ITMO

Face Detection

Computer Vision

Outline

- Face detection problem
- Weak classifiers
- Integral images
- Strong classifiers and boosting
- Cascade classifiers

Detection algorithms requirements

Algorithm requirements

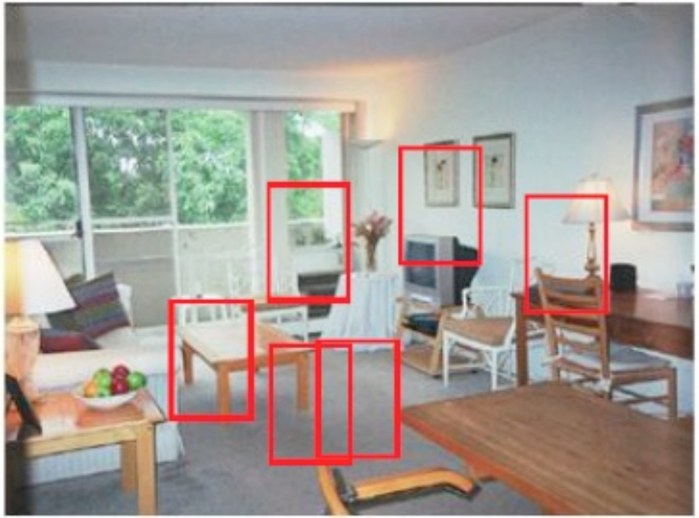
- For a 1MP image, you need to check about 1M windows (for example, for the case of face detection)
- One image usually has up to 10 faces
- To avoid false positives, the type 2 error must be below 10^{-6}
- Need to reject false windows as fast as possible

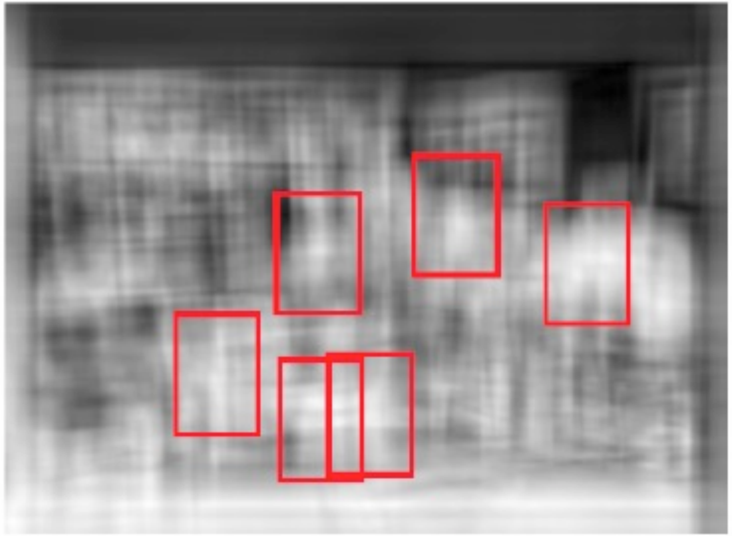
Viola-Jones detector

- Fundamental method for finding objects in a real-time image
- Learning is very slow, but searching is very fast
- Main ideas:
  - Use *weak classifiers* to find features of objects
  - Use *integral images* for fast feature calculation
  - Use *boosting* to select features
  - Use *cascade* for quick rejection of windows without a face

Weak classifiers. Example: find a chair

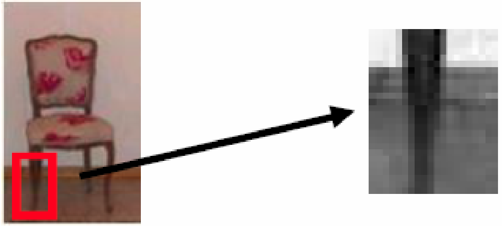
Weak classifiers. Example: find a chair





Let use use small fragments of an image and look not for a chair, but for a part of the chair

Weak classifiers. Example: find a chair

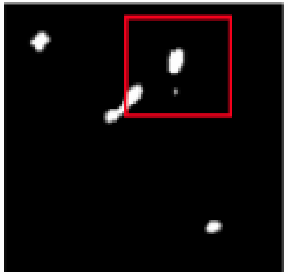


Allow finding chair legs

Weak classifiers

- Let's define a family of visual features that can be used as weak classifiers

$$\longrightarrow h_i(I, x, y) \longrightarrow$$



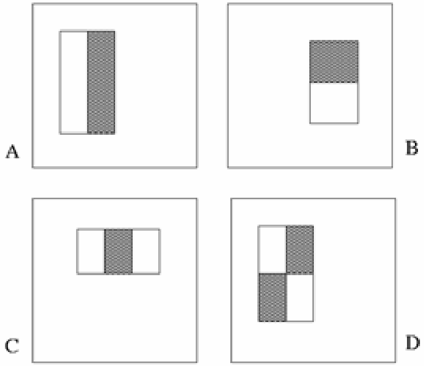
- **Weak classifier** has
  - **An image** as an **input**
  - **A binary** response as an **output**

Haar-like features

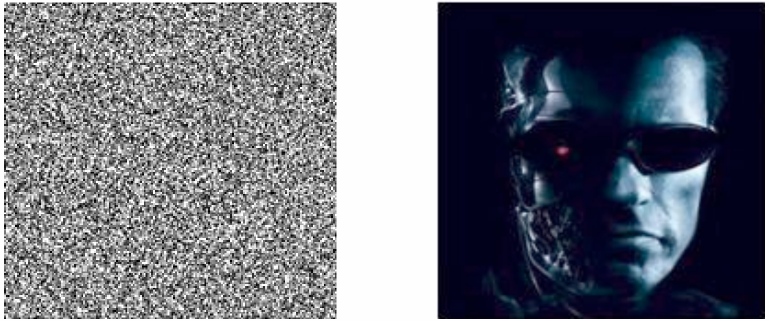
- A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image.

$$\boldsymbol{Value} = \sum(\boldsymbol{pixels\ in\ black\ area}) - \sum(\boldsymbol{pixels\ in\ white\ area})$$

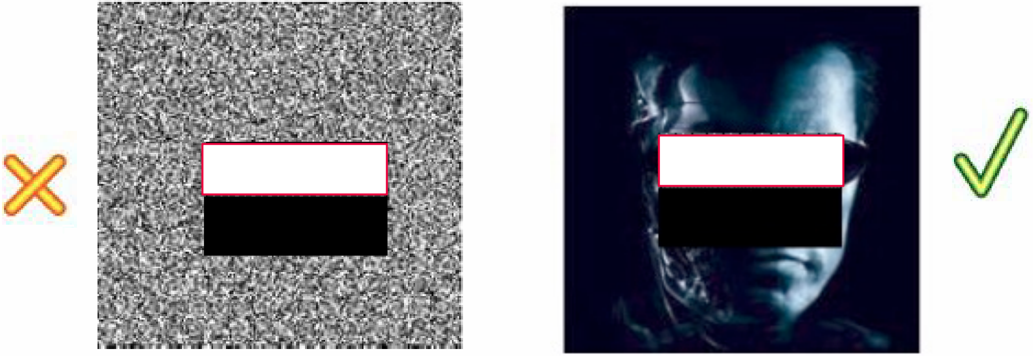
  - 2-rectangle feature (A, B)
  - 3-rectangle feature (C)
  - 4-rectangle feature (D)



Haar-like features. Example



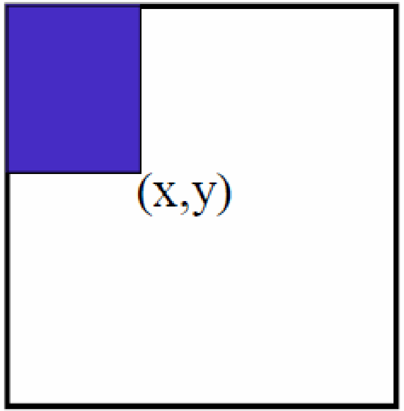
Source



Result

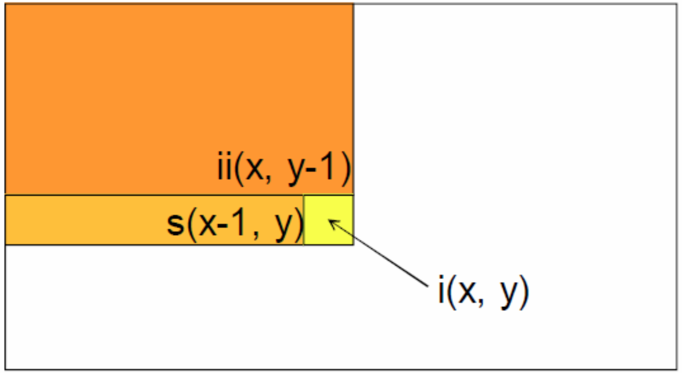
Integral image

- The intensity value of each pixel (x,y) is equal to the sum of the intensity values of all pixels to the left and above the pixel (x,y) inclusive
- The integral image is calculated in one image pass



Integral image. Calculation

- Single line 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);`
- OpenCV C++: `cv::integral(i, ii);`
- OpenCV Python: `ii = cv2.integral(i)`

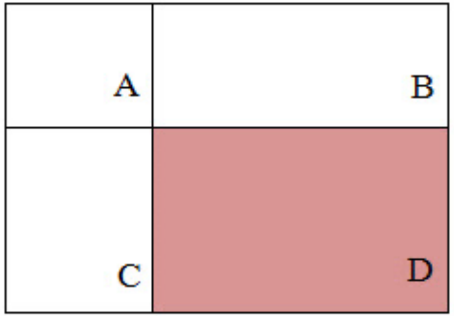


Integral image

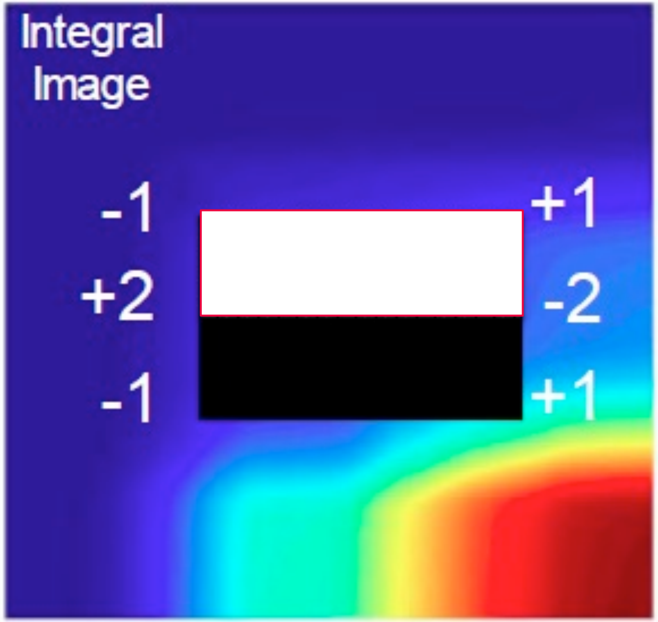
- **Calculating the sum in a rectangle**
  - Let A, B, C, D be the integral image values at the corners of the rectangle
  - Then the sum of the pixel intensities in the original image can be calculated by the following formula:

$$sum = A - B - C + D$$

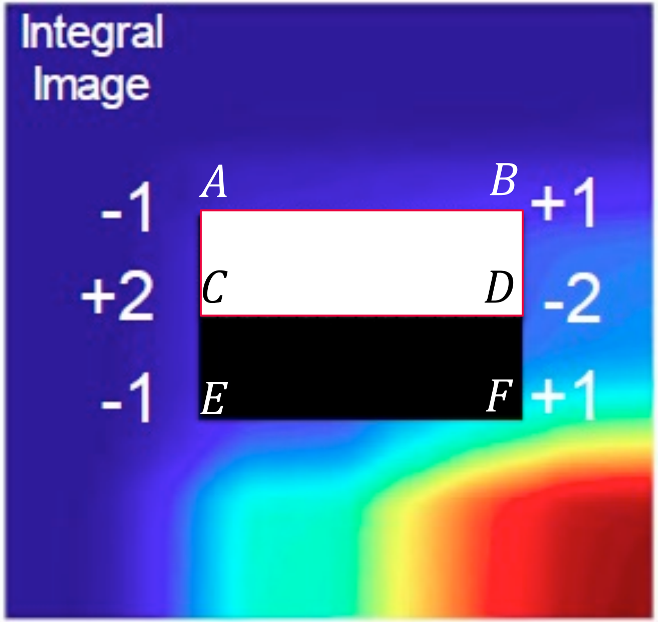
  - 3 addition operations for any rectangular area



Integral image. Example



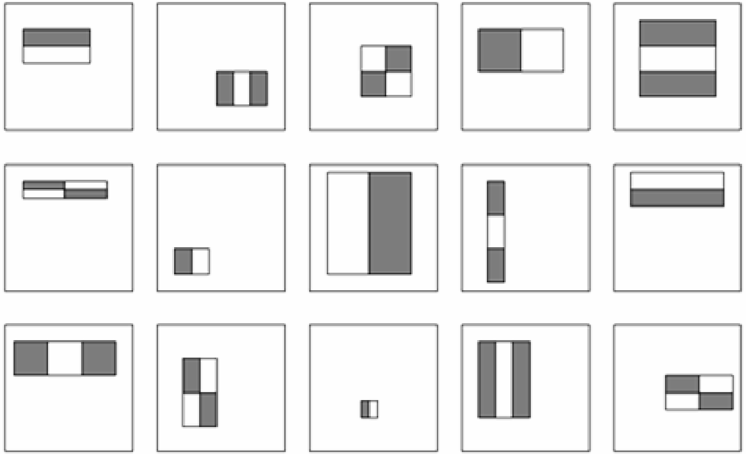
Integral image. Example



$$
\begin{aligned}
sum &= C - D - E + F - (A - B - C + D) \\
&= -A + B + 2C - 2D - E + F
\end{aligned}
$$

Viola-Jones detector

- **Feature selection**
  - For a 24x24 pixel window, the number of possible rectangular features is ~160,000
  - During the search process, it is impossible to calculate all these features
  - A good classifier should use a small subset of all possible features
  - How to choose such a subset?



Strong classifier. Boosting

ITMO

- *Boosting* is a classification scheme based on combining weak classifiers into a more accurate strong classifier
- **Training** consists of several *boosting rounds:*
    - At each stage, a weak classifier is selected, which worked best on samples that turned out to be difficult for previous classifiers
    - "Difficulty" is encoded using the weights assigned to the samples from the training set
    - A strong classifier is compiled as a linear combination of weak classifiers

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

Strong classifier. AdaBoost

- Let us have:
  - a set of weak classifiers {h}
  - a training set *T*: $(x_1, y_1), \ldots, (x_m, y_m)$, where $x_i \in X, y_i \in \Theta = \{-1,+1\}$
- Initialize weights $D_1(i) = {}^1/_m$
- For each $k = \overline{1, K}$
  1. Train h_k with minimal error ε_k
  2. Calculate the weight α_k of hypothesis h_k
  3. Reweight samples D_{k+1}
- Strong classifier is a weighted sum

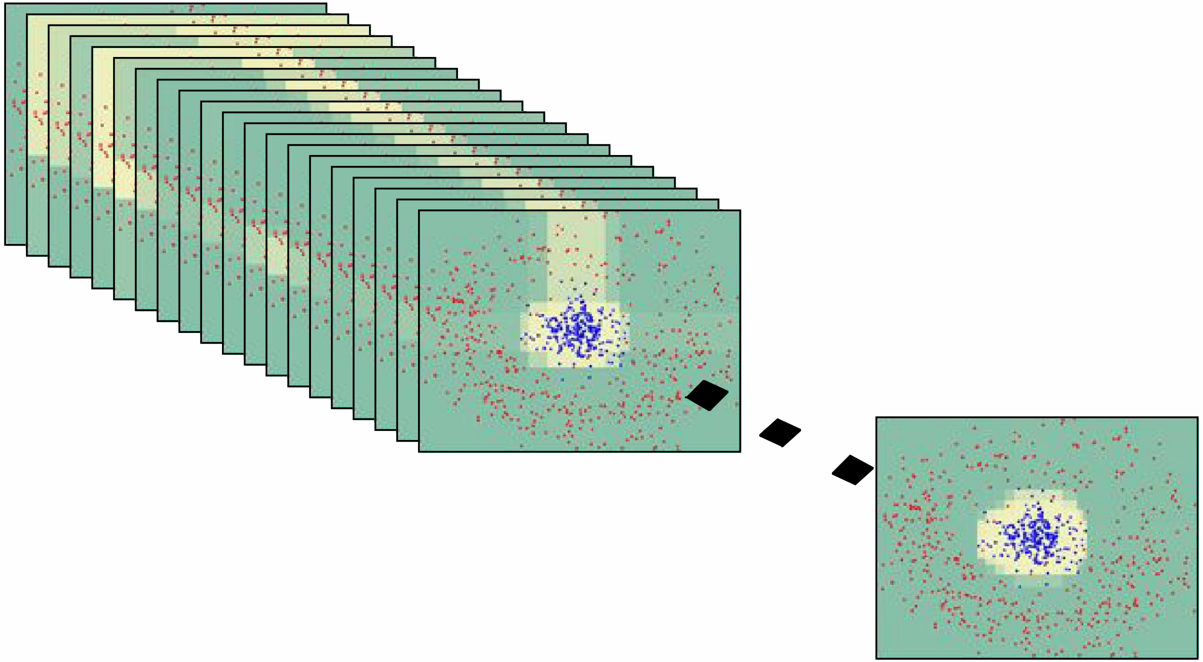
$$\varepsilon_k = Pr_{i\sim D_k}[h_k(x_i) \neq y_i]$$

$$\alpha_k = \frac{1}{2} ln\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$$

$$D_{k+1}(i) = \frac{D_k(i)}{Z_k} \cdot \begin{cases} e^{-\alpha_k}, if\ h_k(x_i) = y_i \\ e^{\alpha_k}, if\ h_k(x_i) \neq y_i \end{cases}$$

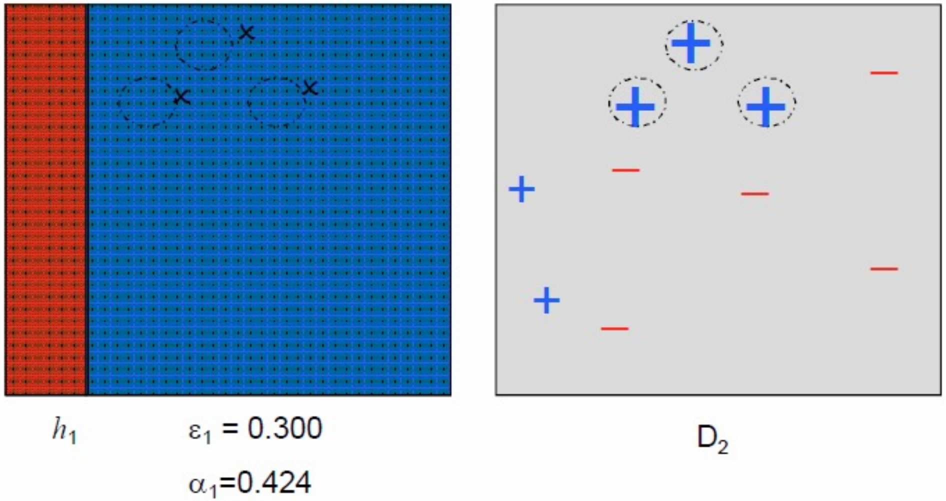
$$H(x) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k h_k(x)\right)$$

Strong classifier. AdaBoost



Strong classifier. Example

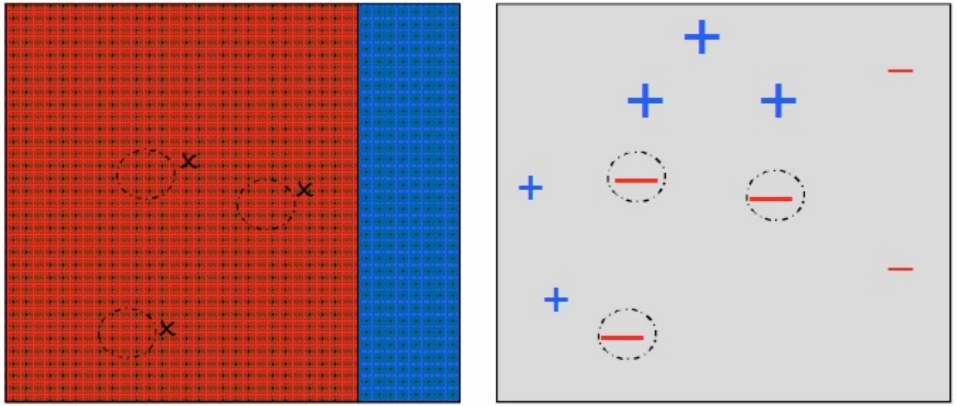
Phase 1 of 3



α_i – weights
ε_i – errors
h_i – weak classifiers

Strong classifier. Example

Phase 2 of 3



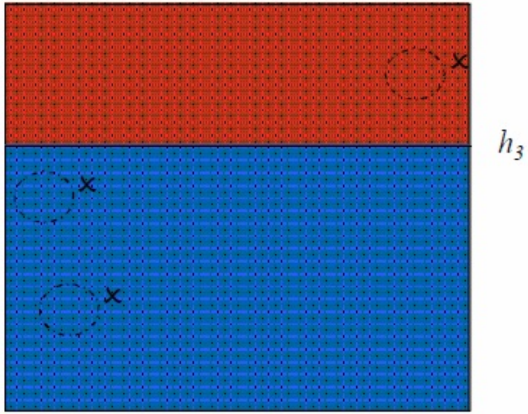
$\varepsilon_2 = 0.196$ h_2 D_2

$\alpha_2 = 0.704$

α_i – weights
ε_i – errors
h_i – weak classifiers

Strong classifier. Example

Phase 3 of 3



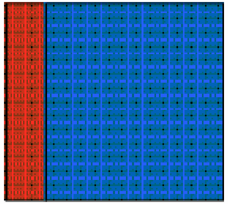
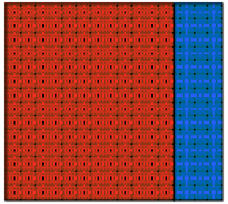
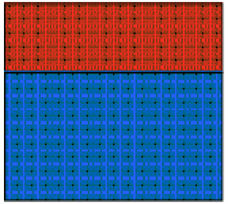
$\varepsilon_3 = 0.344$

$\alpha_2 = 0.323$

α_i – weights
ε_i – errors
h_i – weak classifiers

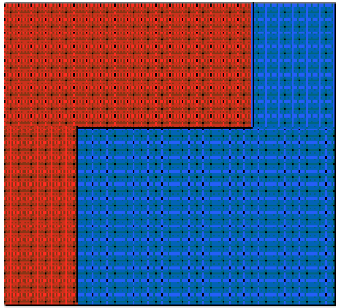
Strong classifier. Example

Final strong classifier

0,42  +0,70  +0,32 

$$H = \operatorname{sign}\big(0{,}42 \cdot h_1(x) + 0{,}70 \cdot h_2(x) + 0{,}32 \cdot h_3(x)\big),$$

$$h_i(x) = \begin{cases} -1, x \to \Theta_0 \\ +1, x \to \Theta_1 \end{cases}.$$



Strong classifier. AdaBoost

- **Advantages**
  - High convergence rate
  - High generalizing ability
  - Possibility of very efficient software implementation and parallelization
  - Simplicity of the method and lack of parameters
- **Disadvantages**
  - Hard to estimate the required number of training iterations

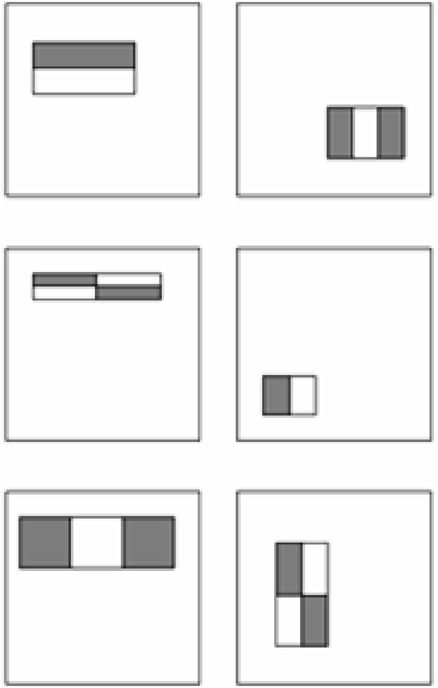
Face detection

- **Boosting for face detection**
  - Define weak classifiers based on rectangular features
  - For each boosting phase:
    - Calculate each rectangular feature on each training set sample
    - Choose the best threshold for each feature
    - Choose best feature/threshold pair
    - Reweight the training set
  - Computational complexity of training process is $O(MNK)$
    - M – number of phases,
    - N – number of samples,
    - K – number of features.

Face detection

- **Weak classifiers**

$$h_t(x) = \begin{cases} 1, if\ f_t(x) > \Theta_t \\ 0, else \end{cases},$$

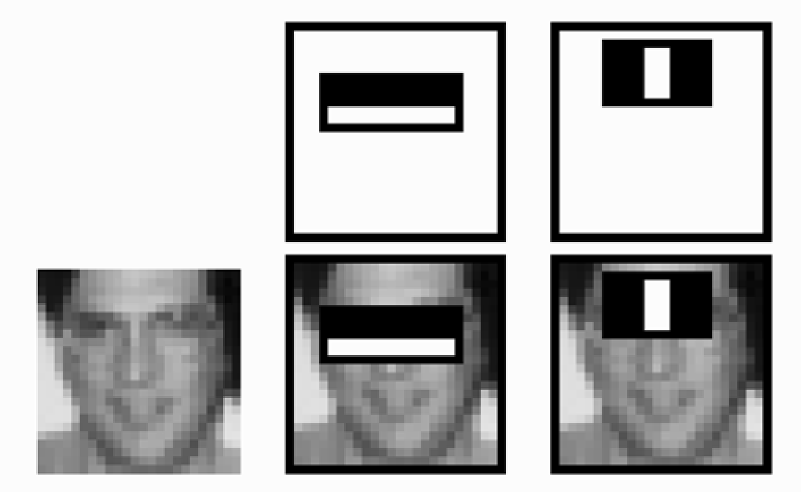


where $h_t(x)$ – weak classifier

- $f_t(x)$ – feature value
- Θ_t – threshold

Face detection

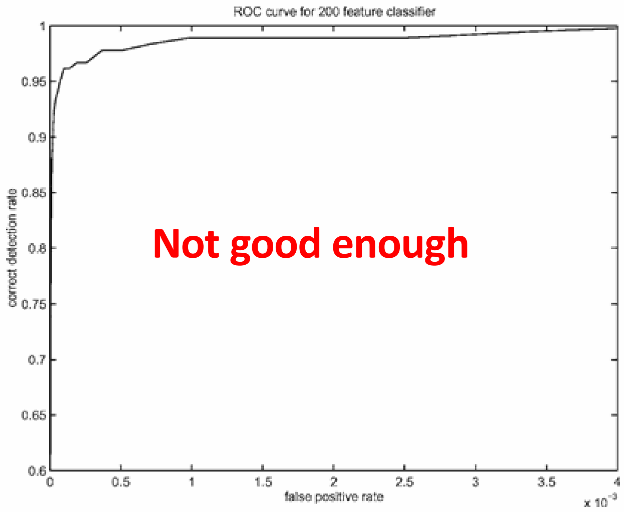
- **Boosting for face detection**
  - The first two features selected by the boosting method:



  - This combination results in 100% face detection and 50% false positives

Face detection

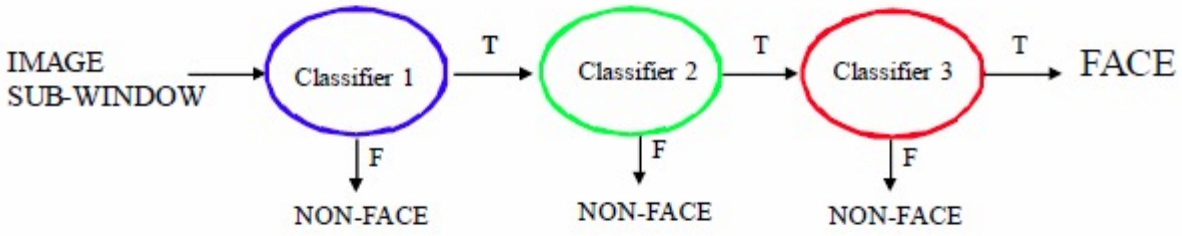
- **Boosting for face detection**
  - A classifier containing 200 features gives 95% accuracy and a false positive rate of 1 out of 14084



ROC-curve (receiver operating characteristic)

Face detection

- **Cascade**
  - Start with simple classifiers that reject some of the negative windows while accept almost all of the positive windows
  - A positive response from the first classifier starts the calculation of the second, more complex classifier, and so on
  - A negative response at any stage results in an immediate rejection of the window



Face detection

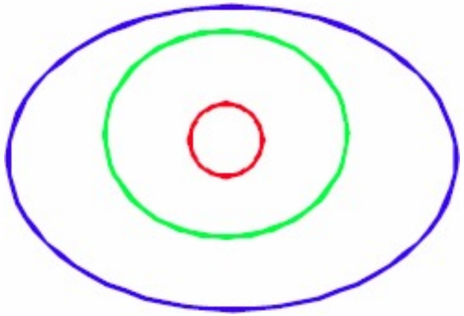
- **Cascade**

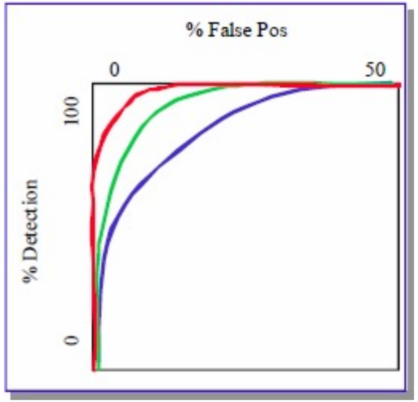
  - Slow classifiers are applied only to some windows that speeds up the whole process
  - Classifier Complexity / Speed Control:
    - Number of support vectors
    - Number of features
    - SVM kernel

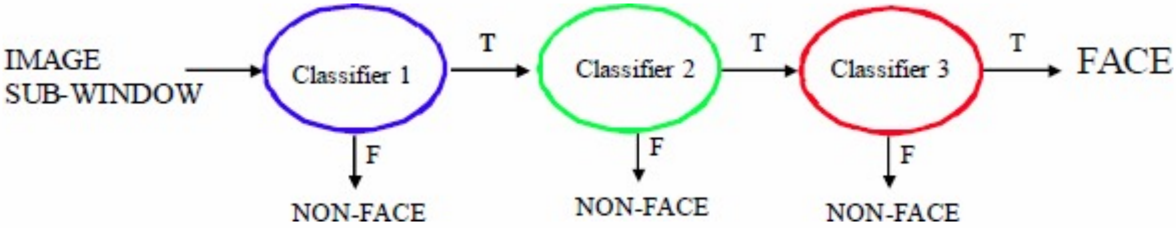
Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001. Vol. 1. Ieee, 2001.
Vedaldi, Andrea, et al. "Multiple kernels for object detection." 2009 IEEE 12th international conference on computer vision. IEEE, 2009.

Face detection

- **Cascade**
  - The classifiers becomes more complex with each next cascade level, so the error of the second type is constantly decreasing







Face detection

- **Cascade**
  - A cascade detection rate of 0.9 and a false positive rate of 10^{-6} is achieved with a cascade of 10 stages if at each stage:
    - The detection rate is about equal to $0{,}99$ $(0{,}99^{10} \approx 0{,}9)$
    - The false positive rate is about equal to $0{,}30$ $(0{,}3^{10} \approx 6\times10^{-6})$

Face detection

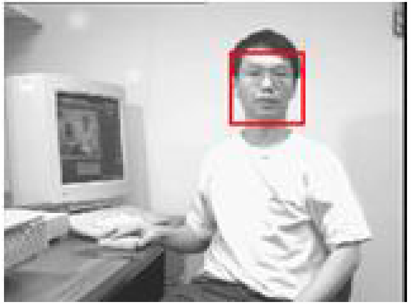
- **Cascade training**
  - Set the required values for the detection level and the false positive ratios for each cascade stage
  - Add features until the parameters of the current stage reach the specified level:
    - Have to lower the AdaBoost threshold level to maximize the detection rate (as opposed to minimizing total classification error)
    - Test on a separate validation set
  - If the overall false positive rate is not low enough, add another feature step
  - False detections in the current stage are used as negative samples in the next stage

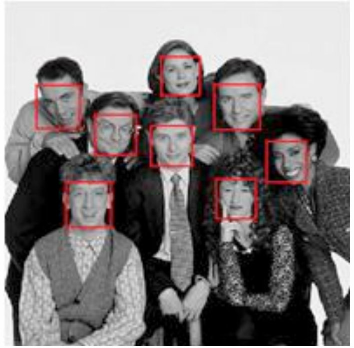
Face detection

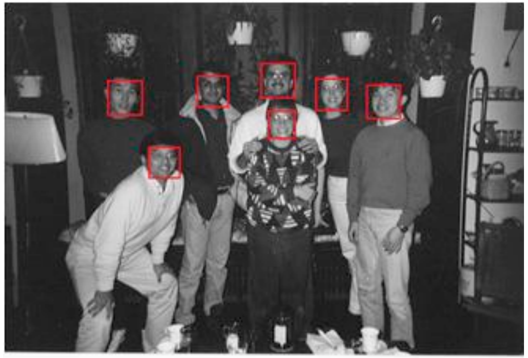
- **Training set**
  - 5000 faces
    - All shoot from front, scaled to 24x24 pixels
    - All normalized
  - 300M false samples
    - 9500 images without faces
  - High variety
    - Different people
    - Different illumination
    - Different face positioning

Face detection

- **Examples**

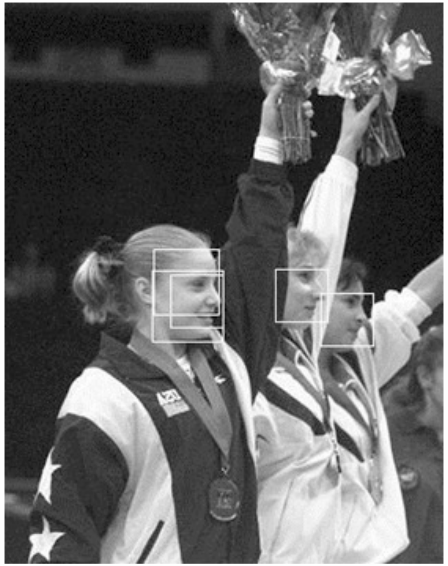






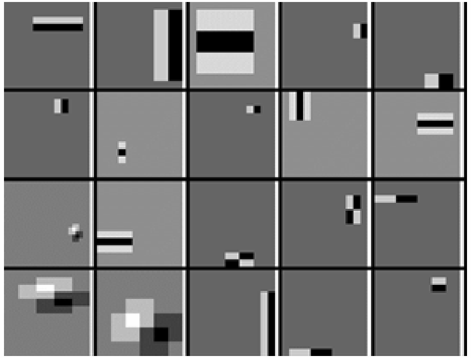
Face detection

- **Face profile detection**



Face detection

- **Profile feature detectors**



Viola-Jones detector

- **Summary**
  - Rectangular features are used as **weak classifiers**
  - **Integral image** is used for fast feature calculation
  - **Boosting** is used for feature selection
  - **Cascade** of classifiers is used for fast rejecting of negative windows

THANK YOU FOR YOUR TIME!

s.shavetov@itmo.ru