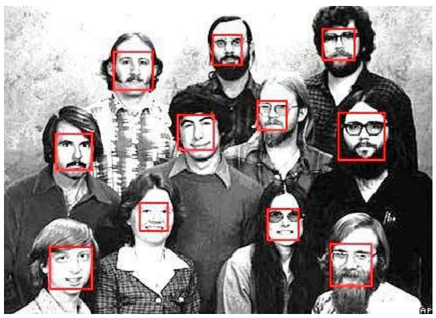
Face Detection

Importance of Face Detection

- The first step for any automatic face recognition system
- First step in many Human Computer Interaction systems
  - Expression Recognition
  - Cognitive State/Emotional State Recognition
- First step in many surveillance systems
- Tracking: Face is a highly non rigid object
- A step towards Automatic Target Recognition(ATR) or generic object detection/recognition
- Video coding……
- Facebook, google photos,…

What is Face Detection?

Given an image, tell whether there is any human face, if there is, where is it(or where they are).



Face Detection should

- Identify and locate human faces in an image regardless of their
  - position
  - scale
  - in-plane rotation
  - orientation
  - pose (out-of-plane rotation)
  - and illumination

Face Detection: current state

- State-of-the-art :
  - Front-view face detection can be done at >15 frames per second on 320x240 black-and-white images on a 700MHz PC with ~95% accuracy.
  - Detection of faces is faster than detection of edges!
- Side view face detection remains to be difficult.

Why Is Face Detection Difficult?

- **Pose (Out-of-Plane Rotation)**: frontal, 45 degree, profile, upside down
- **Presence or absence of structural components**: beards, mustaches, and glasses
- **Facial expression**: face appearance is directly affected by a person's facial expression
- **Occlusion**: faces may be partially occluded by other objects
- **Orientation (In-Plane Rotation)**: face appearance directly vary for different rotations about the camera's optical axis
- **Imaging conditions**: lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, gain control, lenses), resolution

Related Problems

- Face localization:
  - Aim to determine the image position of a single face
  - A simplified detection problem with the assumption that an input image contains only one face
- Facial feature extraction:
  - To detect the presence and location of features such as eyes, nose, nostrils, eyebrow, mouth, lips, ears, etc
  - Usually assume that there is only one face in an image
- Face recognition (identification)
- Facial expression recognition
- Human pose estimation and tracking

Research Issues

- Representation: How to describe a typical face?
- Scale: How to deal with face of different size?
- Search strategy: How to spot these faces?
- Speed: How to speed up the process?
- Precision: How to locate the faces precisely?
- Post processing: How to combine detection results?

Different Approaches

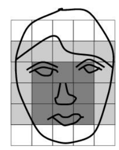
- Knowledge-based methods:
  - Encode what constitutes a typical face, e.g., the relationship between facial features
- Feature invariant approaches:
  - Aim to find structure features of a face that exist even when pose, viewpoint or lighting conditions vary
- Template matching:
  - Several standard patterns stored to describe the face as a whole or the facial features separately
- Appearance-based methods:
  - The models are learned from a set of training images that capture the representative variability of faces.
- Color/Skin Detection methods

Knowledge-Based Methods

- Top Top-down approach: Represent a face using a set of human-coded rules
  Example:
  - The center part of face has uniform intensity values
  - The difference between the average intensity values of the center part and the upper part is significant
  - A face often appears with two eyes that are symmetric to each other, a nose and a mouth
- Use these rules to guide the search process

Knowledge-Based Method: [Yang and Huang 94]

- Multi-resolution focus-of-attention approach
- Level 1 (lowest resolution): apply the rule "the center part of the face has 4 cells with a basically uniform intensity" to search for candidates
- Level 2: local histogram equalization followed by edge detection
- Level 3: search for eye and mouth features for validation



Knowledge-based Methods: Summary

- Pros:
  - Easy to come up with simple rules
  - Based on the coded rules, facial features in an input image are extracted first, and face candidates are identified
  - Work well for face localization in uncluttered background
- Cons:
  - Difficult to translate human knowledge into rules precisely: detailed rules fail to detect faces and general rules may find many false positives
  - Difficult to extend this approach to detect faces in different poses: implausible to enumerate all the possible cases

Feature-Based Methods

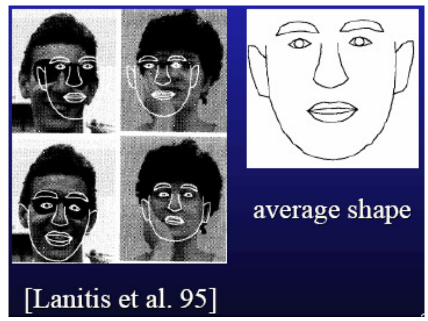
- Bottom-up approach: Detect facial features (eyes, nose, mouth, etc) first
- Facial features: edge, intensity, shape, texture, color, etc
- Aim to detect invariant features
- Group features into candidates and verify them

Feature-Based Methods: Summary

- Pros: Features are invariant to pose and orientation change
- Cons:
  - Difficult to locate facial features due to several corruption (illumination, noise, occlusion)
  - Difficult to detect features in complex background

Template Matching Methods

- Store a template
  - Predefined: based on edges or regions
- Deformable: based on facial contours (e.g., Snakes)
- Templates are hand-coded (not learned)
- Use correlation to locate faces



Template-Based Methods: Summary

- Pros:
  - Simple
- Cons:
  - Templates needs to be initialized near the face images
  - Difficult to enumerate templates for different poses (similar to knowledge-based methods)

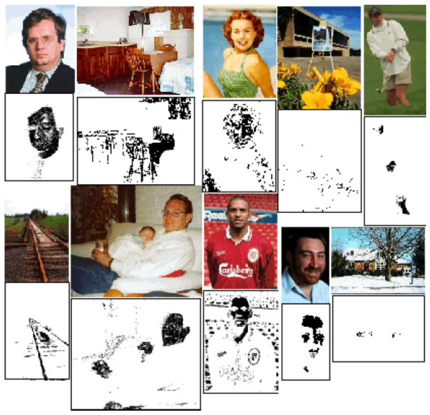
Appearance-Based Methods: Classifiers

- Neural network
  - Multilayer Perceptrons
- Principapl Component Analysis (PCA), Factor Analysis
- Support vector machine (SVM)
- Mixture of PCA, Mixture of factor analyzers
- Distribution Distribution-based method
- Naïve Bayes classifier
- Hidden Markov model
- Sparse network of winnows (SNoW)
- Kullback relative information
- Inductive learning: C4.5
- Adaboost □□
- □□...

Color-Based Face Detector

- Distribution of skin color across different ethnic groups
  - Under controlled illumination conditions: compact
  - Arbitrary conditions: less compact
- Color space
  - RGB, normalized RGB, HSV, HIS, YCrCb, YIQ, UES, CIE XYZ, CIE LIV, …
- Statistical analysis
  - Histogram, look-up table, Gaussian model, mixture model, …

Experimental Results [Jones and Rehg 02]



- Does a decent job
- May have lots of false positives in the raw results
- Need further processing to eliminate false negatives and group color pixels for face detection
- See also [Hsu et al 02]

Color-Based Face Detector: Summary

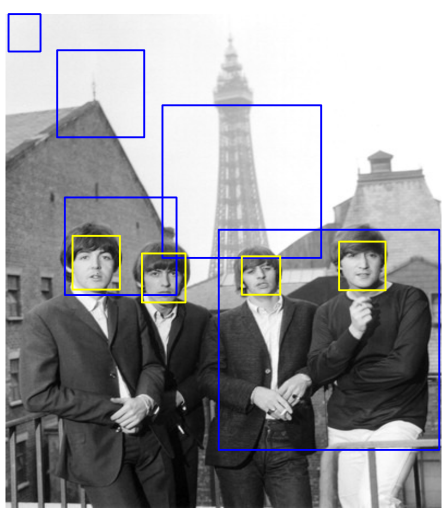
- Pros:
  - Easy to implement
  - Effective and efficient in constrained environment
  - Insensitive to pose, expression, rotation variation
- Cons:
  - Sensitive to environment and lighting change
  - Noisy detection results (body parts, skin-tone line regions)

Robust Real-time Face Detection

by

Paul Viola and Michael Jones, 2002

Face Detect: Sliding Windows



1. Basic idea:

slide a window across image and evaluate a face model at every location. Try all possible rectangle locations, sizes

2. test:

classify if rectangle contains a face (and only the face)

Note: 1000's more false windows then true ones.

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: Overview

- Robust – very high Detection Rate (True-Positive Rate) & very low False-Positive Rate… always.
- Real Time – For practical applications at least 2 frames per second must be processed.
- Face **Detection** – not recognition. The goal is to distinguish faces from non-faces (face **detection** is the first step in the **identification** process)

Three goals

1. *Feature Computation*: what features? And how can they be computed as quickly as possible
2. *Feature Selection*: select the most discriminating features
3. *Real-timeliness*: must focus on potentially positive areas (that contain faces)

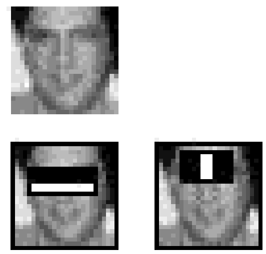
How did Viola & Jones deal with these challenges?

Three solutions

1. *Feature Computation*
   The “**Integral**” image representation
2. *Feature Selection*
   The AdaBoost training algorithm
3. *Real-timeliness*
   A cascade of classifiers

Features

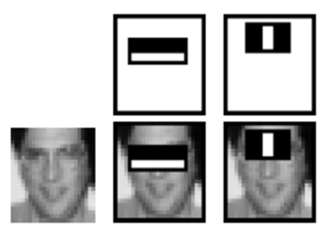
- Can a simple feature (i.e. a value) indicate the existence of a face?
- All faces share some similar properties
  - The eyes region is darker than the upper-cheeks.
  - The nose bridge region is brighter than the eyes.
  - **That is useful domain knowledge**
- Need for encoding of Domain Knowledge:
  - ***Location - Size:*** eyes & nose bridge region
  - ***Value:*** darker / brighter

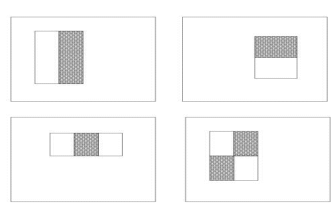


Forehead, eye features can be captured

Rectangle features

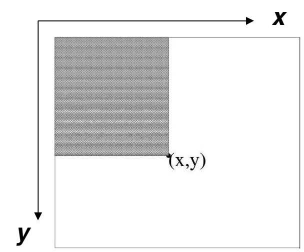
- Rectangle features:
  - Value = ∑ (pixels in black area) - ∑ (pixels in white area)
  - Three types: two-, three-, four-rectangles, Viola&Jones used two-rectangle features
  - For example: the difference in brightness between the white &black rectangles over a specific area
- Each feature is related to a special <u>location</u> in the sub-window
- Each feature may have any <u>size</u>
- Why not pixels instead of features?
  - Features encode domain knowledge
  - Feature based systems operate faster





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:

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

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

Example

IMAGE

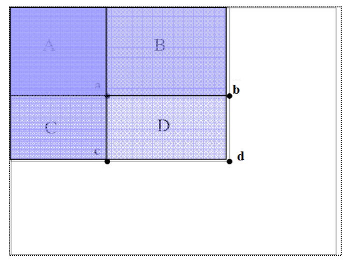
0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

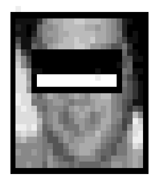
INTEGRAL IMAGE

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

Rapid computation of rectangular features

- Back to feature evaluation . . .
- Using the integral image representation we can compute the value of any rectangular sum (part of features) in **constant time**
  - For example the integral sum inside rectangle D can be computed as: $ii(\mathrm{d}) + ii(\mathrm{a}) - ii(\mathrm{b}) - ii(\mathrm{c})$
- 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)

Overview | **Integral Image** | AdaBoost | Cascade

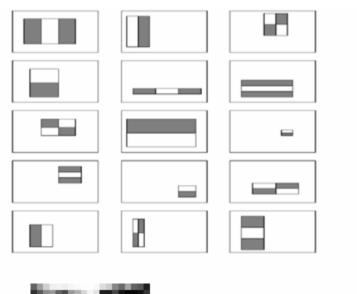
Three goals

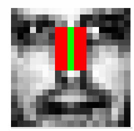
1. *Feature Computation*: features must be computed as quickly as possible 
2. *Feature Selection*: select the most discriminating features
3. *Real-timeliness*: must focus on potentially positive image areas (that contain faces)

How did Viola & Jones deal with these challenges?

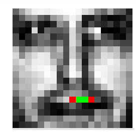
Feature selection

- Problem: Too many features
  - In a sub-window (24x24) there are ~160,000 features (all possible combinations of orientation, location and scale of these feature types)
  - impractical to compute all of them (computationally expensive)
- We have to select a subset of relevant features – which are informative - 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

AdaBoost Algorithm

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

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

Weak classifier

Strong classifier

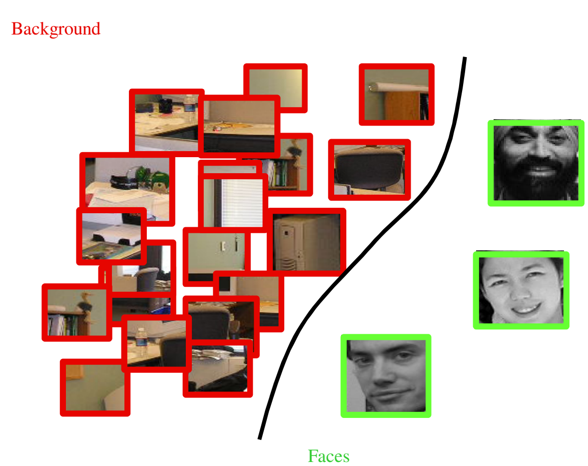
Image

Weight

AdaBoost - *Characteristics*

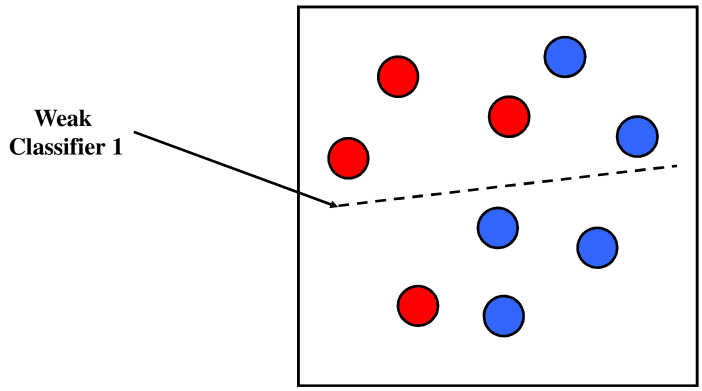
- Features as weak classifiers
  - Each single rectangle feature may be regarded as a simple weak classifier
- An iterative algorithm
  - AdaBoost performs a series of trials, each time selecting a new weak classifier
- Weights are being applied over the set of the example images
  - During each iteration, each example/image receives a weight determining its importance

Classification (Discriminative)

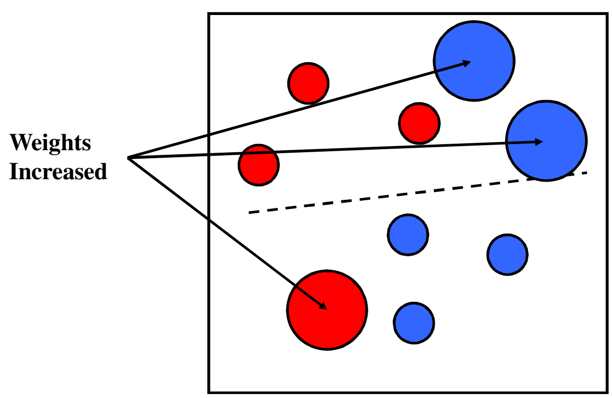


In some feature space

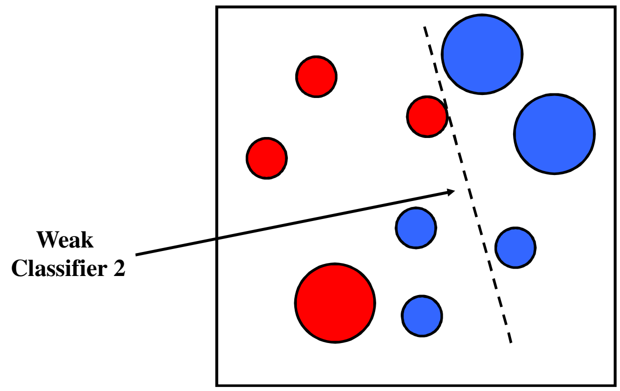
Boosting illustration



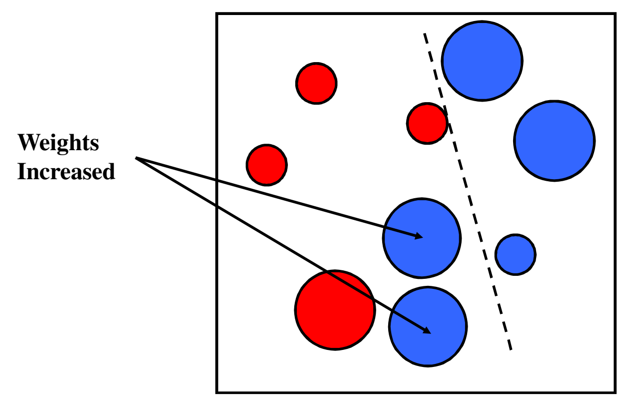
Boosting illustration



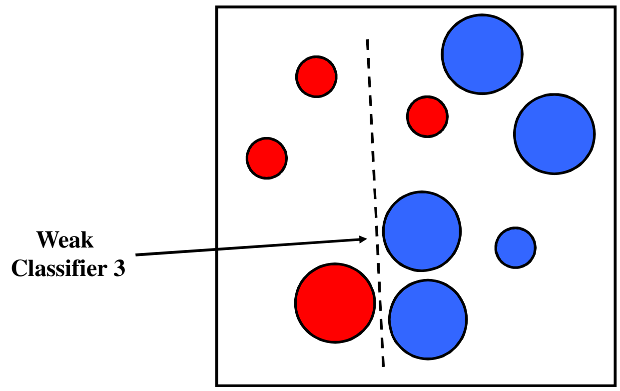
Boosting illustration



Boosting illustration

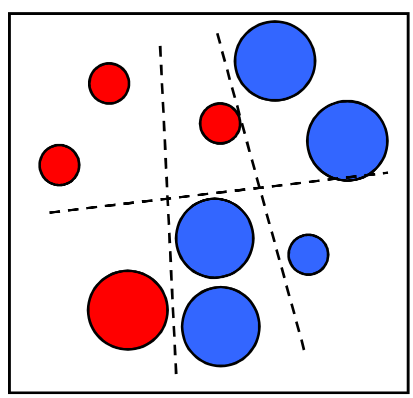


Boosting illustration



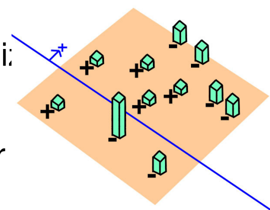
Boosting illustration

Final classifier is
a combination of weak
classifiers



AdaBoost

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



$$h_{\text{strong}}(\mathbf{x}) = \begin{cases} 1 & \alpha_1 h_1(\mathbf{x}) + \ldots + \alpha_n h_n(\mathbf{x}) \geq \frac{1}{2}(\alpha_1 + \ldots + \alpha_n) \\ 0 & \text{otherwise} \end{cases}$$

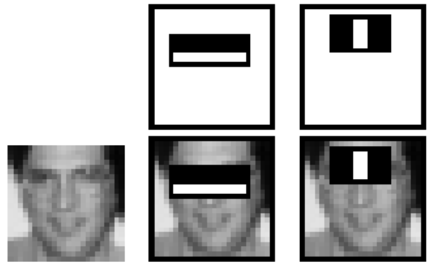
Constructing the classifier

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)
  - Use sorting to quickly scan for optimal threshold
- Select best filter/threshold combination
- Weight is a simple function of error rate
- Reweight examples
  - (There are many tricks to make this more efficient.)

44

Image Features

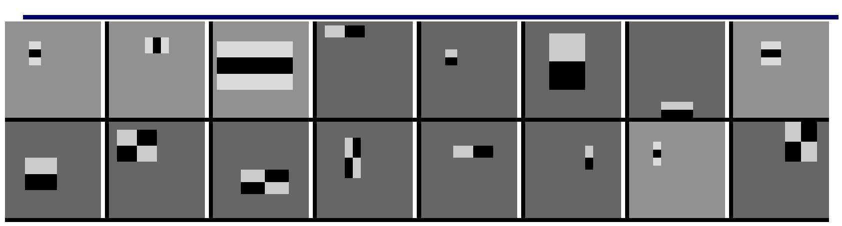


$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \ldots$$

$$f_i(x) = \begin{cases} 1 & \text{if } g_i(x) > \theta_i \\ -1 & \text{otherwise} \end{cases}$$

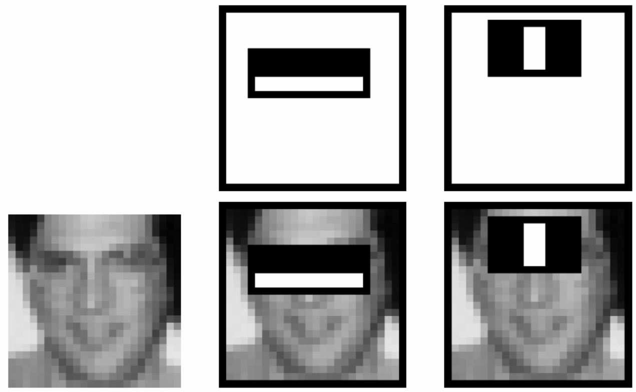
Need to: (1) Select Features i=1..n,
(2) Learn thresholds θ_i ,
(3) Learn weights α_i

Example: the selected/learned features



Boosting for face detection

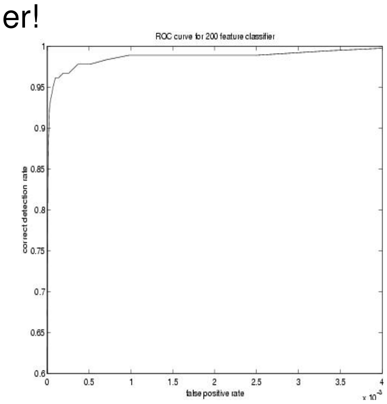
- First two features selected by boosting:



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

Now we have a good face detector

- We can build a 200-feature classifier!
- Experiments showed that a 200-feature classifier achieves:
  - 95% detection rate
  - FP rate (1 in 14084)
  - Scans all sub-windows of a 384x288 pixel image in 0.7 seconds
- The more the better (?)
  - Gain in classifier performance
  - Lose in CPU time
- Verdict: good & fast, but not enough
  - Competitors achieve close to **1 in a 1.000.000 FP rate!**
  - 0.7 sec / frame **IS NOT** real-time.



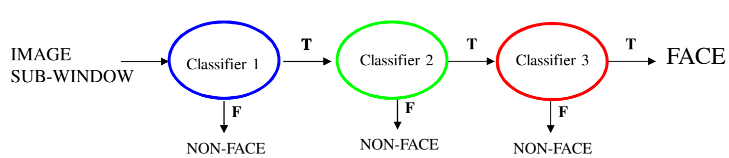
Three goals

1. *Feature Computation*: features must be computed **as quickly as possible**
2. *Feature Selection*: select the **most discriminating** features
3. *Real-timeliness*: must **focus** on **potentially positive** image areas (that contain faces)

How did Viola & Jones deal with these challenges?

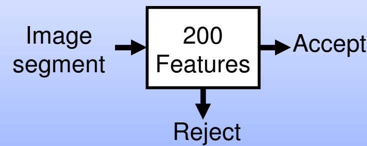
Attentional cascade

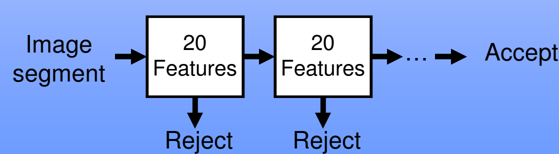
- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



Cascading detectors

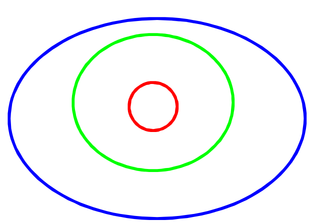
Instead of applying all 200 filters at every location in the image, train several simpler classifiers to quickly eliminate easy negatives.



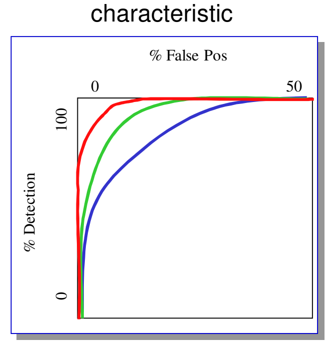


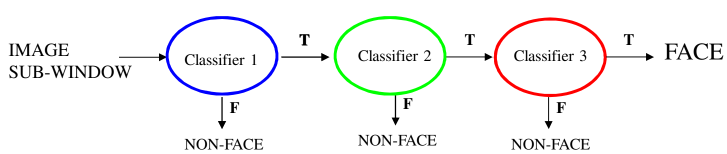
Attentional cascade

- Chain classifiers that are progressively more complex and have lower false positive rates:



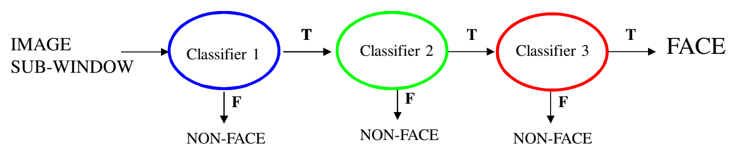
Receiver operating characteristic



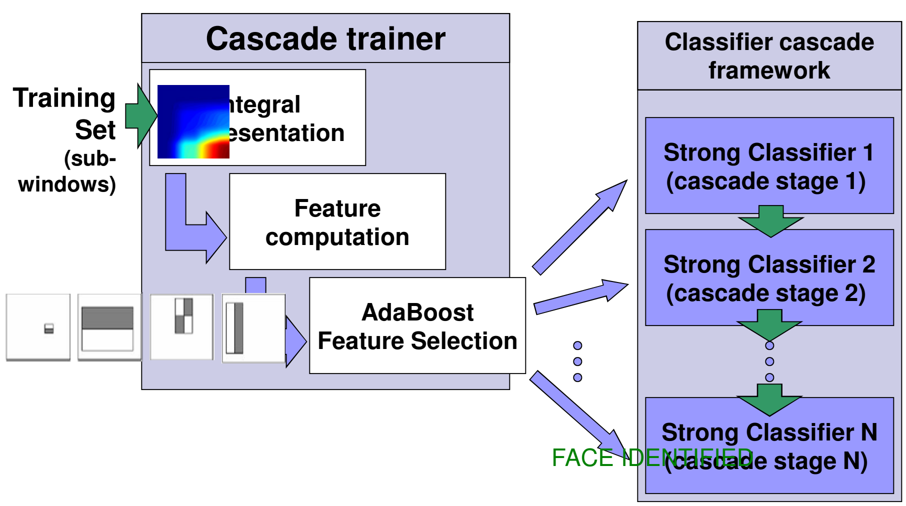


Attentional cascade

- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages
- A detection rate of 0.9 and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$)



Training phase / Testing phase



Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a *validation set*
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

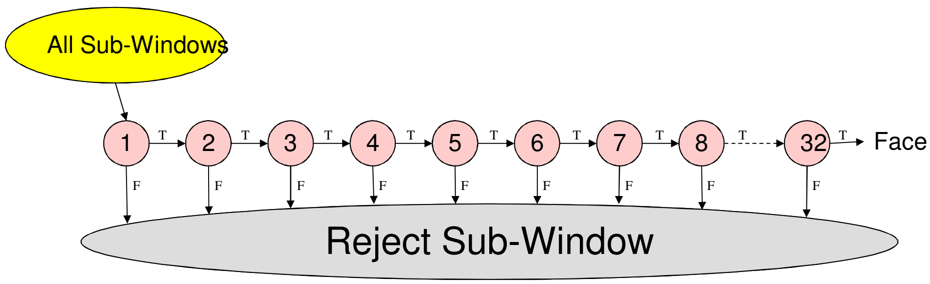
The implemented system

- 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

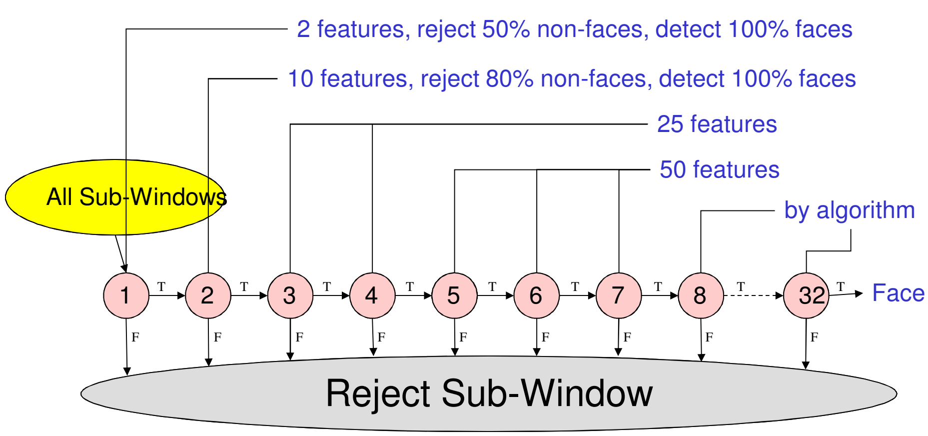
Structure of the Detector Cascade

Combining successively more complex classifiers in cascade

- 32 stages
- included a total of 4297 features



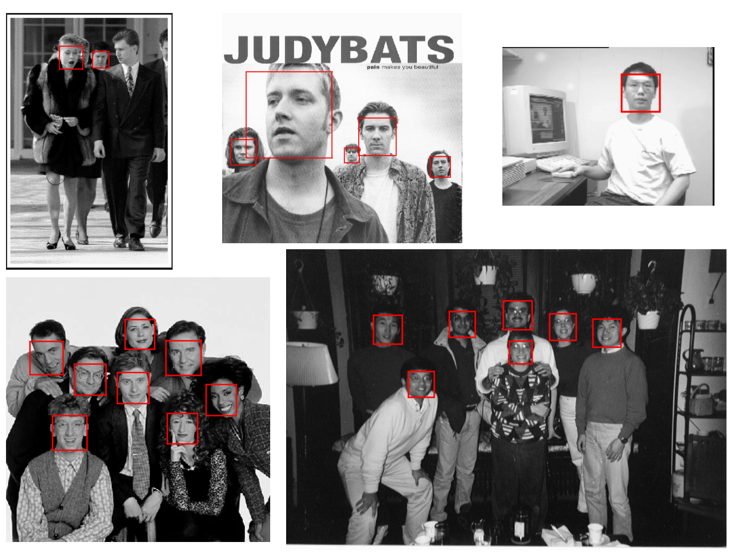
Structure of the Detector Cascade



System performance

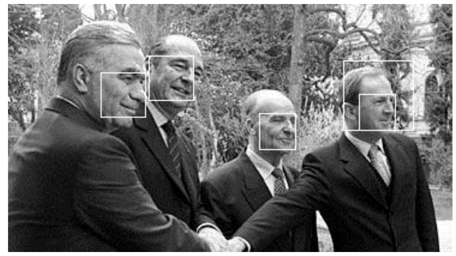
- Training time: “weeks” on 466 MHz Sun workstation
- 32 layers, total of 4297 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

Output of Face Detector on Test Images



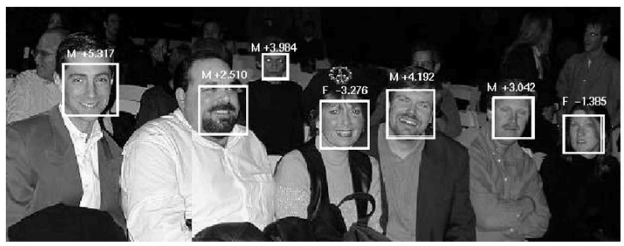
Other detection tasks

Facial Feature Localization

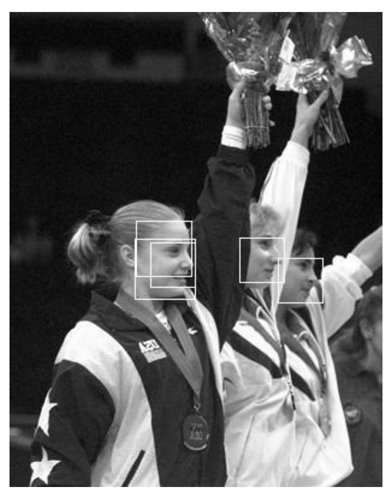


Profile Detection

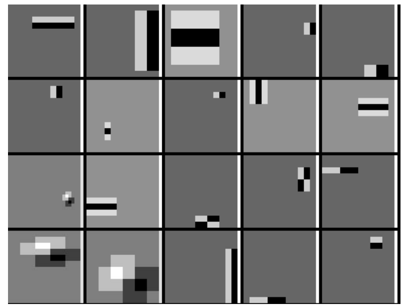
Male vs. female



Profile Detection



Profile Features



pros …

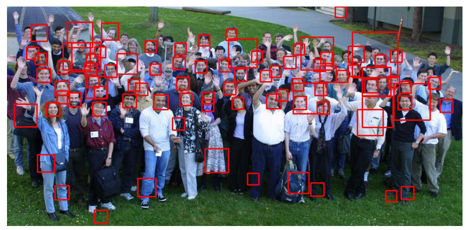
- Extremely fast feature computation
- Efficient feature selection
- Scale and location invariant detector
  - Instead of scaling the image itself (e.g. pyramid-filters), we scale the features.
- Such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, hands)

… and cons

- Detector is most effective only on frontal images of faces
  - can hardly cope with 45º face rotation
- Sensitive to lighting conditions
- We might get multiple detections of the same face, due to overlapping sub-windows.

Face Detection: A Solved Problem?

- Not quite yet…
- Factors:
  - Shadows
  - Occlusions
  - Robustness
  - Resolution
- Lots of potential applications
- Can be applied to other domains



OpenCV

- CascadeClassifier cascade;
- cascade.load("haarcascade_frontalface_default.xml"));
- vector<cv::Rect> faces;
- cascade.detectMultiScale(gray, faces, 1.2, 3);