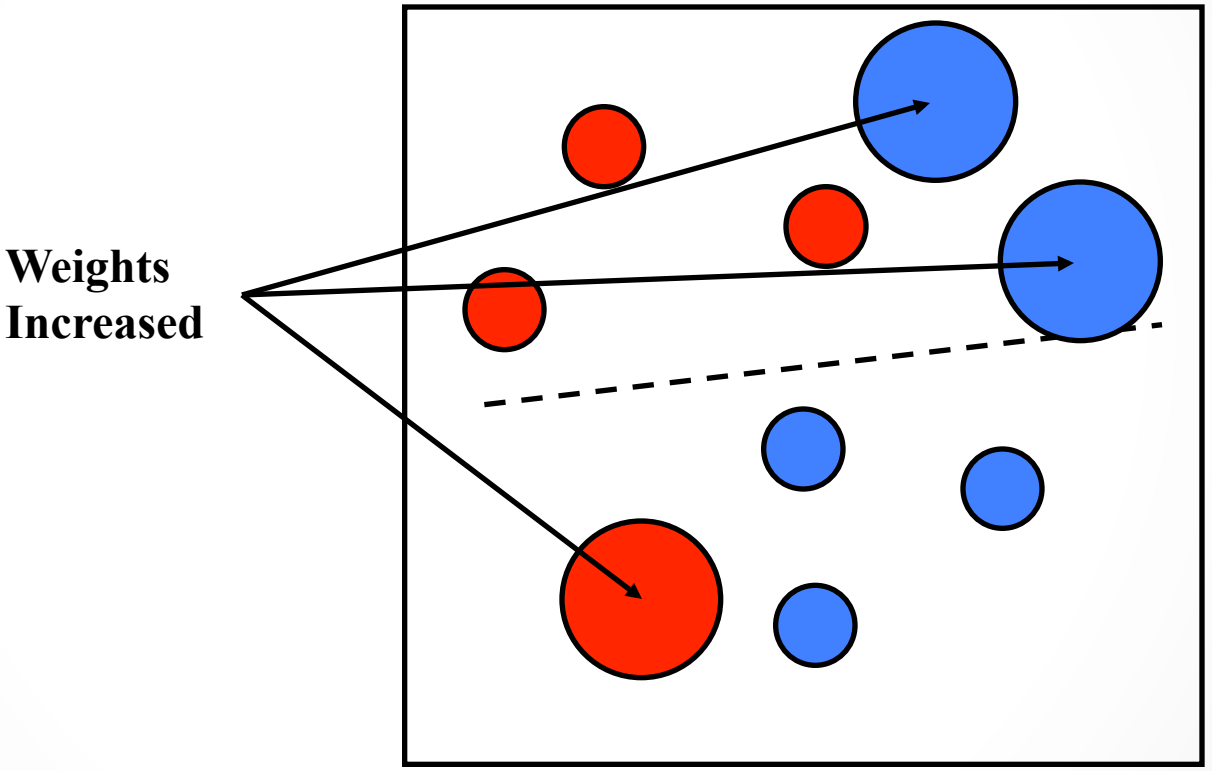
Boosting

- Training set contains face and nonface examples
  - Initially, with equal weight
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples
- Computational complexity of learning: $O(MNK)$
  - M rounds, N examples, K features

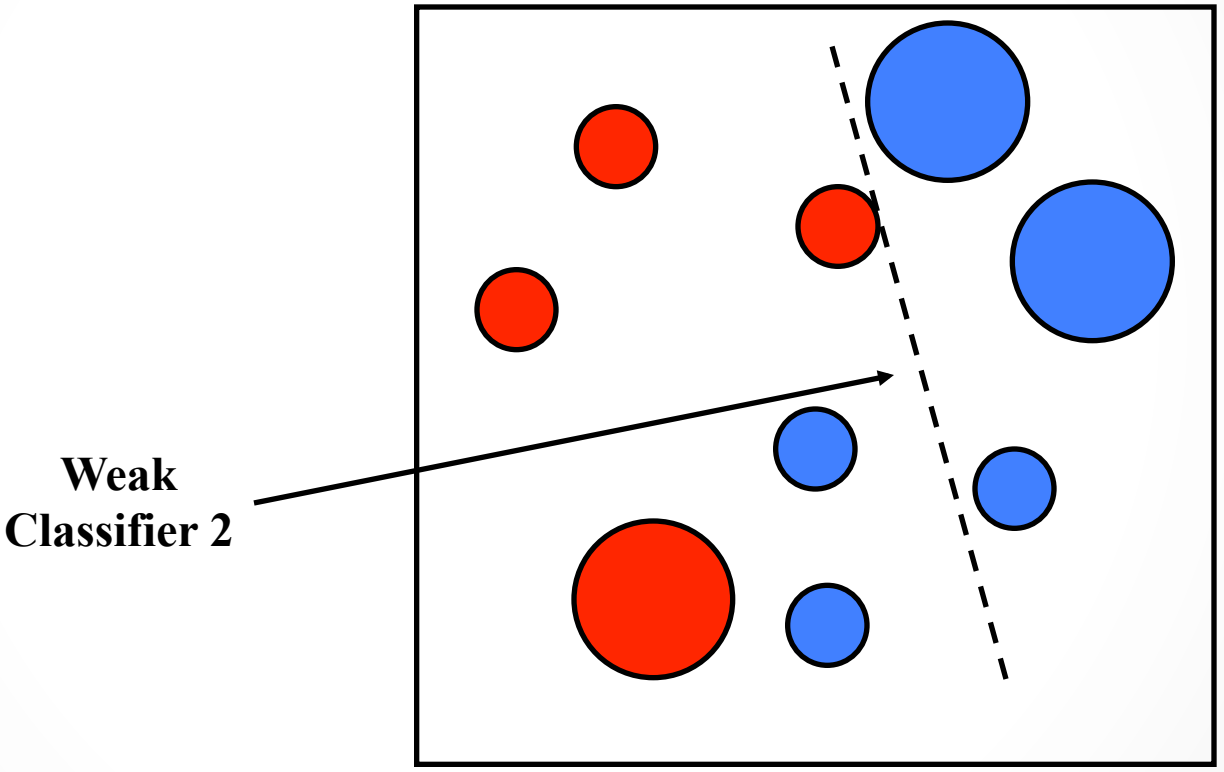
Boosting Illustration



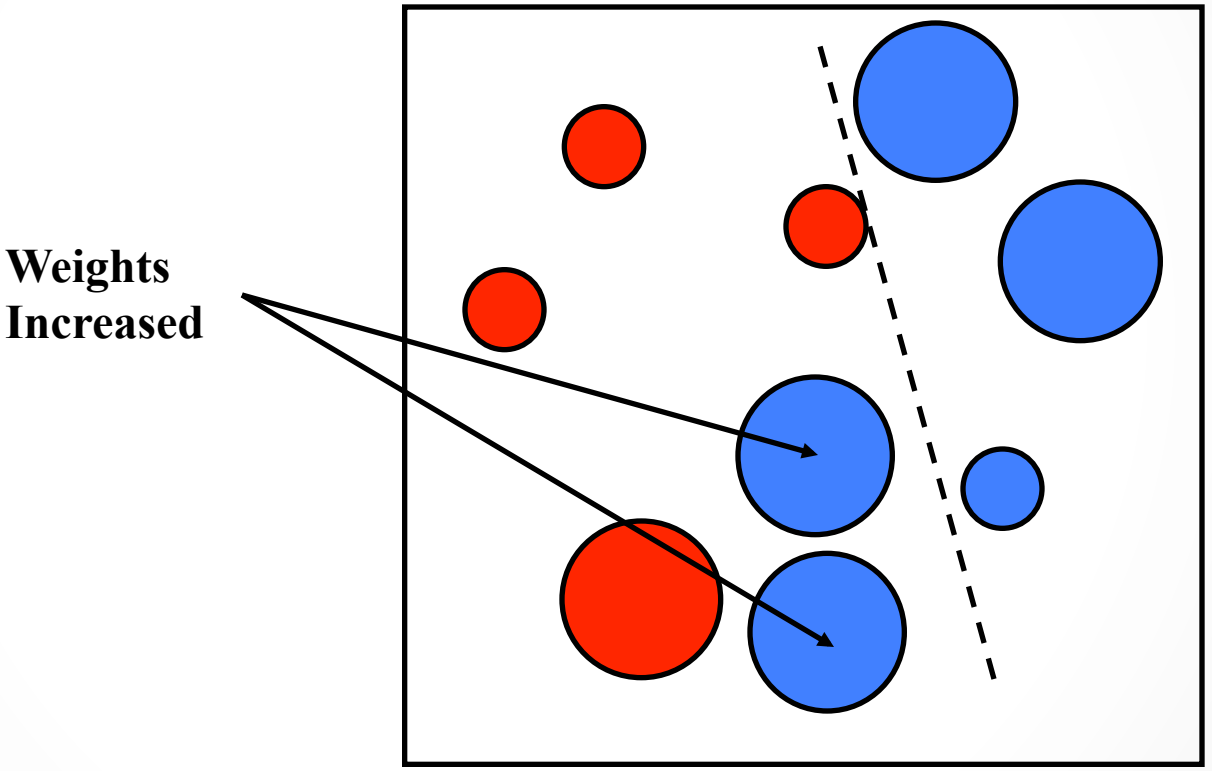
Boosting Illustration



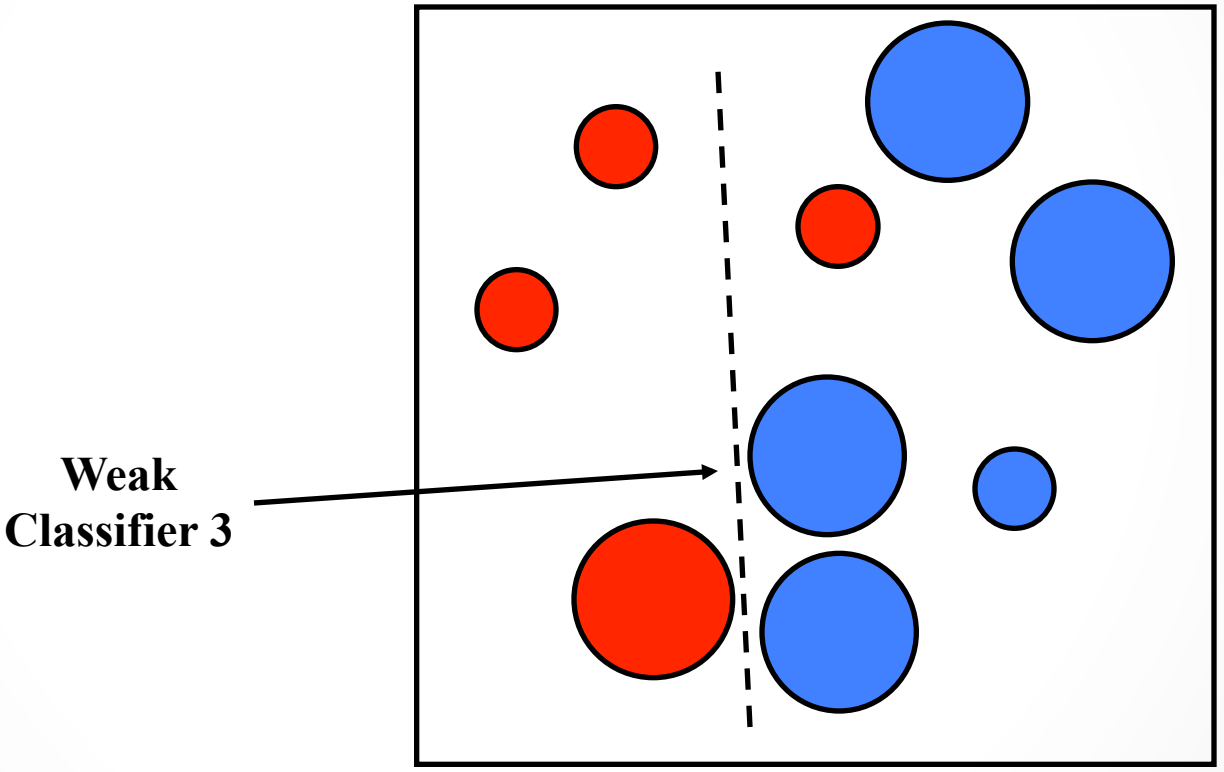
Boosting Illustration



Boosting Illustration

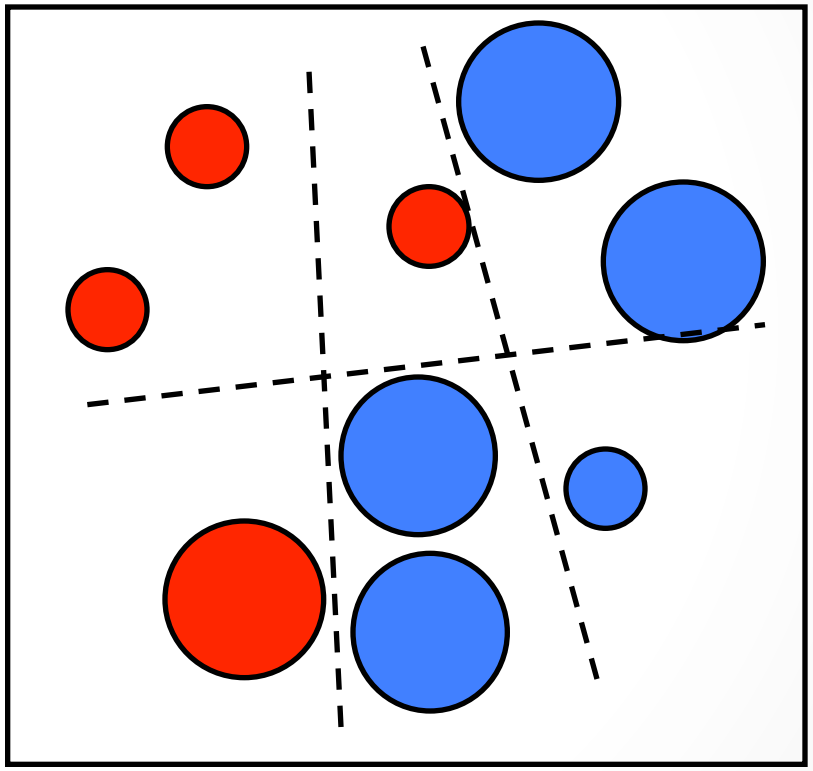


Boosting Illustration



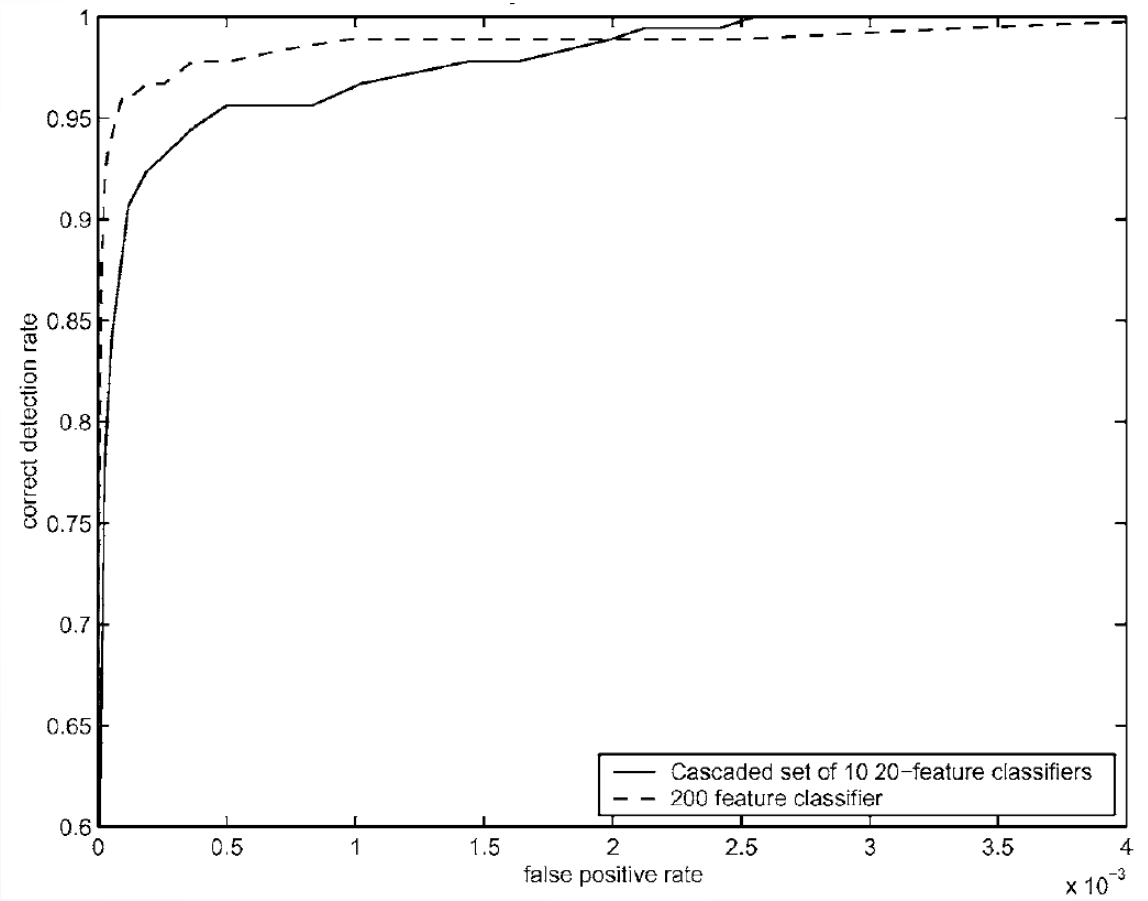
Boosting Illustration

Final classifier is a combination of weak classifiers

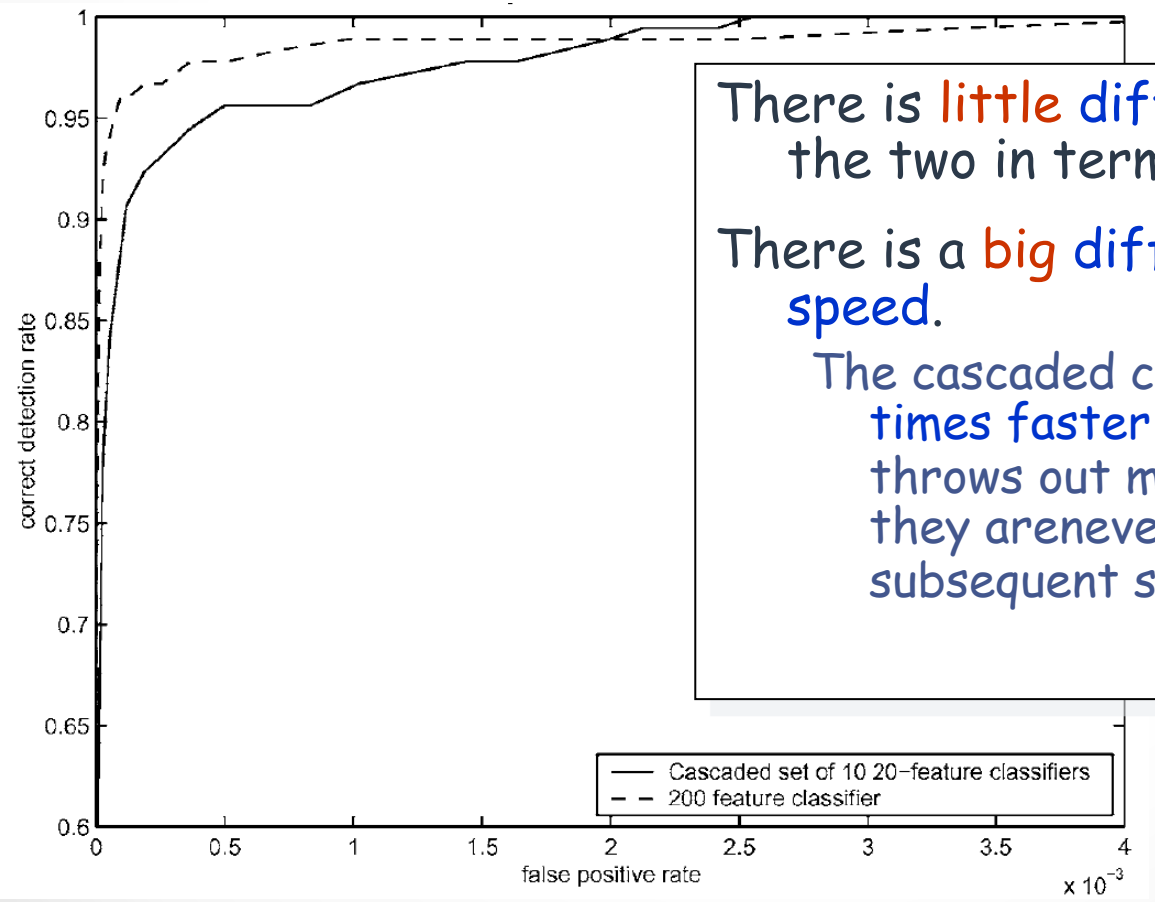


ROC Curves Cascaded Classifier to Monolithic Classifier

Speed of cascade classifier is 10 times faster



ROC Curves Cascaded Classifier to Monolithic Classifier



There is little difference between the two in terms of accuracy.

There is a big difference in terms of speed.

The cascaded classifier is nearly 10 times faster since its first stage throws out most non-faces so that they arenever evaluated by subsequent stages.

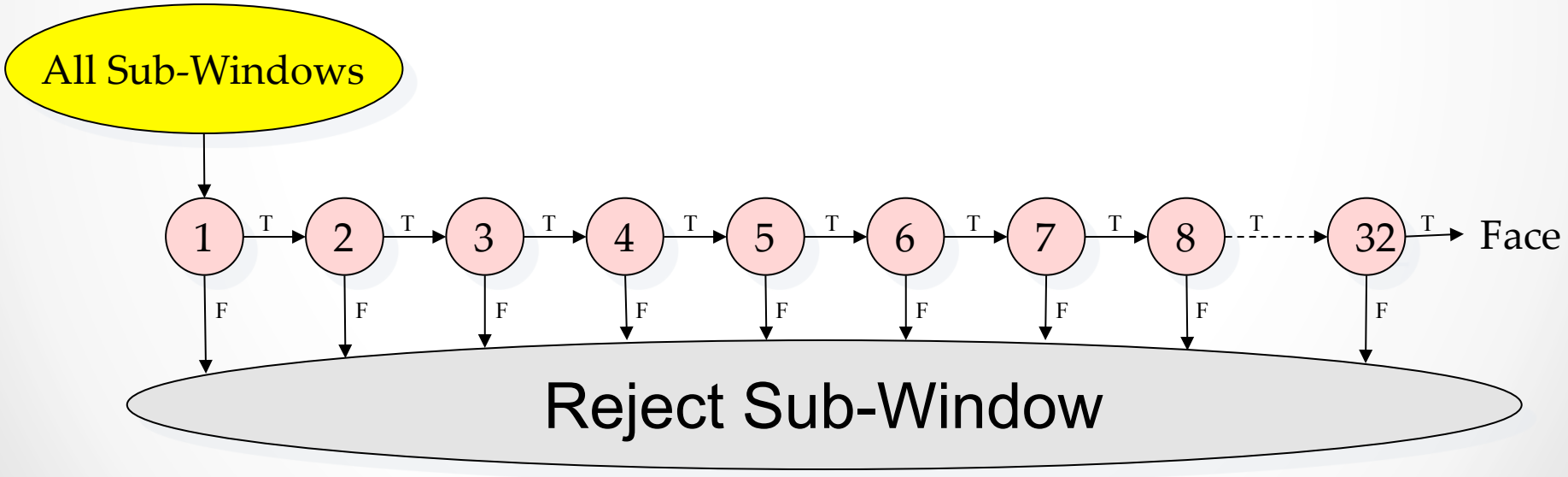
Face Detection System

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

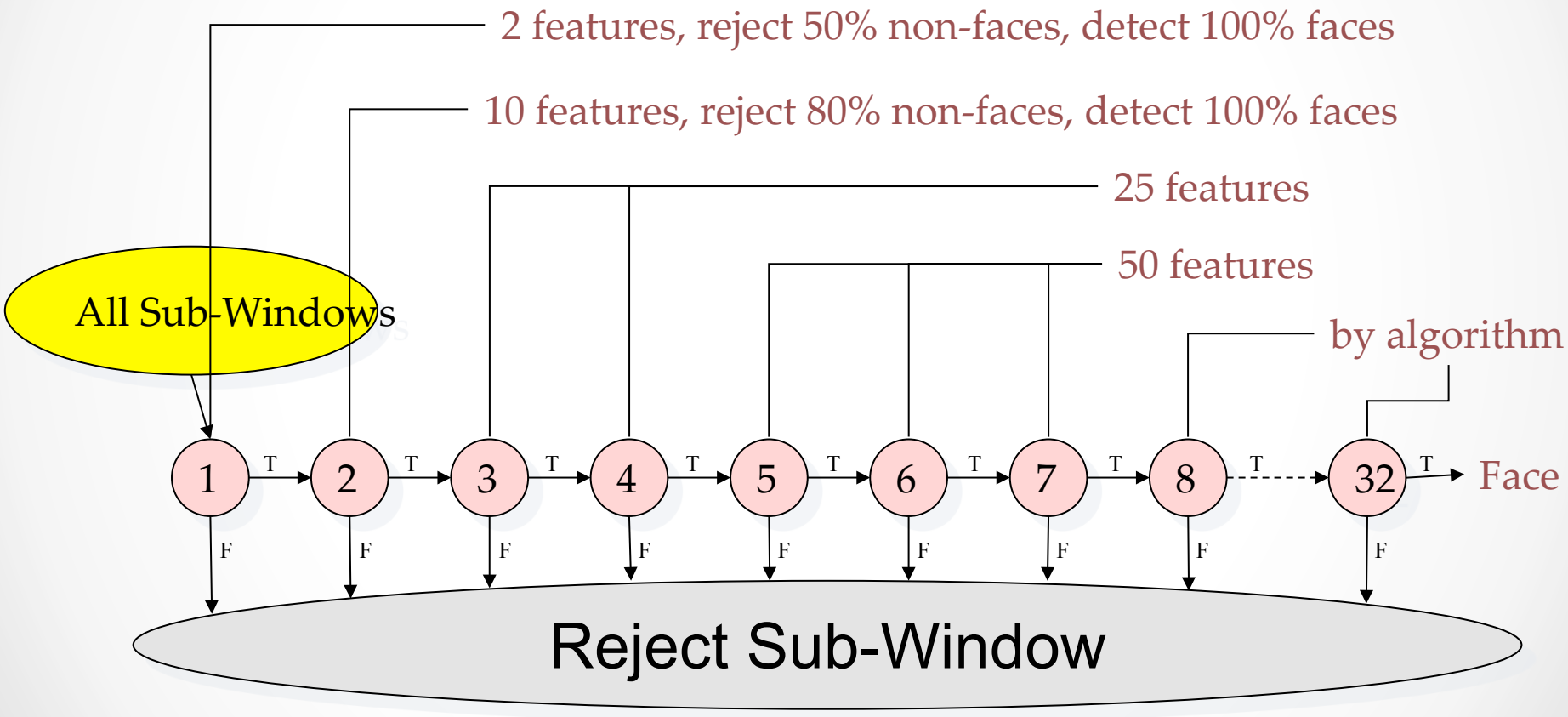
Structure of the Detector Cascade

Increasingly complex classifiers in cascade

- 32 stages
- included a total of 4297 features



Structure of the Detector Cascade



Feature Selection

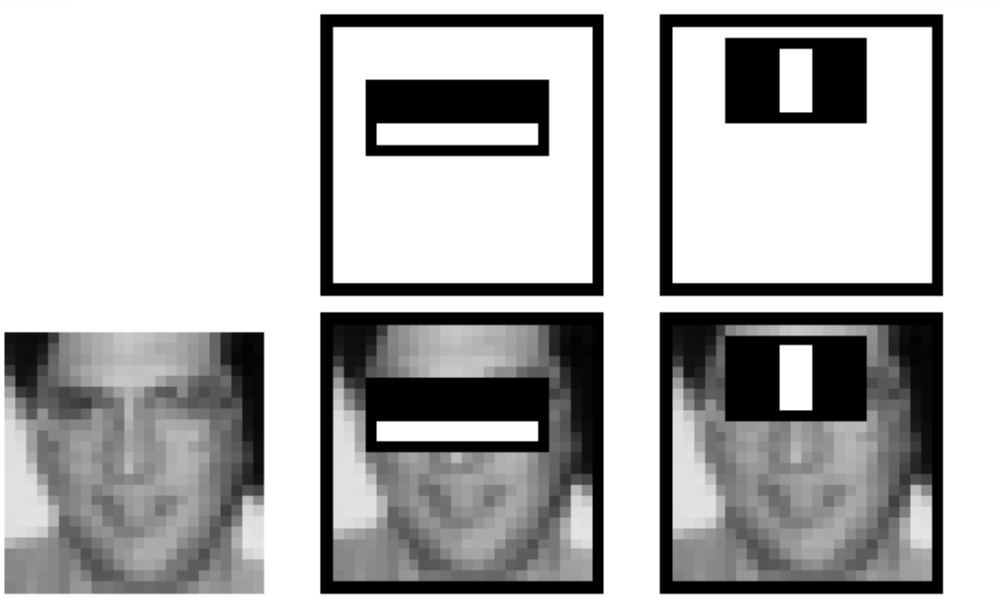


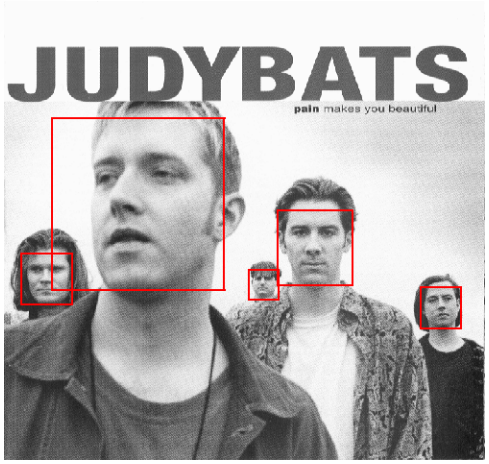
Figure 5: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

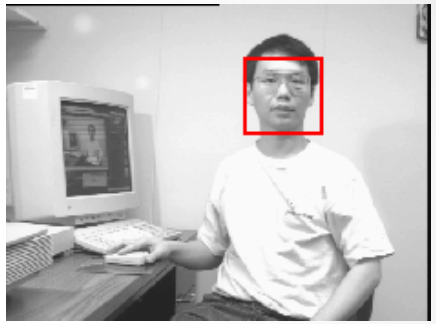
Speed of the Final Detector

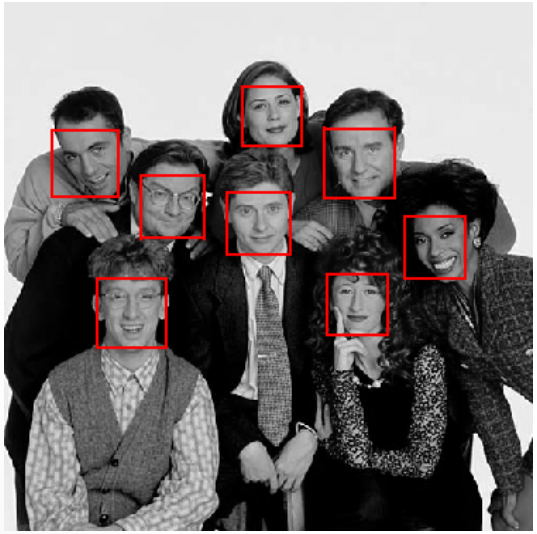
- On a 700 Mhz Pentium III processor, the face detector can process a 384 ×288 pixel image in about ".067 seconds"
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

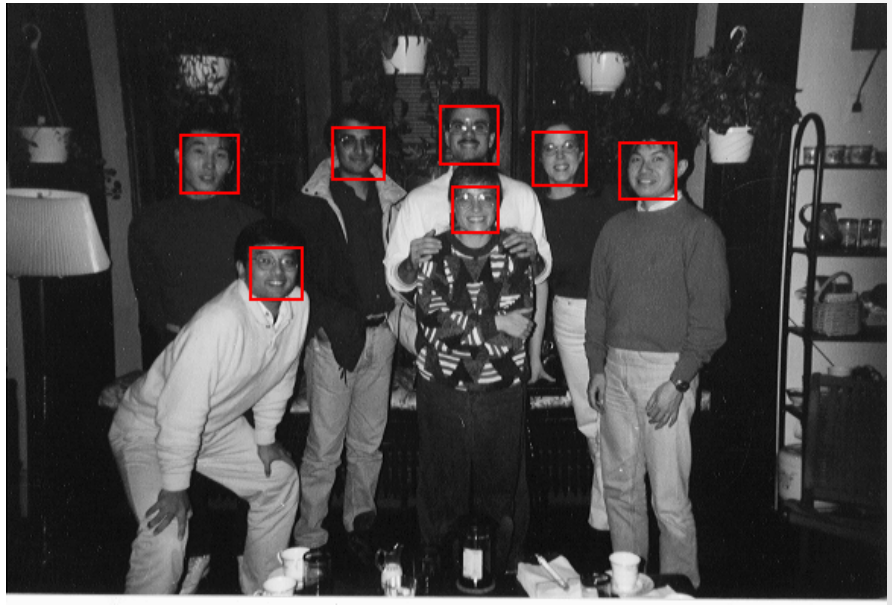
- Average of 8 features evaluated per window on test set

Output of Face Detector on Test Images





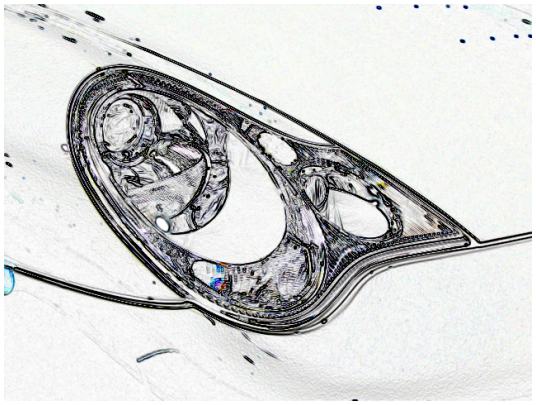




CSE578: Computer Vision

Spring 2016:

Adaboost and Face Detection

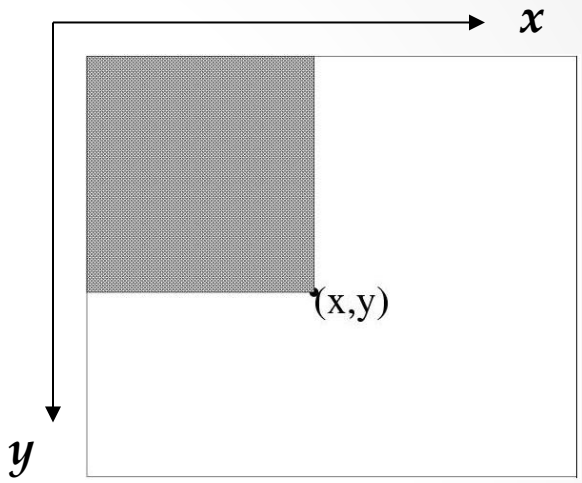

Anoop M. Namboodiri

Center for Visual Information Technology

IIIT Hyderabad, INDIA

Integral Image Representation

- Given a detection resolution of 24x24 (smallest sub-window), the set of different rectangle features is ~160,000 !
- Need for speed
- Introducing Integral Image Representation
  - Definition: The integral image at location (x,y), is the sum of the pixels above and to the left of (x,y), inclusive
- The Integral image can be computed in a single pass and only once for each sub-window!



formal definition:

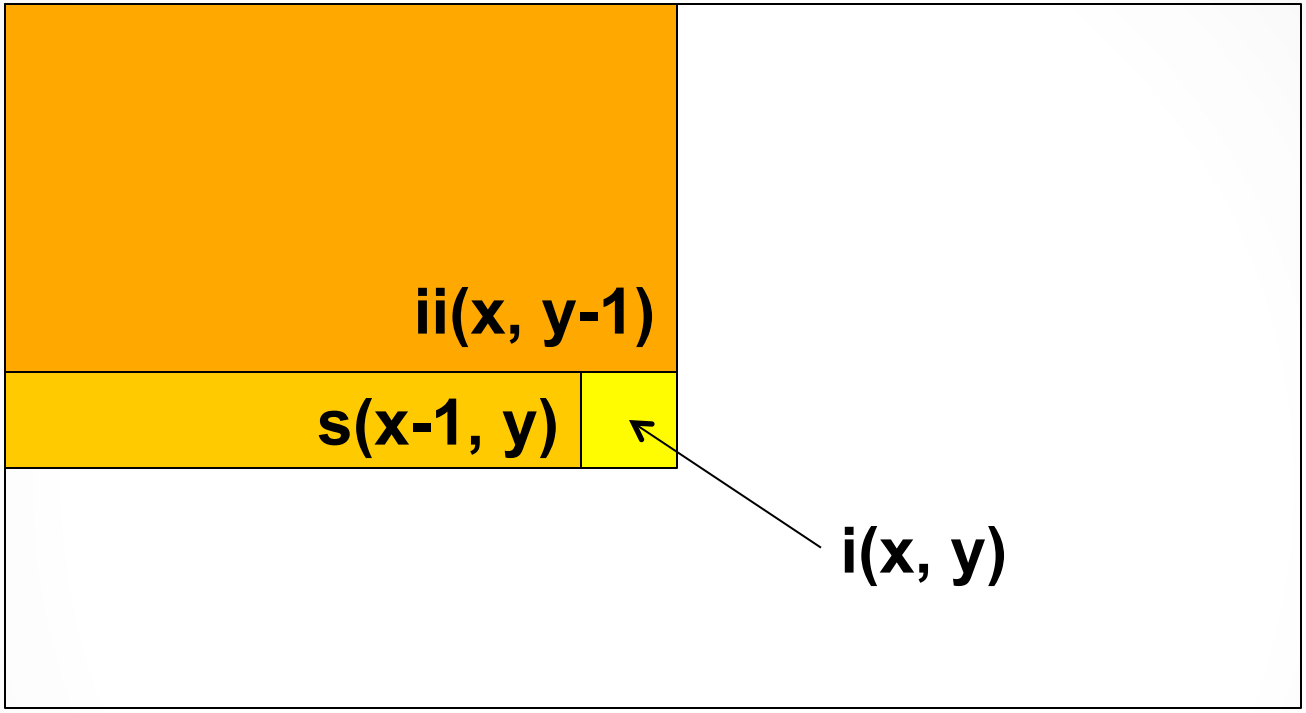
$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y')$$

Recursive definition:

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

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

Computing the Integral Image



- Cumulative row sum: $s(x, y) = s(x–1, y) + i(x, y)$
- Integral image: $ii(x, y) = ii(x, y−1) + s(x, y)$

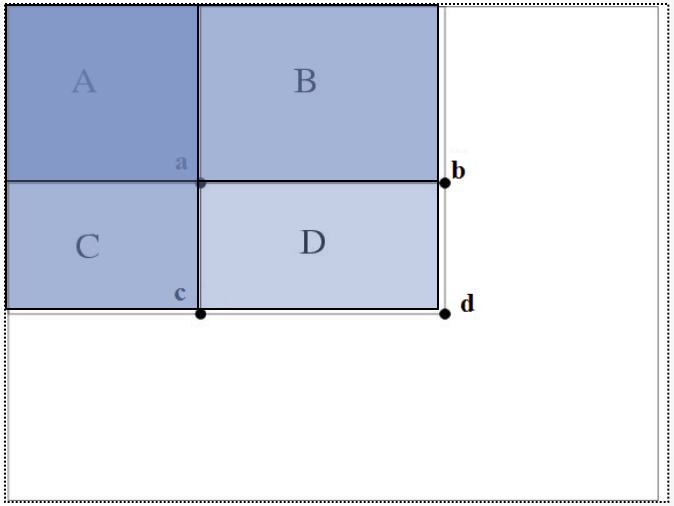
MATLAB: ii = cumsum(cumsum(double(i)), 2);

Rapid Computation of Features

- We can now compute any rectangular sum in **constant time**
  - e.g., integral sum inside D is:

    $ii(d) + ii(a) - ii(b) - ii(c)$

- two-, three-, and four-rectangular features can be computed with 6, 8 and 9 **array references** respectively.
- As a result: feature computation takes less time



ii(a) = A

ii(b) = A+B

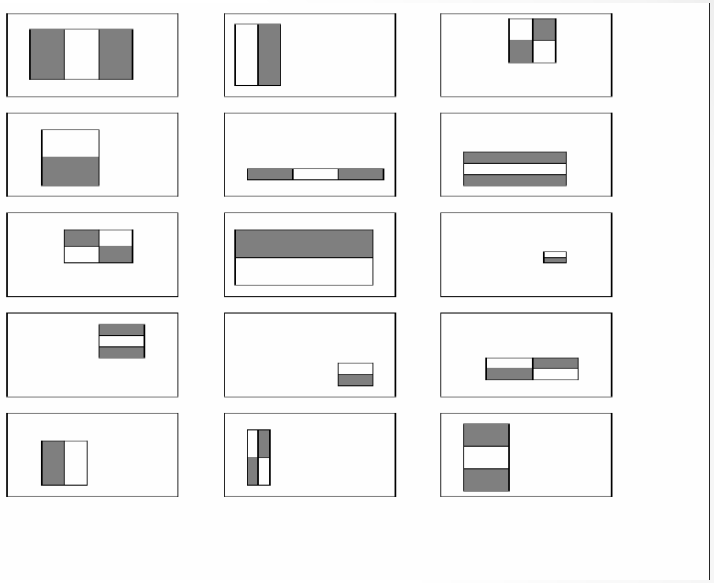
ii(c) = A+C

ii(d) = A+B+C+D

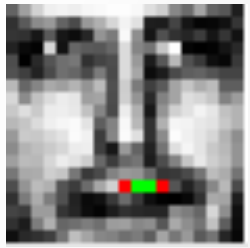
D = ii(d)+ii(a)-
ii(b)-ii(c)

Feature Selection

- Problem: Too many features
  - In a 24x24 sub-window, there are ~160,000 features (all patterns, orientations, locations and scales)
  - impractical to compute all of them (computationally expensive)
- We have to select a subset of relevant (informative) features to model a face
  - **Hypothesis**: "A very small subset of features can be combined to form an effective classifier"
  - How?
    - AdaBoost algorithm



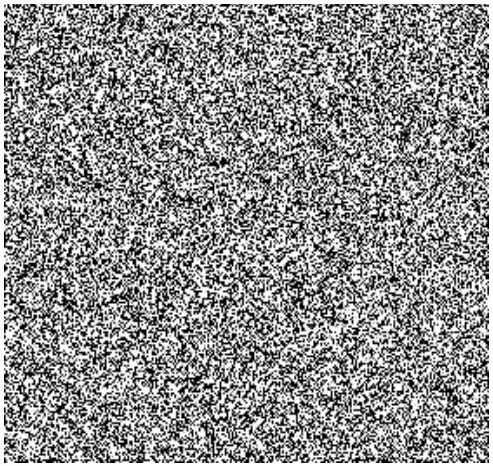
Relevant feature



Irrelevant feature

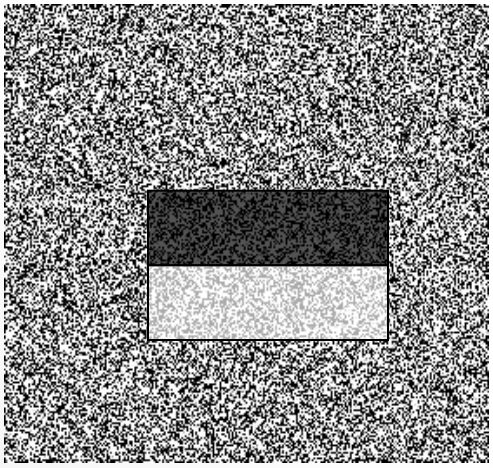
Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Relevant and Irrelevant Images



Input Images

Feature Computation



AdaBoost

- Stands for "**Ada**ptive" **boost**
- Constructs a "strong" classifier as a linear combination of weighted simple "weak" classifiers

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Image

Weight

Weak classifier

AdaBoost: The Algorithm

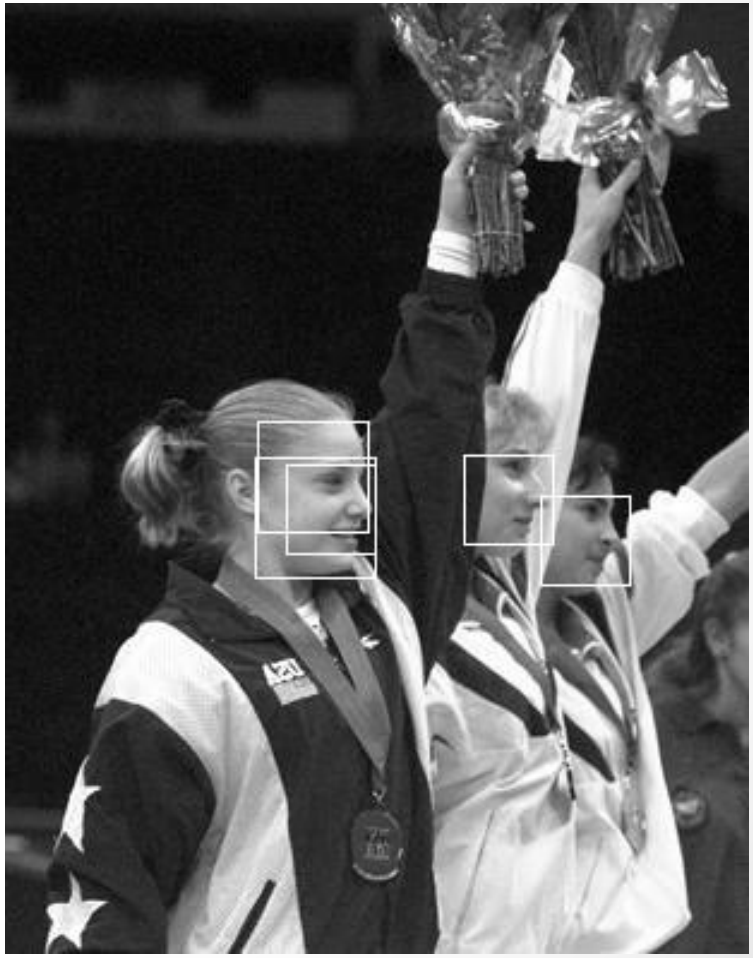
- Given: example images labeled +/-
  - Initially, all weights set equally
- Repeat T times
  - Step 1: choose the most efficient weak classifier (Problem! Remember the huge number of features…)
  - Step 2: Update the weights to emphasize the examples which were incorrectly classified
    - This makes the next weak classifier to focus on "harder" examples
- Final (strong) classifier is a weighted combination of the T "weak" classifiers
  - Weighted according to their accuracy

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

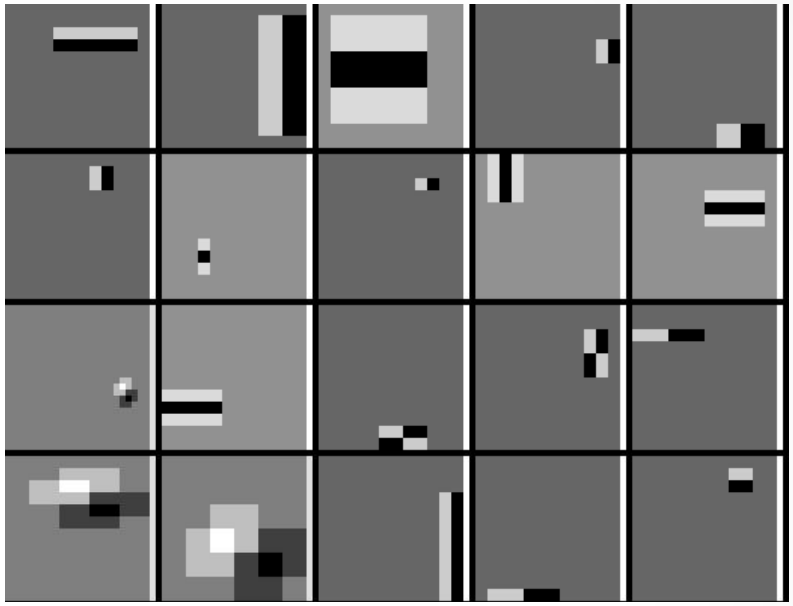
Features of Boosting

- Integrates classifier learning with feature selection
- Training complexity is linear (not quadratic)
- Flexibility in choice of weak learners, boosting scheme
- Testing is fast
- Easy to implement
- Needs many training samples
- May not work as well as SVMs

Profile Detection



Profile Features



Summary: Viola/Jones detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows
- Can be made scale invariant (image pyramids)
- Generic Method: other detectors with same approach
- Cons:
  - Needs special training for non-frontal faces
  - Sensitive to lighting conditions
  - Multiple detections from the same face (overlapping)