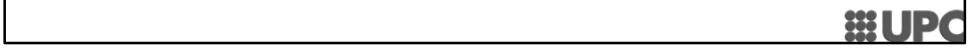
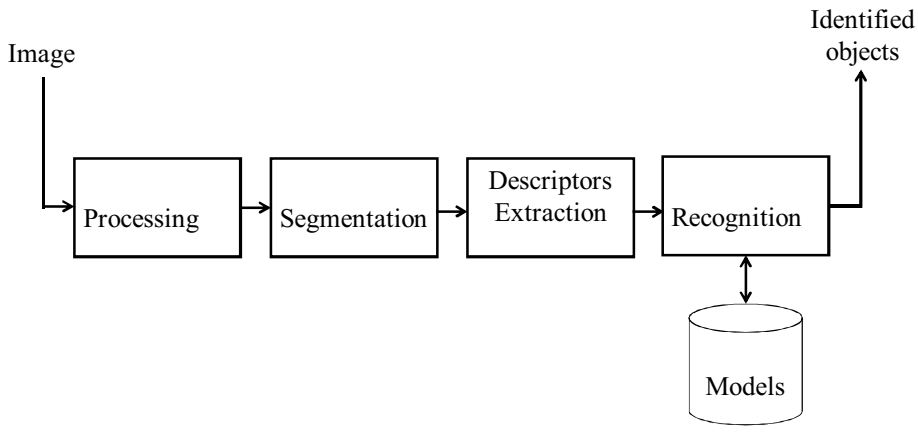
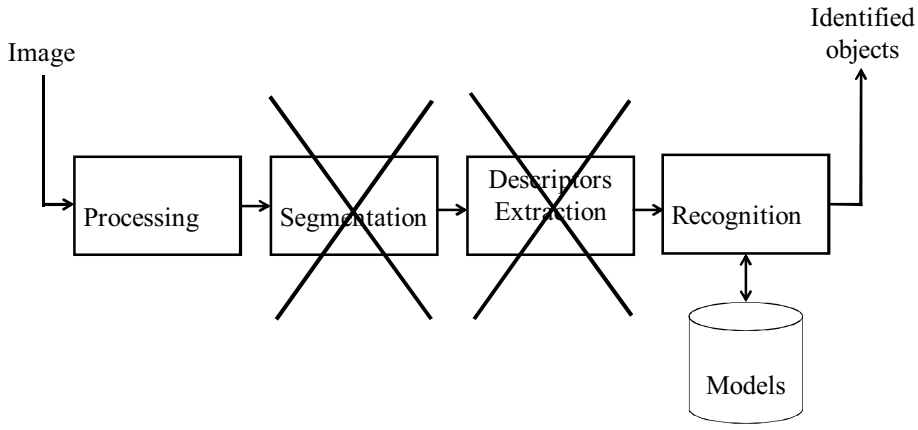
Local Features



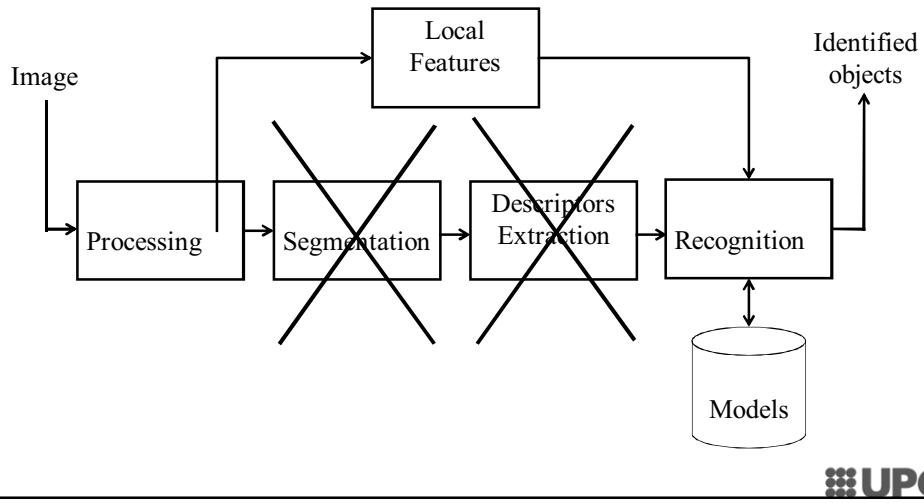
A computer vision system



A computer vision system



A computer vision system



Local Features

- Histograms
- Hough transform
- Corners
- Scale Invariant Feature transform (SIFT)
- Haar Features (face detection)

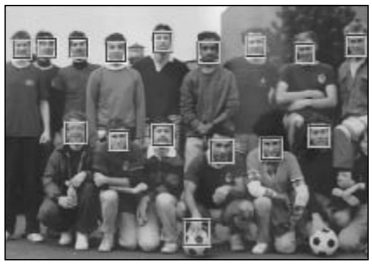
Haar features

- Real-time X detection

- Where X can be:
  - Faces
  - Traffic signs
  - Cats
  - Any object

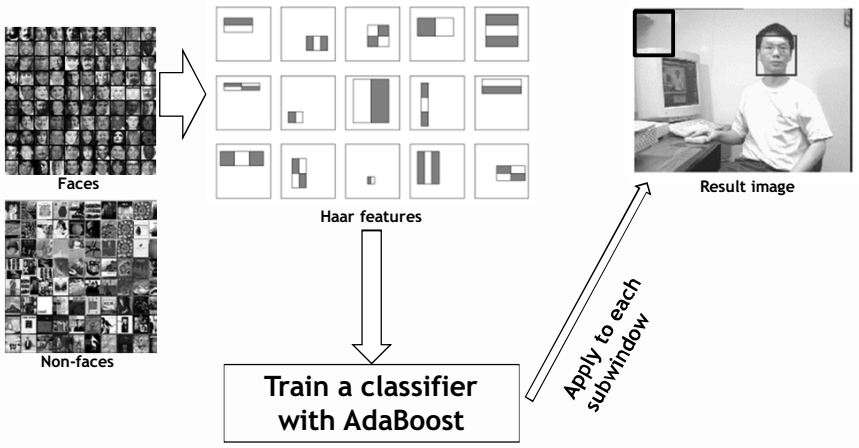
Robust Real-time Object Detection

Paul Viola and Michael Jones



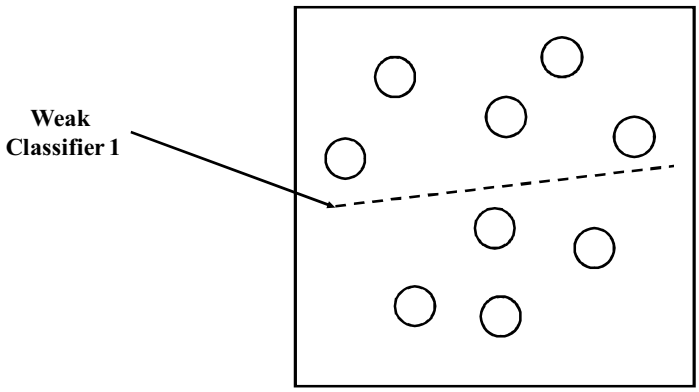
High-speed face detection with good accuracy

Viola-Jones Face Detector

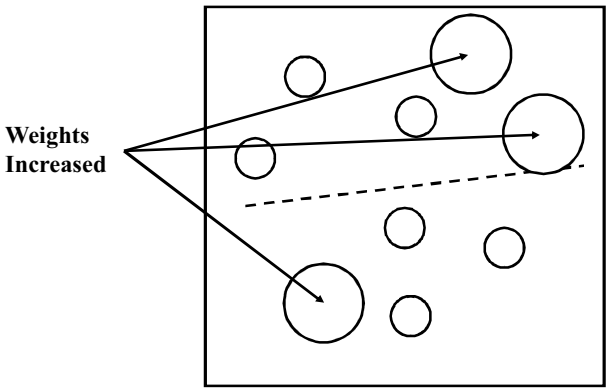


- Train with 5K positives, 350M negatives
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

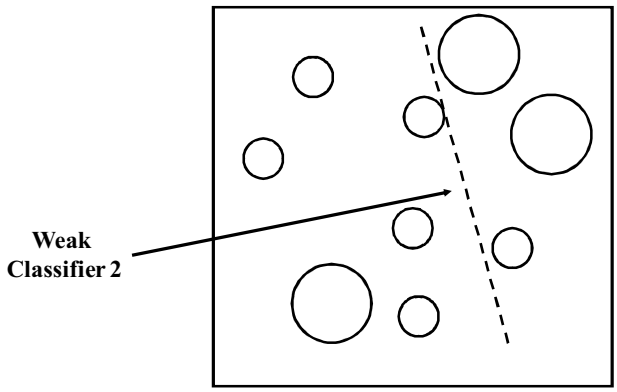
The Adaboost classifier



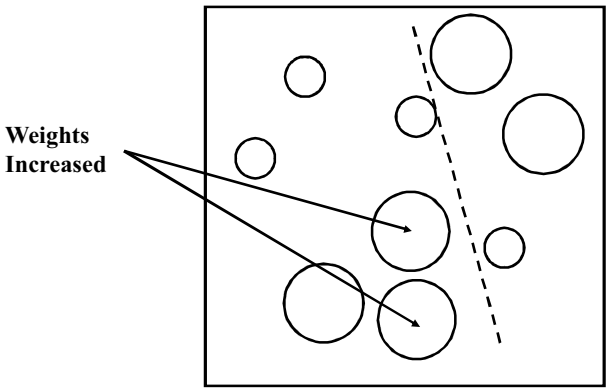
The Adaboost classifier



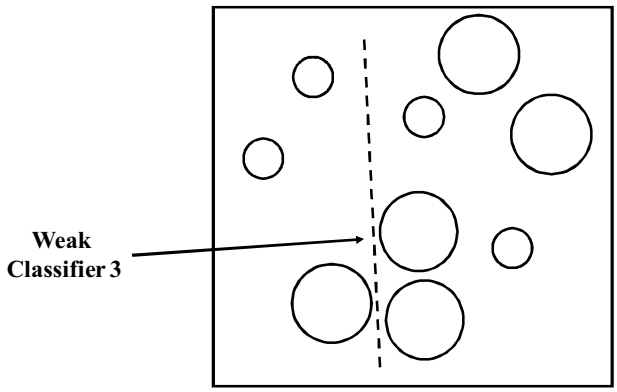
The Adaboost classifier



The Adaboost classifier

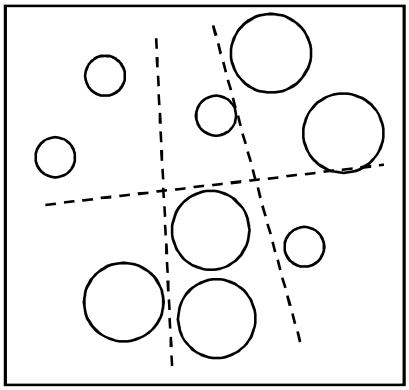


The Adaboost classifier



The Adaboost classifier

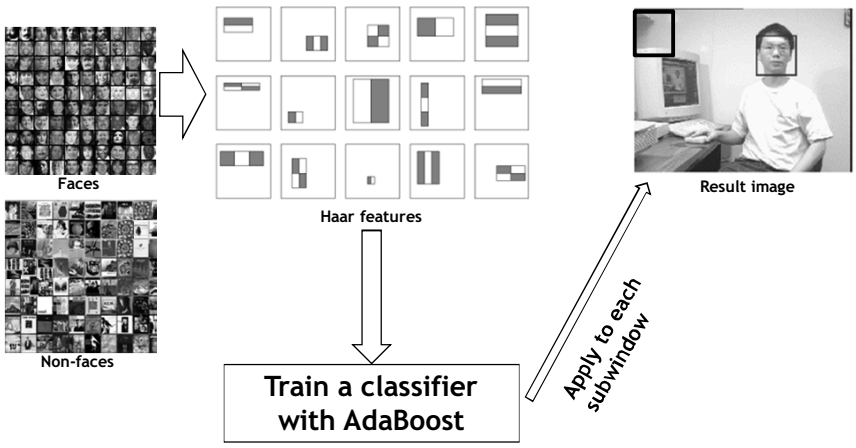
Final classifier is a combination of weak classifiers



Boosting Algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Viola-Jones Face Detector

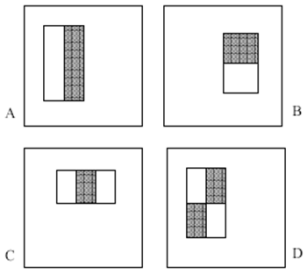


- Train with 5K positives, 350M negatives
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

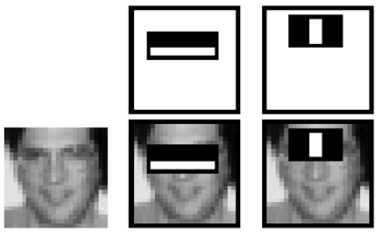
Face Detection System

- 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

Haar Features



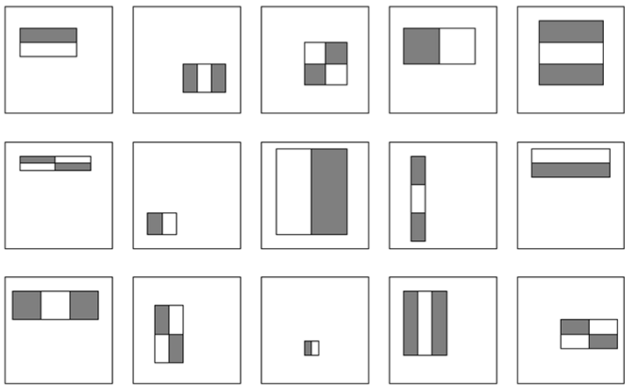
Rectangular filters



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

Local features: Subtract sum of pixels in white area from the sum of pixels in black area;

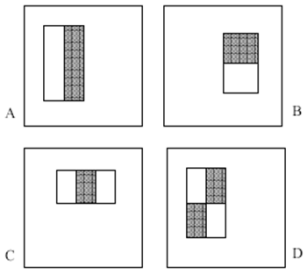
Large library of filters



Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

Image Features



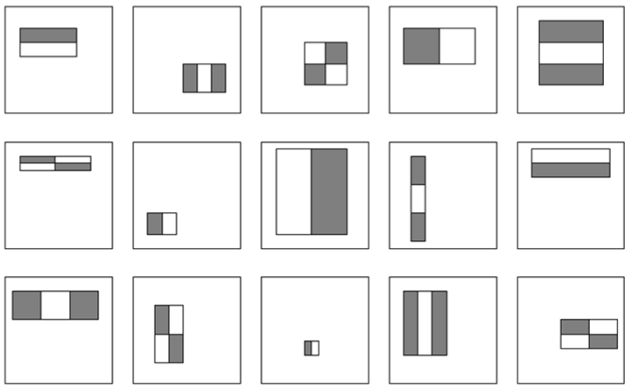
Rectangular filters

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

Local features: Subt… …m of pixels in black area

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

Large library of filters

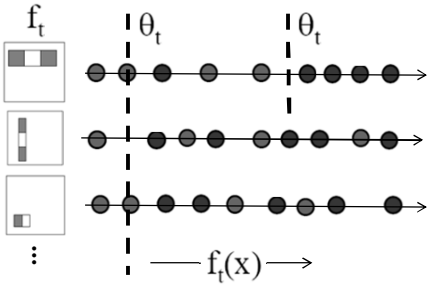


Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

AdaBoost for feature+classifier selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and (non-faces) training examples, in terms of *weighted* error.



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

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

For next round, reweight the examples according to errors, choose another filter/threshold combo.

The detector

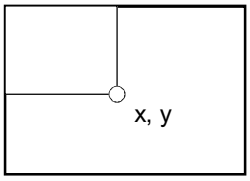
- A simple filter bank with learned weights applied across the image
- But with some notable performance-boosting implementation tricks…

Three big speed gains

- Integral image representation and rectangle features
- Selection of a small but effective feature set with AdaBoost
- Cascading simple detectors to quickly eliminate false positives

The integral image representation

An image representation that stores the sum of the intensity values above and to the left of the image point.

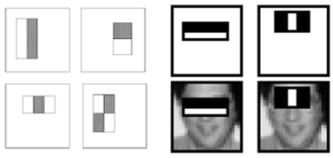


IntegralImage(x,y) = Sum of the values in the grey region

So what's it good for?

The integral image

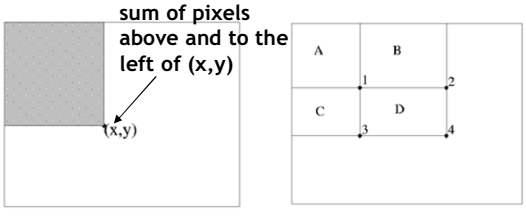
"Rectangular" filters



Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost



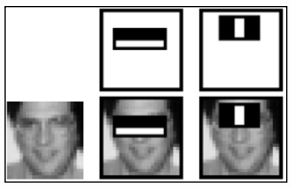
$$D = 1 + 4 - (2 + 3)$$
$$= A + (A + B + C + D) - (A + C + A + B)$$
$$= D$$

Speed gain number two: AdaBoost selected features

AdaBoost is used to select the best set of rectangular features.

AdaBoost iteratively trains a classifier by emphasizing misclassified training data.

Assigned feature weights are used to select the “most important” features.



Top two features weighted by AdaBoost

Intermediate results

The face detector using 200 AdaBoost-selected features achieved a 1 in 14084 false positive rate when turned for a 95% classification rate.

An 384x288 image took 0.7 seconds to scan.

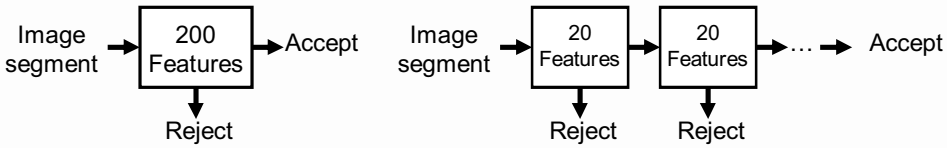
There are more improvements to be made…

Speed gain number three: Cascading detectors

Instead of applying all 200 filters at every location in the image, train several simpler classifiers to quickly eliminate easy negatives.

Each successive filter can be trained on true positives and the false positives passed by the filters before it.

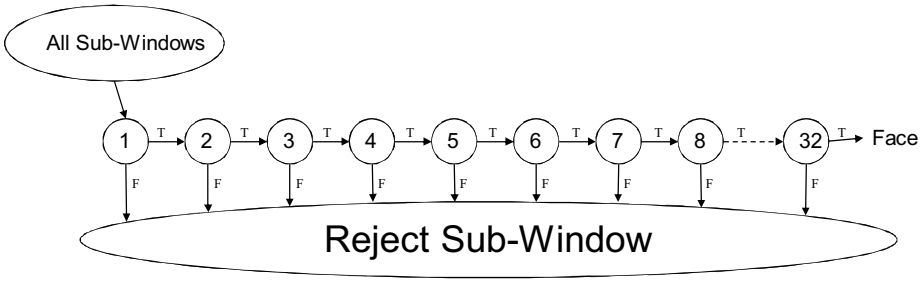
The filters are trained to allow approximately 10% false positives.



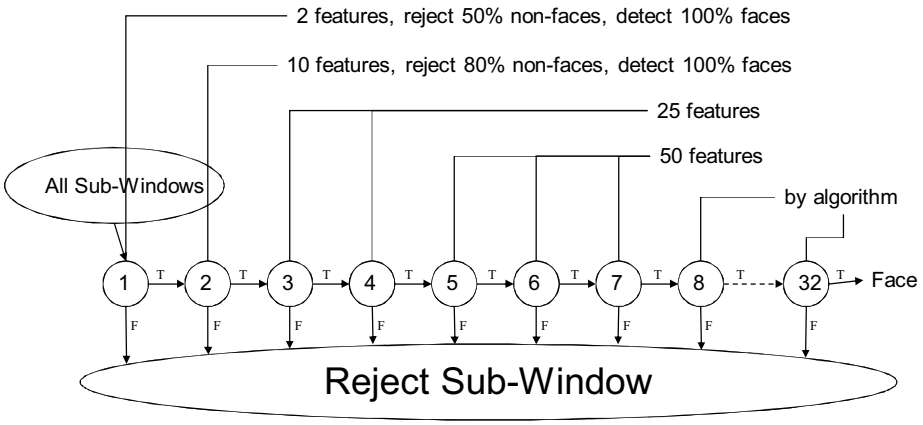
Structure of the Detector Cascade

Combining successively more complex classifiers in cascade

- 32 stages
- included a total of 4297 features



Structure of the Detector Cascade



Viola-Jones Face Detector: Results

Viola-Jones Face Detector: Results