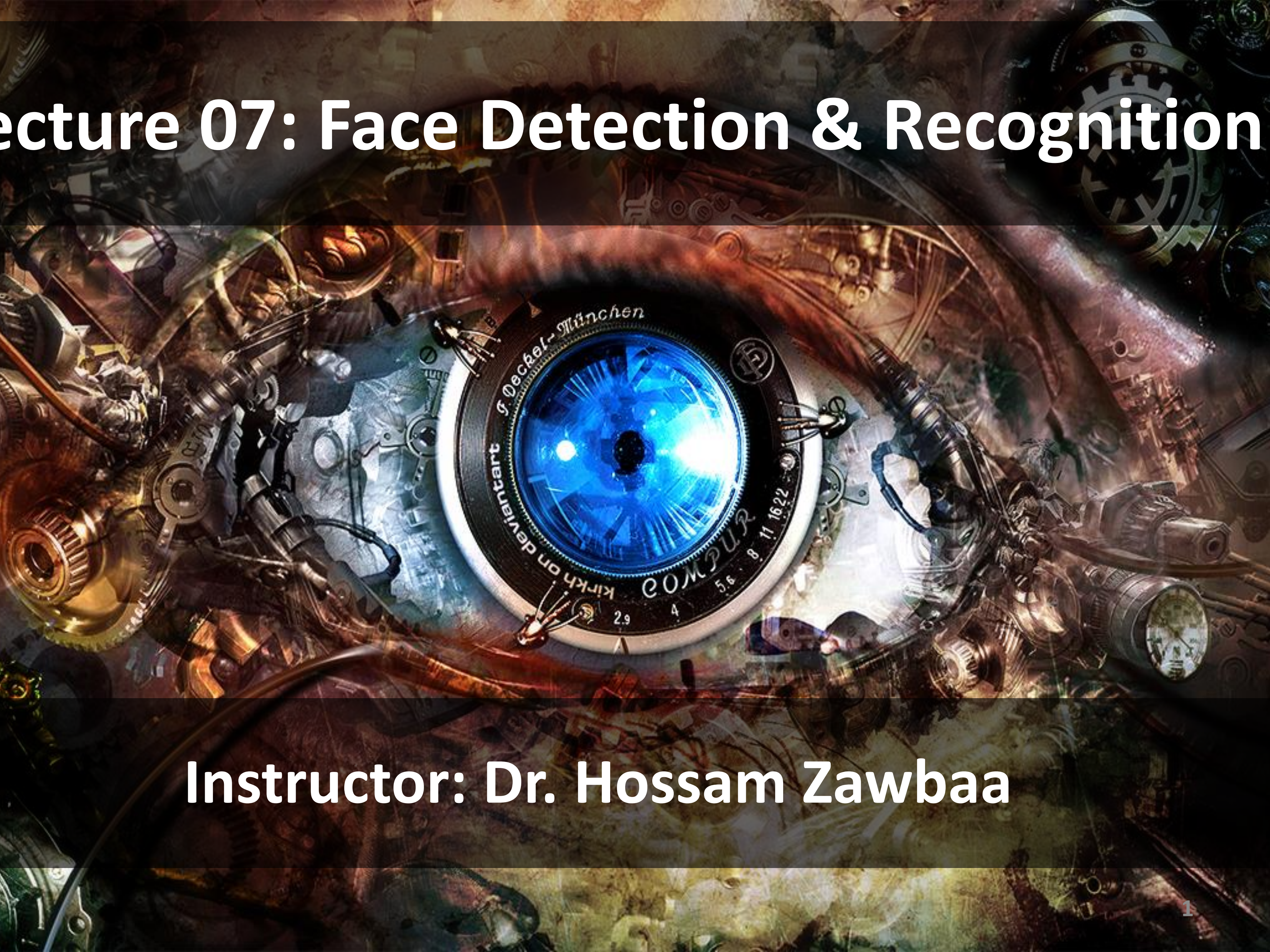
ecture 07: Face Detection & Recognition



Instructor: Dr. Hossam Zawbaa

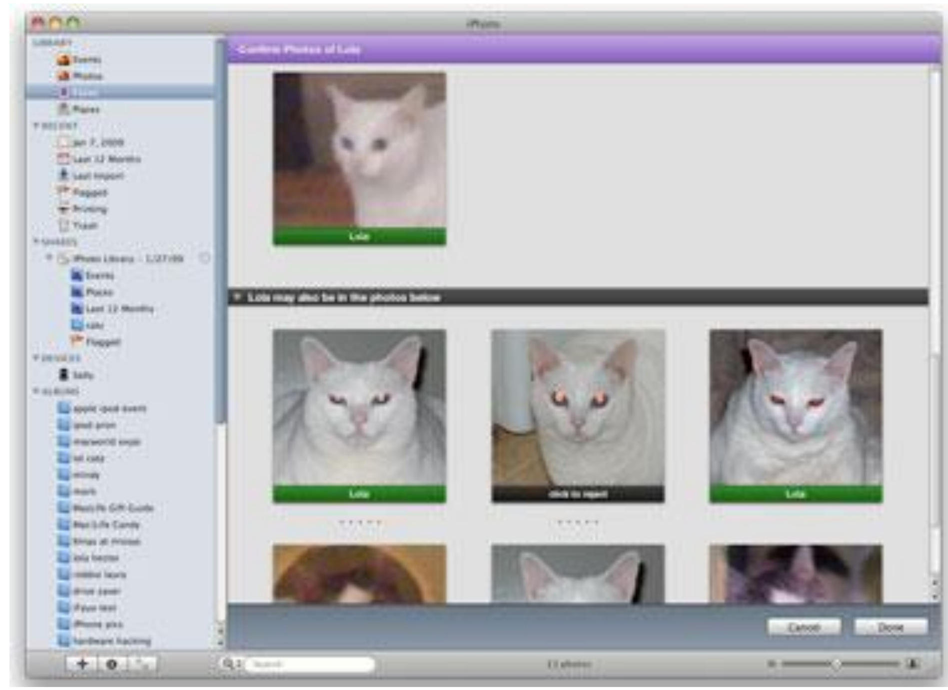
Face detection and recognition

Application: Apple iPhoto

http://www.apple.com/ilife/iphoto/

Application: Apple iPhoto

- Can be trained to recognize cats!



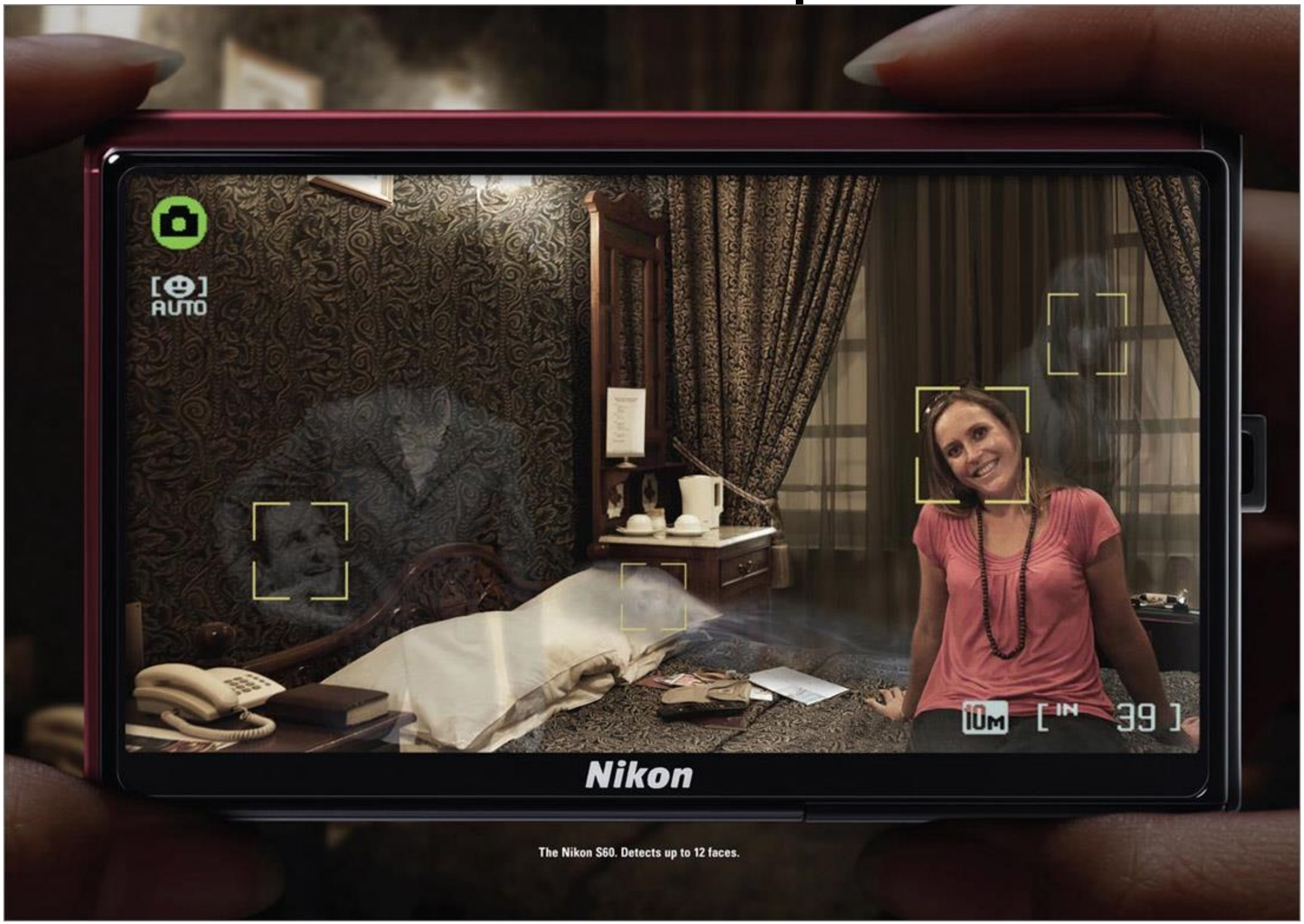
http://www.maclife.com/article/news/iphotos_faces_recognizes_cats

Application: Apple iPhoto

- Things iPhoto thinks are faces

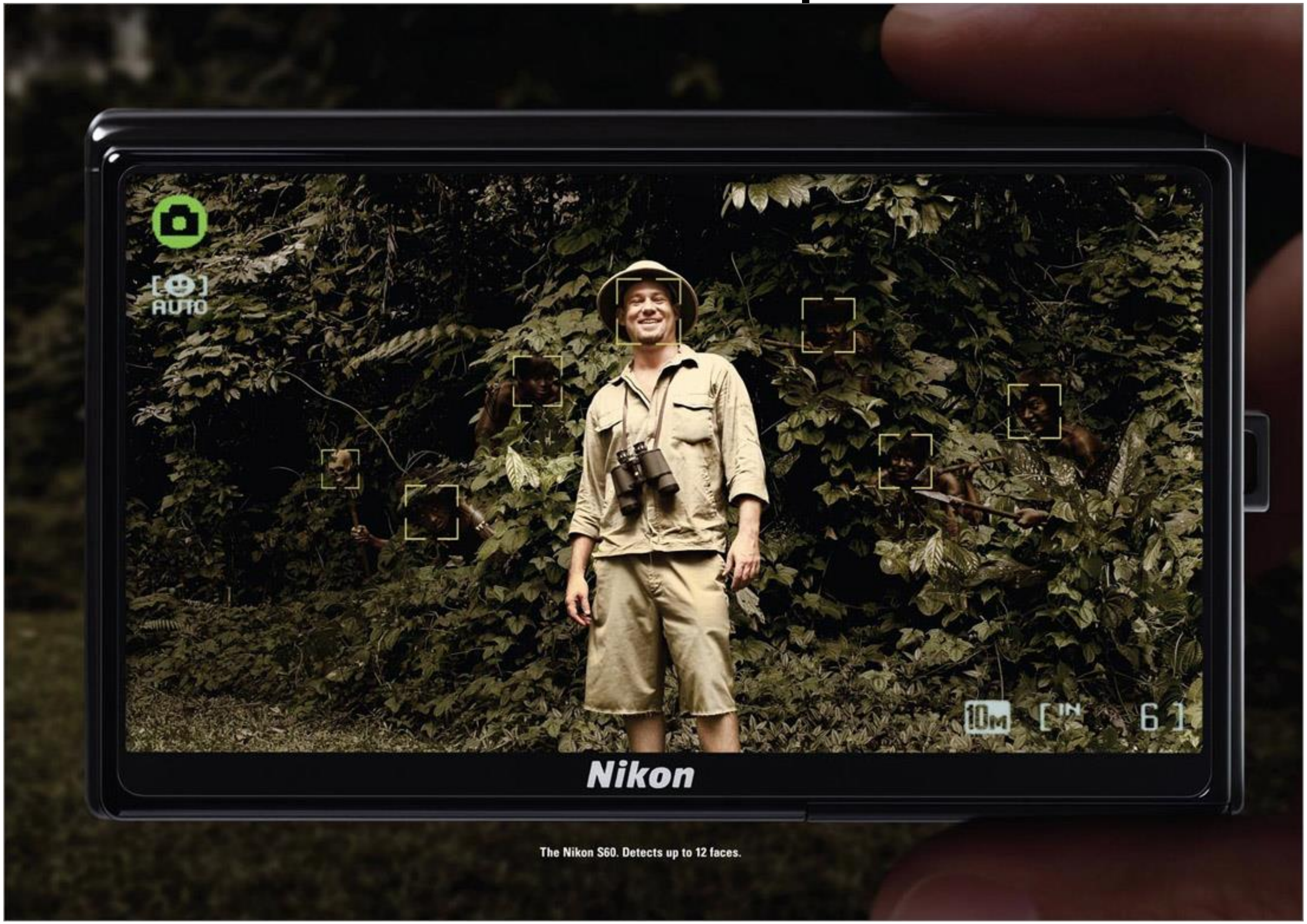
Funny Nikon ads

"The Nikon S60 detects up to 12 faces."



Funny Nikon ads

"The Nikon S60 detects up to 12 faces."



Challenges of face detection

- **Sliding window detector** must *evaluate tens of thousands of location/scale combinations*
- **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*
  - A **megapixel image has ~10^6 pixels** and a **comparable number of candidate face locations**

The Viola/Jones Face Detector

- An approach to real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  1. **Integral images** for fast feature evaluation
  2. **Boosting** for feature selection
  3. **Attentional cascade** for fast rejection of non-face windows

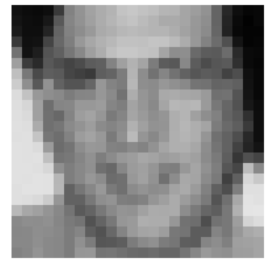
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

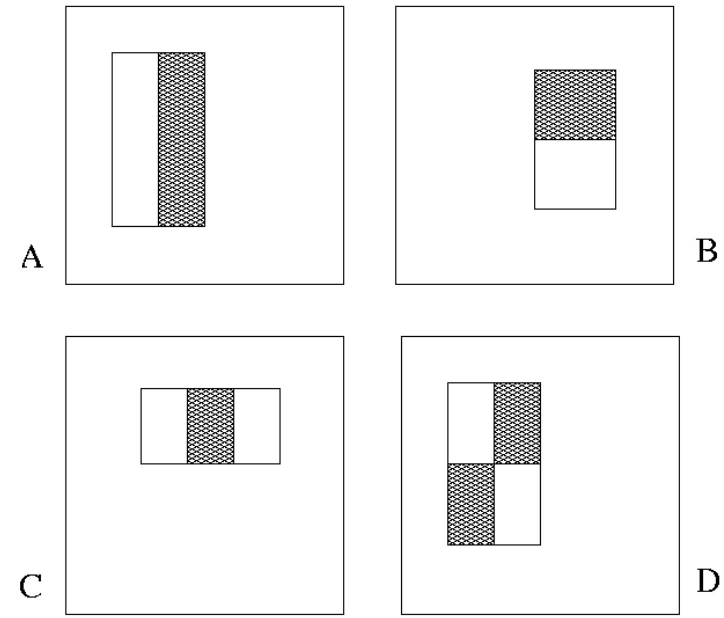
P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

~8000 citations!

Image Features

"Rectangle filters"

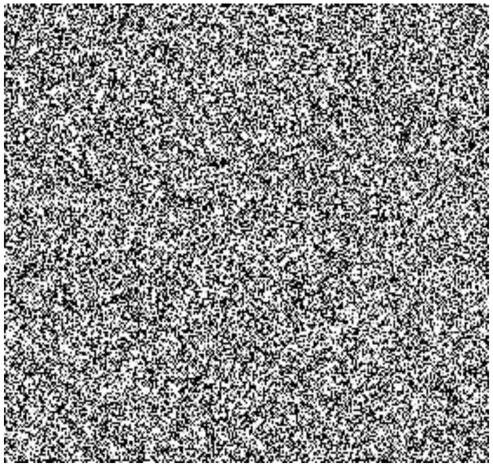




Value =

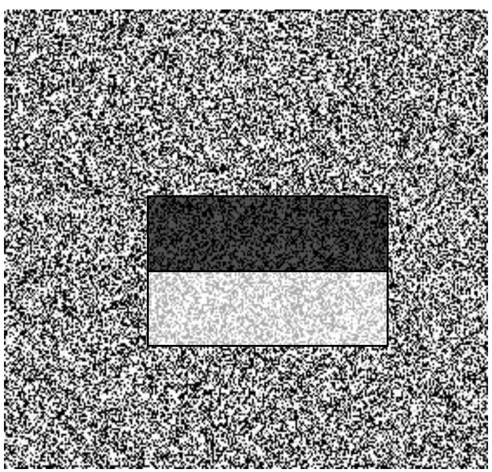
∑ (pixels in white area) –
∑ (pixels in black area)

Example



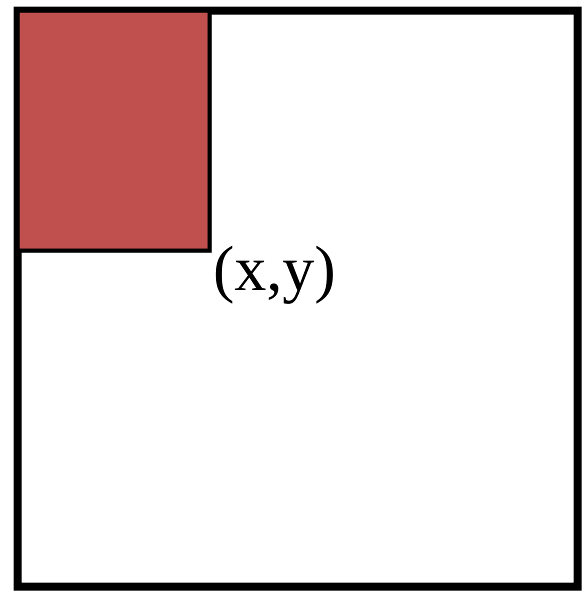
Source

Result

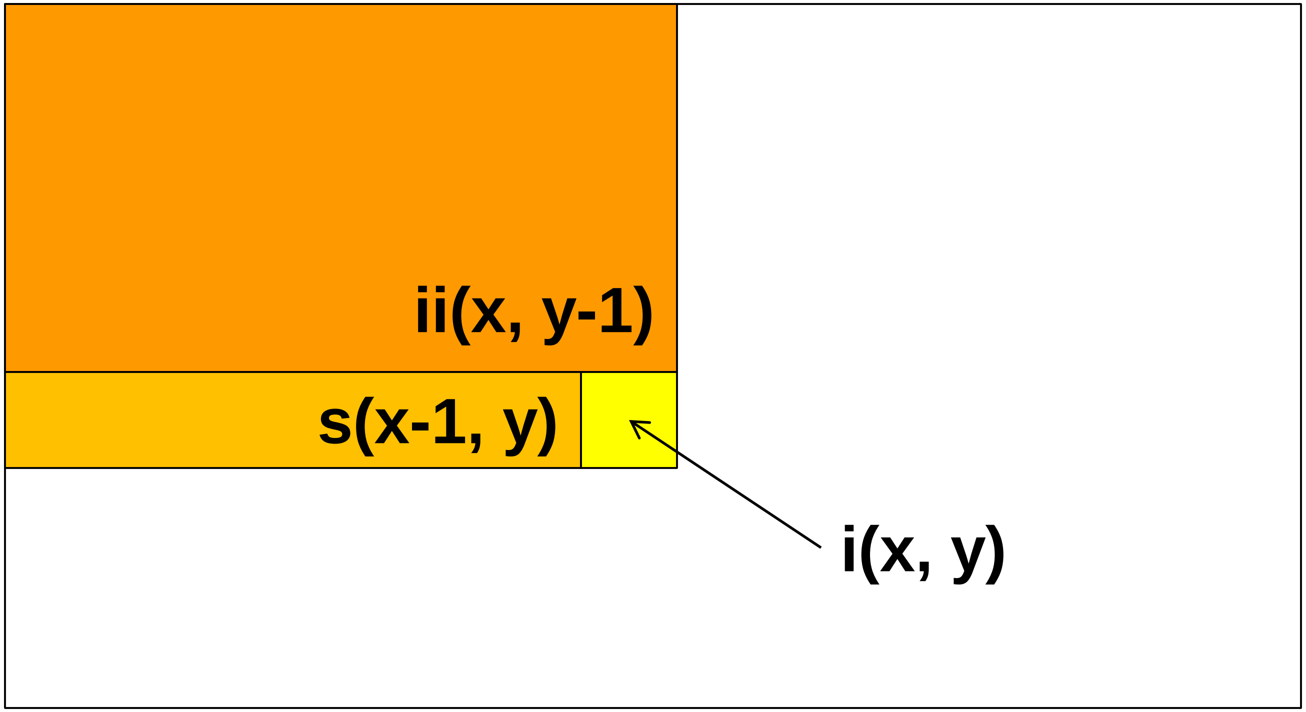


Fast computation with integral images

- 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 the integral image



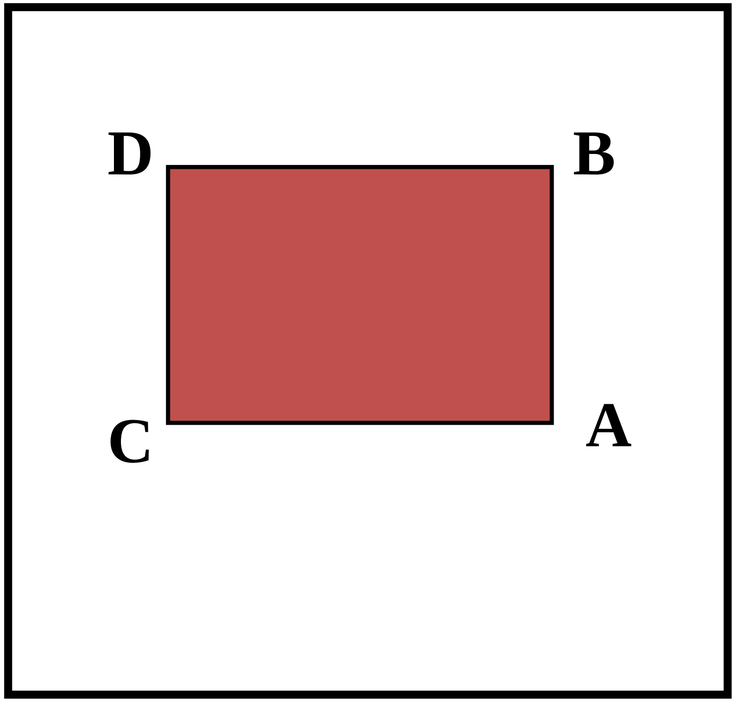
- Cumulative row sum: $s(x, y) = s(x-1, y) + i(x, y)$
- Integral image: $ii(x, y) = ii(x, y-1) + s(x, y)$

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

Computing sum within a rectangle

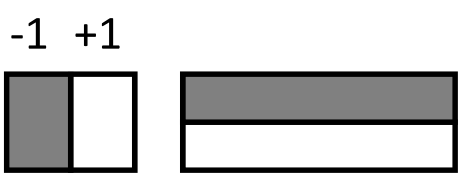
- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then, **the sum of original image values** within the rectangle can be computed as:

  sum = A – B – C + D

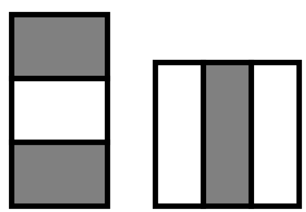


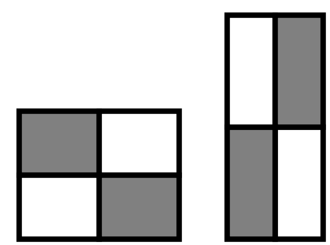
Features that are fast to compute

- "**Haar-like features**"
  - Differences of sums of intensity
  - **Thousands, computed at various positions and scales** within detection window

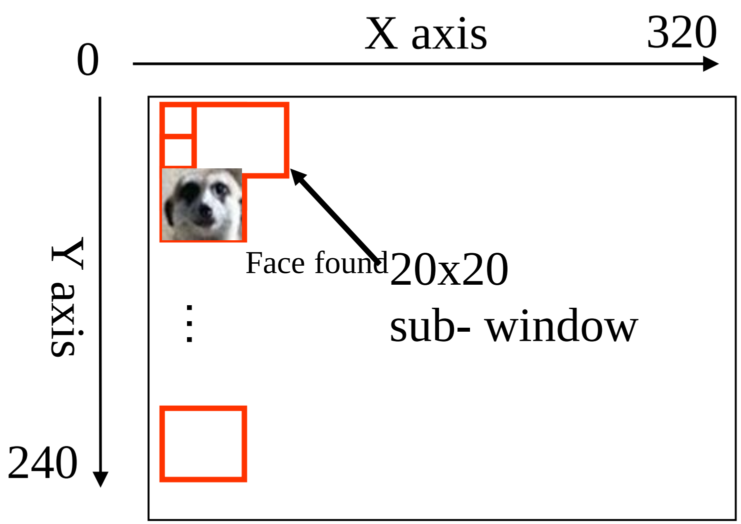


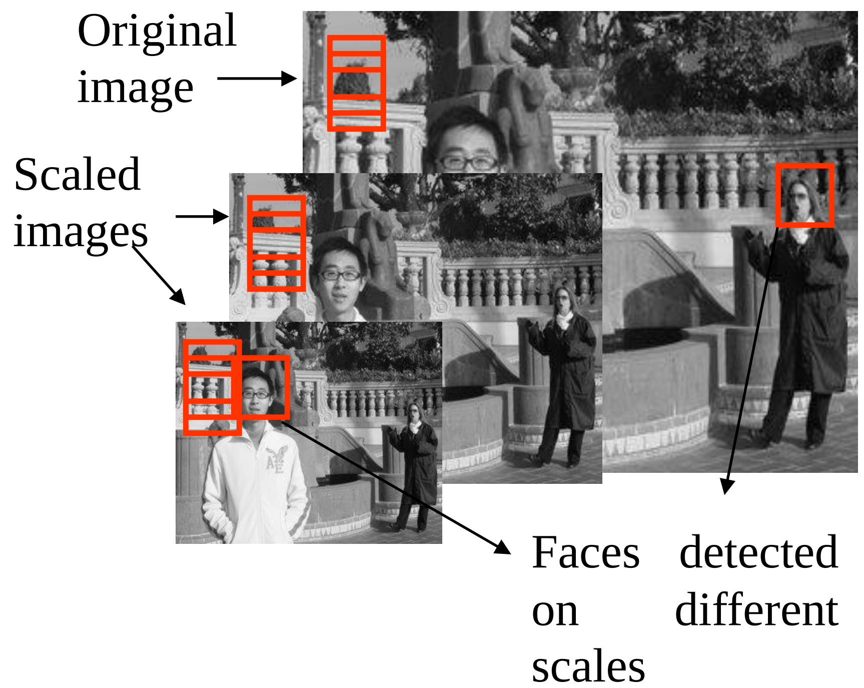
Two-rectangle features


Three-rectangle features


Etc.

Haar-Feature based object detection algorithm



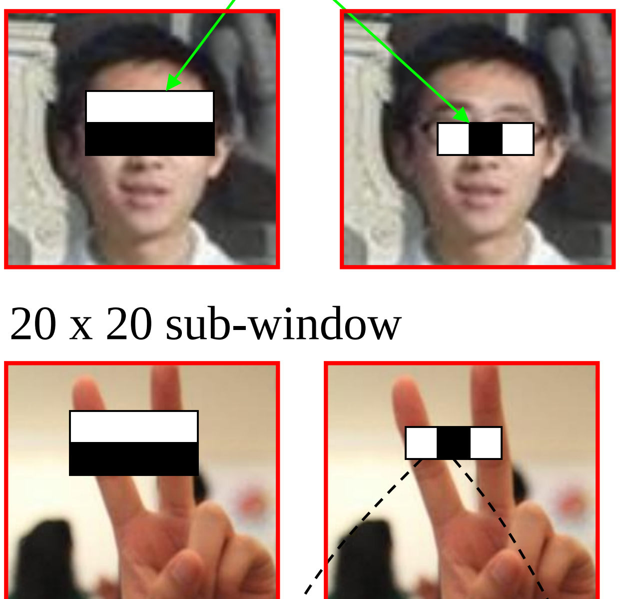


Movement of sub-window

(320 – 20) * (240 – 20) = **66,000** sub-windows

Face detection in sub-window

Facial Haar ***features***



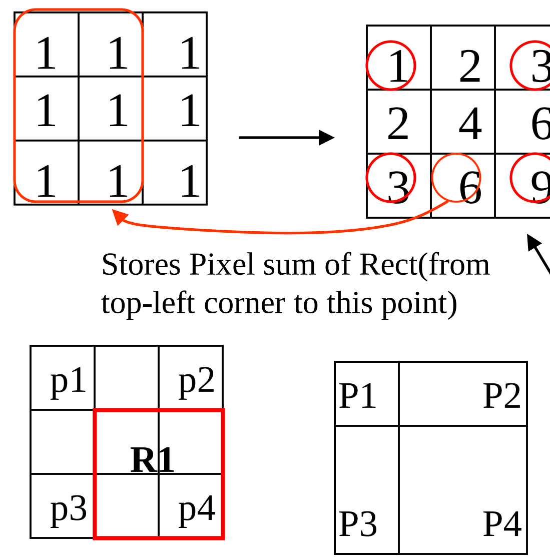
20 x 20 sub-window

Pass

Fail

Original image

Integral Image



Stores Pixel sum of Rect(from top-left corner to this point)

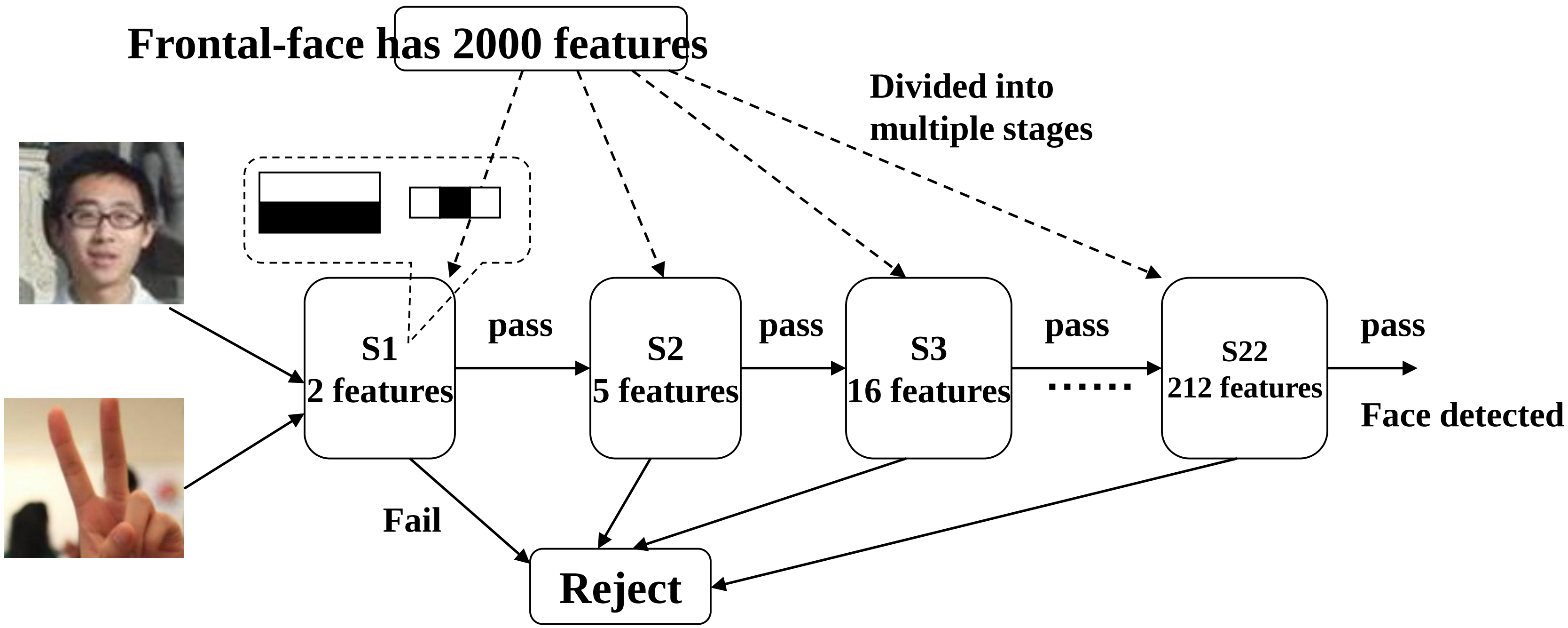
Need 4 corner values

Pixel_Sum(R1) =
P4 - P2 - P3 + P1 = 4

Calculate Haar-feature value:

Pixel_Sum(Rect_W) – Pixel_Sum(Rect_B)

Cascade decision process



Fail any stage will reject current sub-window

- Fast classifiers early in cascade
- Slow classifiers later, but most examples don’t get there

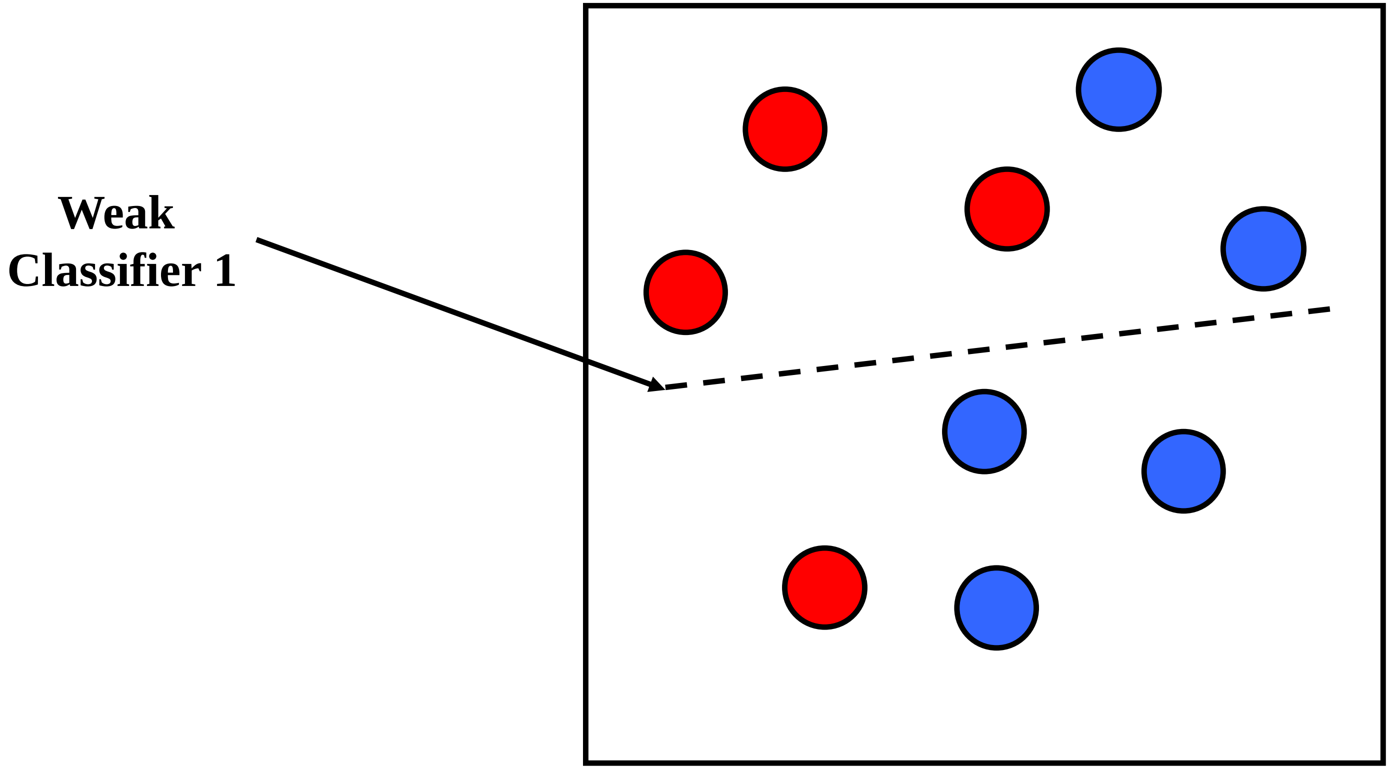
Training procedure

- Initially, weight each training example equally
- In each boosting round:
  - **Find the weak learner** that achieves the lowest *weighted* training error
  - **Increase 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 directly proportional to its accuracy).

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

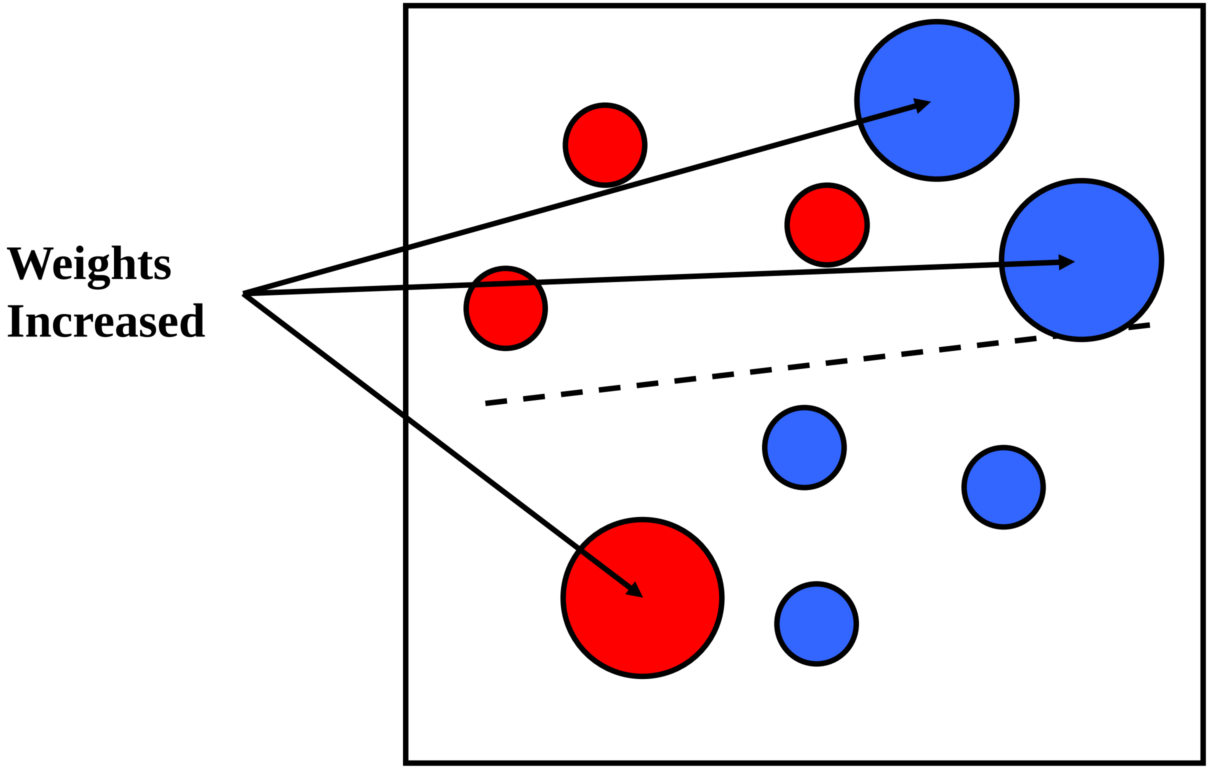
Boosting intuition

- *Boosting* is a classification scheme that combines *weak learners* into a more accurate *ensemble classifier*

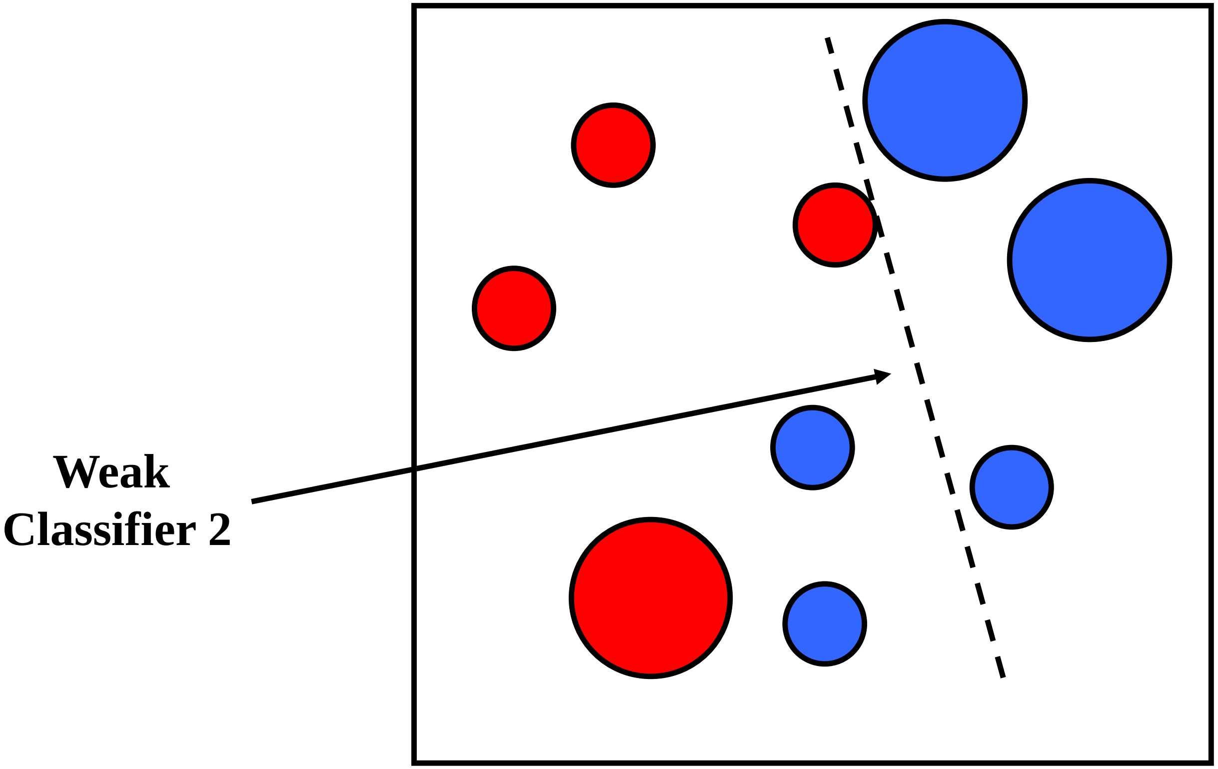


Slide credit: Paul Viola

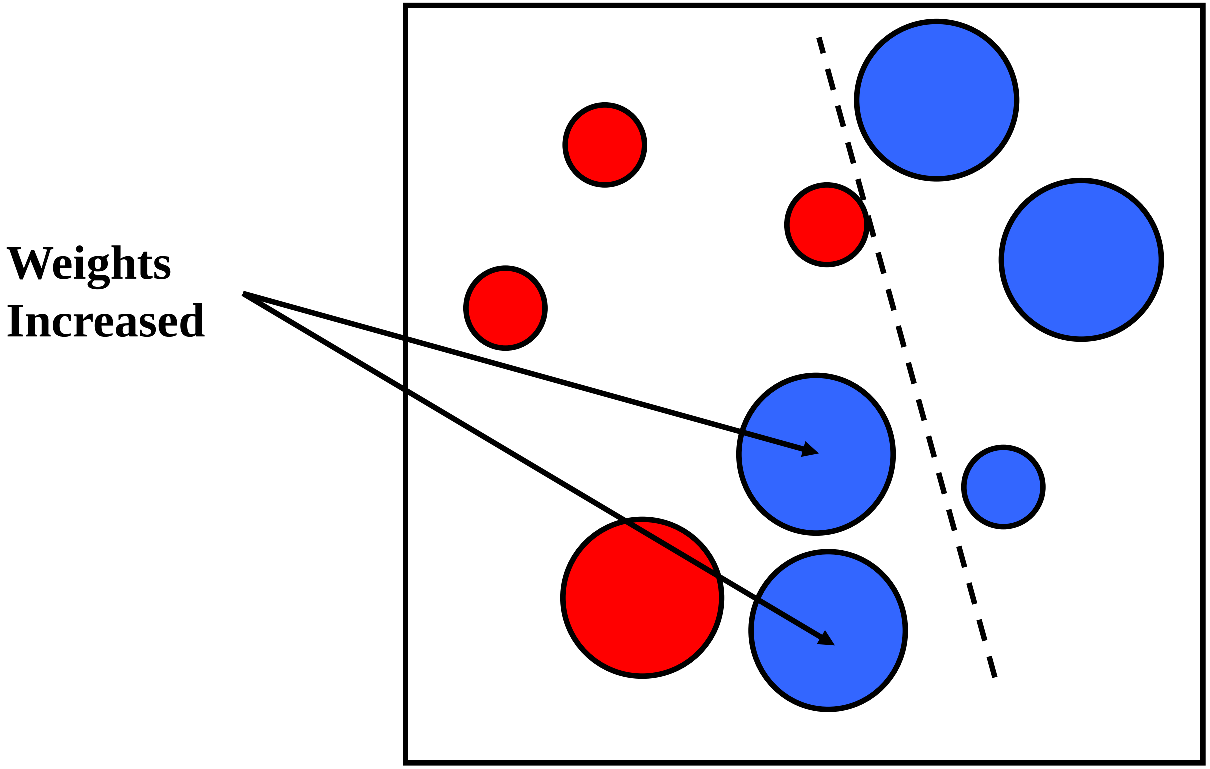
Boosting illustration



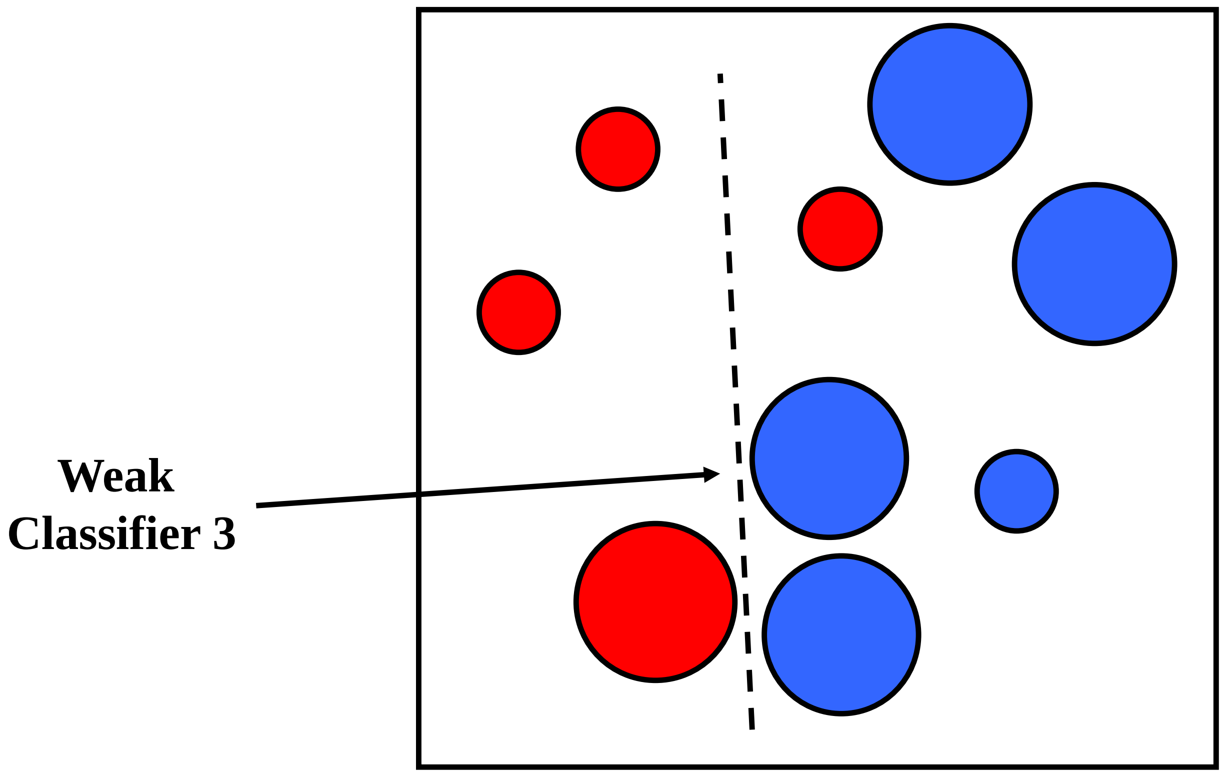
Boosting illustration



Boosting illustration

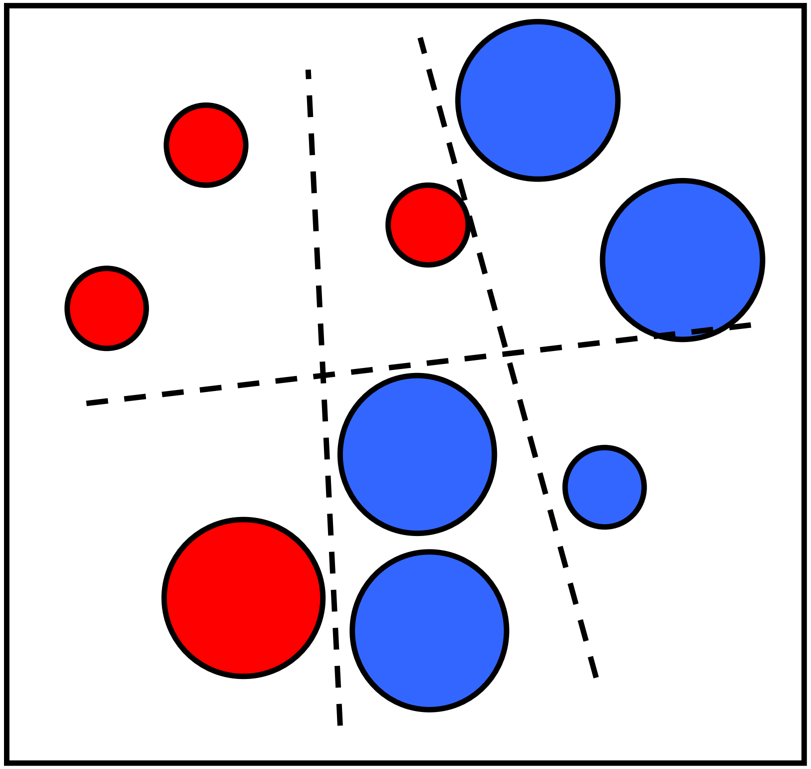


Boosting illustration



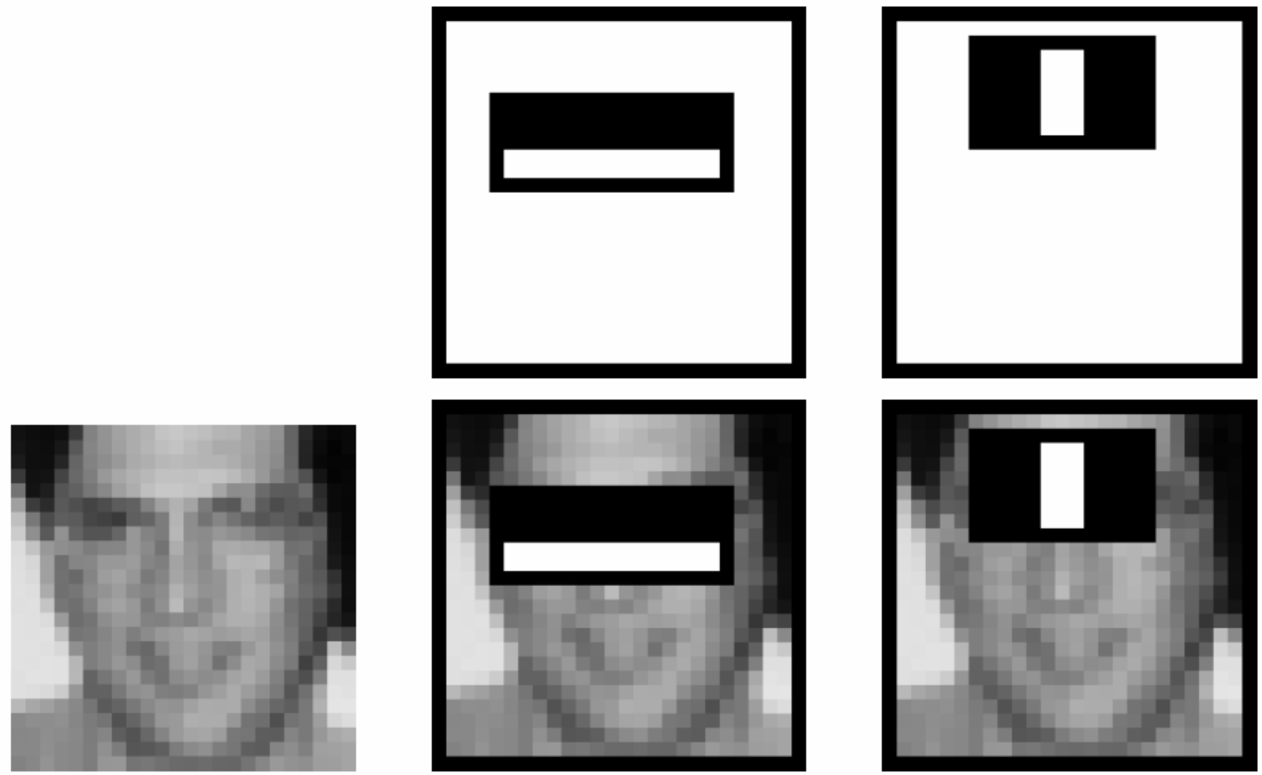
Boosting illustration

Final classifier is a combination of weak classifiers



Boosting for face detection

- First two features selected by boosting:

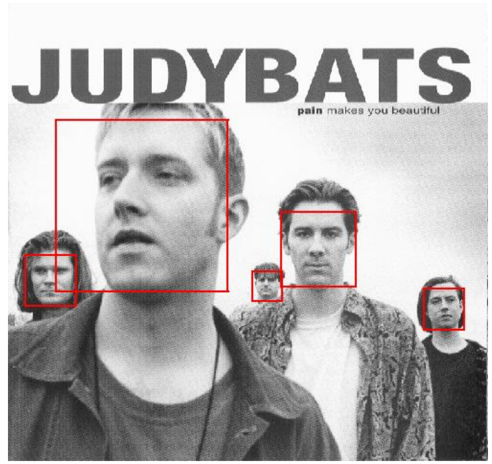


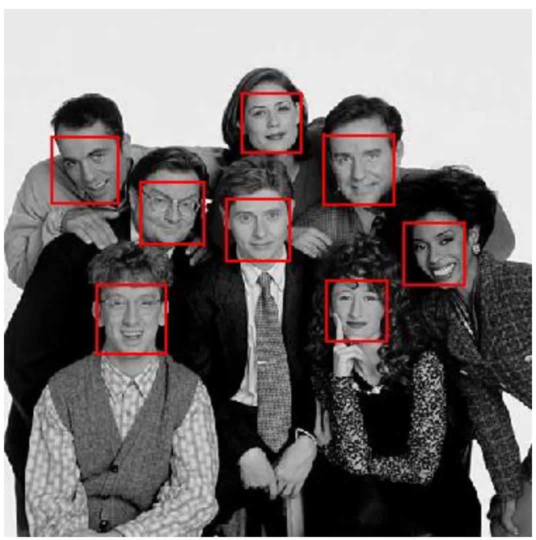
This feature combination can yield 100% detection rate and 50% false positive rate

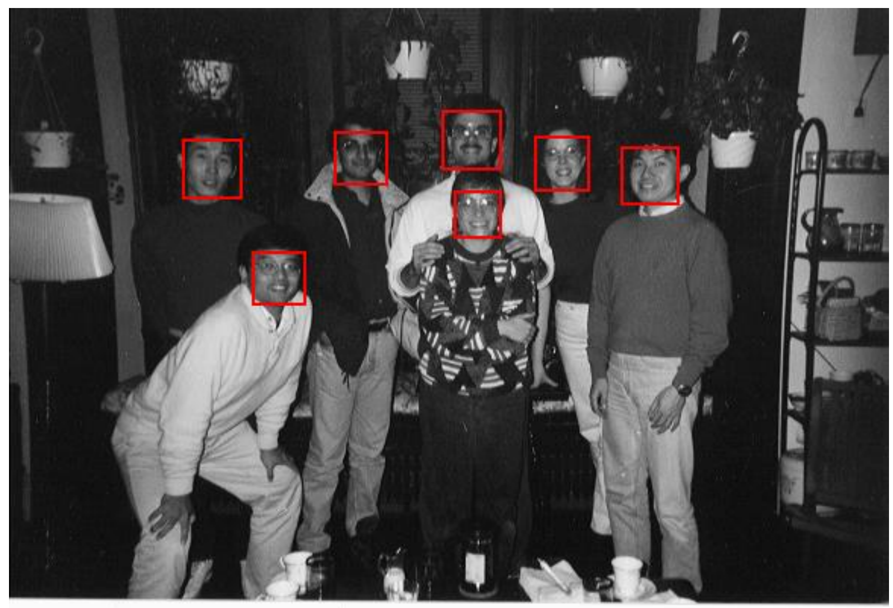
Boosting vs. SVM (ANN)

- Advantages of boosting
  - Integrates classifier training with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast
  - Easy to implement
- Disadvantages
  - Needs many training examples
  - Training is slow
  - Often doesn’t work as well as SVM or ANN (especially for many-class problems)

Output of Face Detector

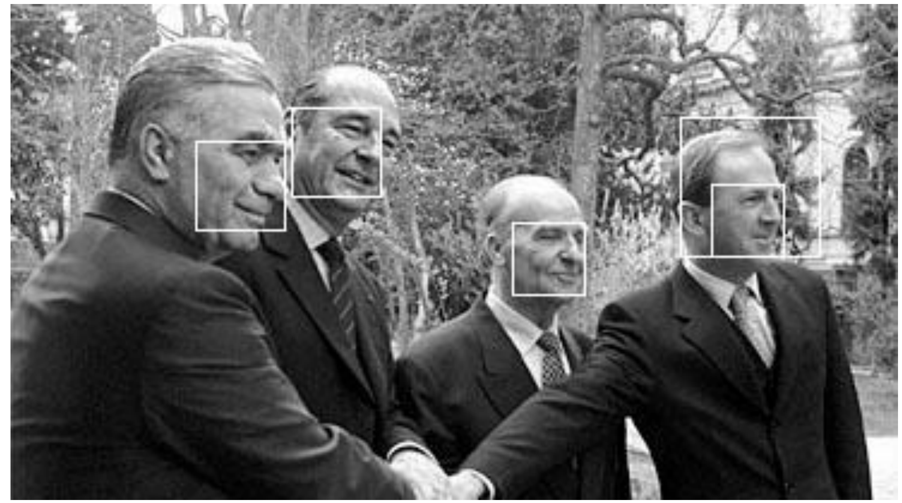






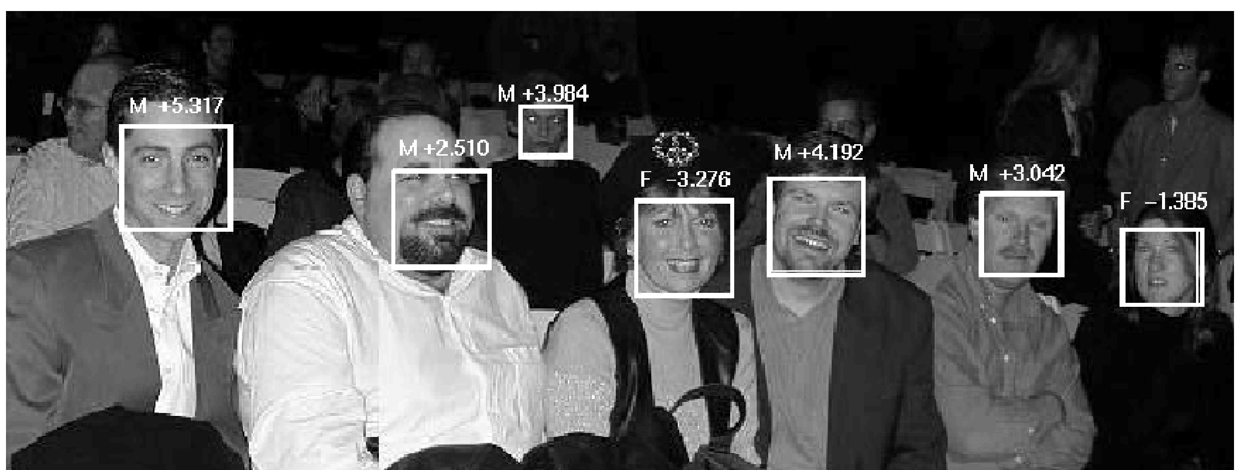
Other detection tasks

Facial Feature Localization

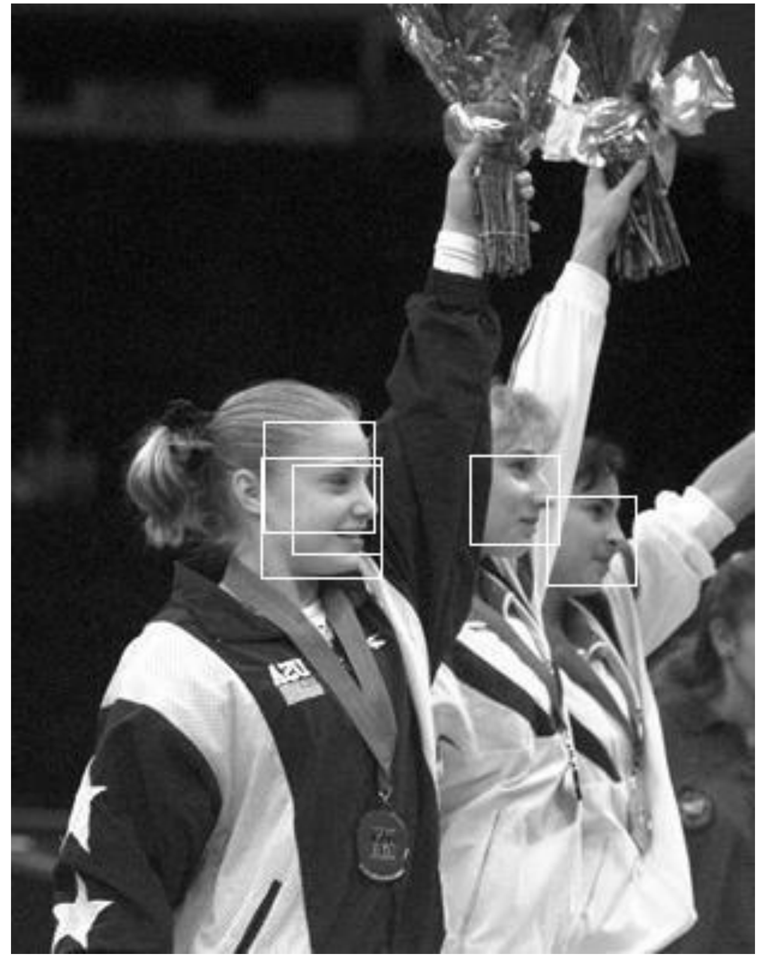


Profile Detection

Male vs. female



Profile Detection



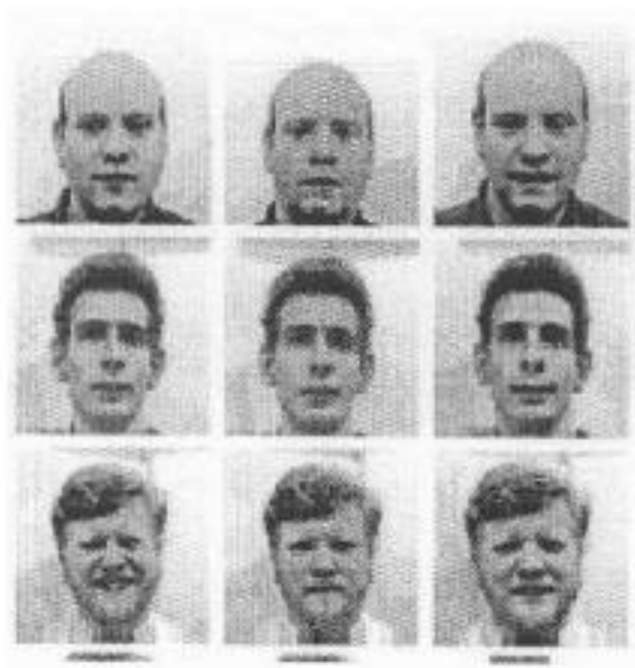
Face Detection and Tracking

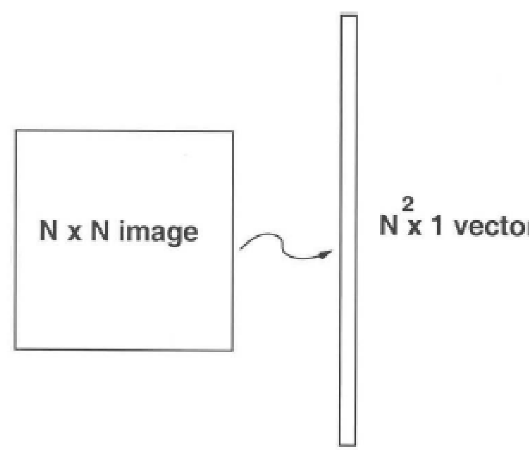
Face Recognition

- Computation of low-dimensional basis (i.e.,eigenfaces):

Step 1: obtain face images $I_1, I_2, \ldots, I_M$ (training faces)

(**very important:** the face images must be *centered* and of the same *size*)





Step 2: represent every image I_i as a vector Γ_i

Face Recognition

- Computation of the eigenfaces

Step 3: compute the average face vector Ψ:

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

Step 4: subtract the mean face:

$$\Phi_i = \Gamma_i - \Psi$$

Step 5: compute the covariance matrix C:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \quad (N^2 \mathrm{x} N^2 \text{ matrix})$$

$$\text{where } A = [\Phi_1\ \Phi_2 \cdots \Phi_M] \quad (N^2 \mathrm{x} M \text{ matrix})$$

Face Recognition

- Computation of the eigenfaces

Step 6: compute the eigenvectors u_i of AA^T ⇨ $AA^T u_i = \lambda_i u_i$

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

Face Recognition

Training images

Face Recognition

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$