Biometric Systems
Lesson 5: More about face localization

Maria De Marsico
demarsico@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Dipartimento di Informatica

A practical example of face localization: the algorithm A by Hsu, Mottaleb and Jain (2002) based on traditional image processing

Two macro phases:

Face candidates detection

Face candidates verification through facial features detection

Algorithm A: detail of the steps

Algorithm main steps:

- **Illumination compensation**
- **Color space transformation**
- Localization based on skin model

- Localization of the main face features
  - eyes
  - mouth
  - face contour

Algorithm A: Illumination compensation

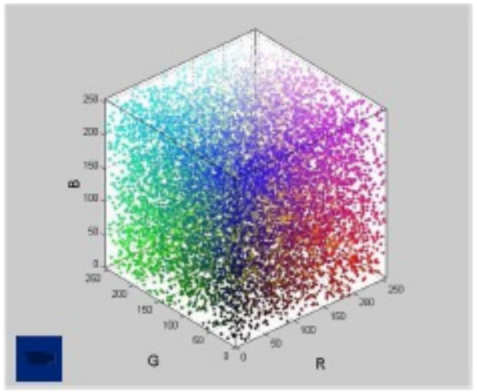
Why: the skin-tone depends on the scene overall illumination

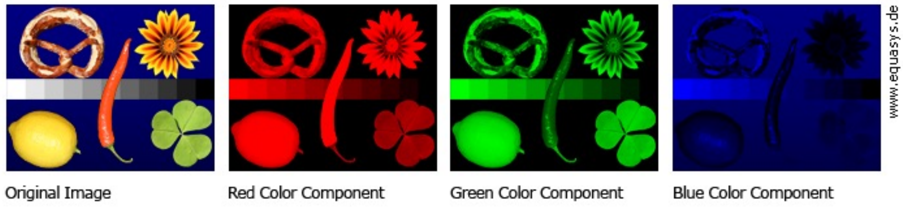
How:

- the illumination compensation uses «reference white» to normalize color appearance
  - Reference white = pixels with the top 5 percent of **luma** values (if they are in a sufficient number with respect to the image size)
    - Luma is the weighted sum of gamma-compressed RGB components (denoted as R'G'B') → $V_{out}=V_{in}^{\gamma}$ with $\gamma < 1$
- If a sufficient number of reference white is found, or the average color is not similar to skin tone, the RGB components are adjusted so that the average grey level of the reference white is linearly scaled to 255

Algorithm A: color space transformation

- RGB is not a perceptually uniform space = colors which are close to each other in RGB space may not be perceived as similar
- Skin model: a set of close colors (cluster) within the color space
- It is advisable to perform the detection in a different space

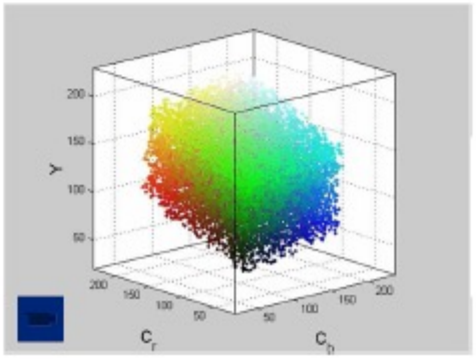


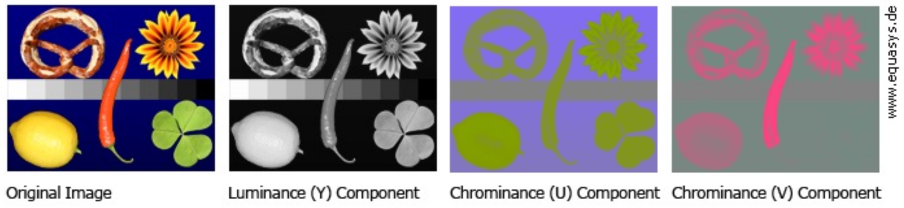


Original Image Red Color Component Green Color Component Blue Color Component

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256} \begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

http://www.equasys.de/colorformat.html





Original Image Luminance (Y) Component Chrominance (U) Component Chrominance (V) Component

Y is the luminance component
C_B and C_R are the blue-difference and red-difference chroma components (similar to UV in YUV)

Algorith A: localization

- Localization based on skin model
- Variance-based segmentation
- Connected components

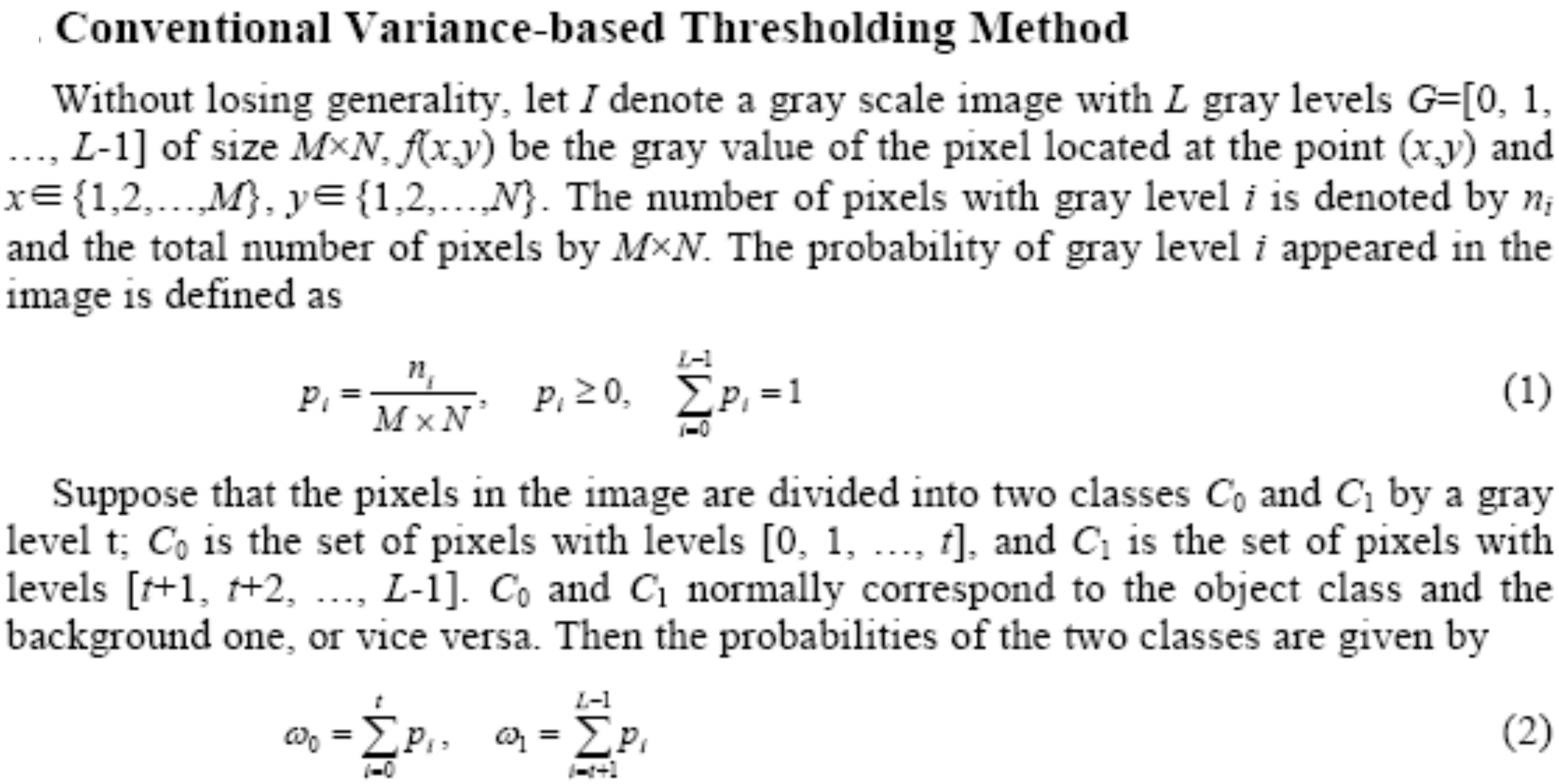
Algoritm A: variance-based segmentation (1)

- The simplest method of image segmentation is called the thresholding method.

- The key of this method is to select the threshold value (or values when multiple-levels are selected).

- Several popular methods are used in industry including the maximum entropy method, Otsu's method (maximum variance), and k-means clustering.

Algoritm A: variance-based segmentation (2)



Conventional Variance-based Thresholding Method

Without losing generality, let I denote a gray scale image with L gray levels $G=[0, 1, \ldots, L\text{-}1]$ of size $M\times N$, $f(x,y)$ be the gray value of the pixel located at the point (x,y) and $x\in\{1,2,\ldots,M\}$, $y\in\{1,2,\ldots,N\}$. The number of pixels with gray level i is denoted by n_i and the total number of pixels by $M\times N$. The probability of gray level i appeared in the image is defined as

$$p_i = \frac{n_i}{M\times N}, \quad p_i \geq 0, \quad \sum_{i=0}^{L-1} p_i = 1 \tag{1}$$

Suppose that the pixels in the image are divided into two classes C_0 and C_1 by a gray level t; C_0 is the set of pixels with levels $[0, 1, \ldots, t]$, and C_1 is the set of pixels with levels $[t+1, t+2, \ldots, L\text{-}1]$. C_0 and C_1 normally correspond to the object class and the background one, or vice versa. Then the probabilities of the two classes are given by

$$\omega_0 = \sum_{i=0}^{t} p_i, \quad \omega_1 = \sum_{i=t+1}^{L-1} p_i \tag{2}$$

Algoritm A: variance-based segmentation (3)

The mean gray levels of the two classes can be defined as

$$\mu_0 = \sum_{i=0}^{t} i p_i / \omega_0 , \quad \mu_1 = \sum_{i=t+1}^{L-1} i p_i / \omega_1 \tag{3}$$

and corresponding class variances are given by

$$\sigma_0^2 = \sum_{i=0}^{t} (i - \mu_0)^2 p_i / \omega_0 , \quad \sigma_1^2 = \sum_{i=t+1}^{L-1} (i - \mu_1)^2 p_i / \omega_1 \tag{4}$$

the within-class variance in Otsu method is defined by

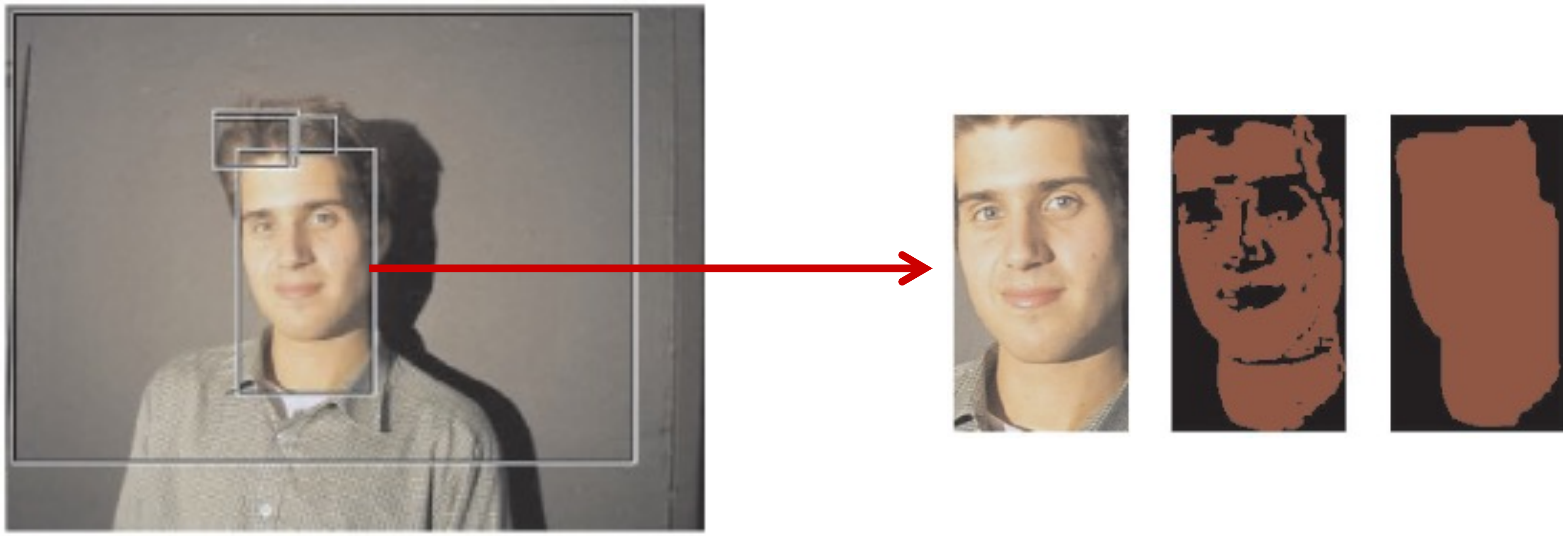
$$\sigma_w^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2 \tag{5}$$

the optimal threshold t^* can be determined by

$$t^* = \arg\min_{t \in G} [\sigma_w^2(t)] \tag{6}$$

Algoritm A: Connected components

The detected skin tone pixels are iteratively segmented using local color variance → connected components →grouped according to spatial closeness and similar color →face candidates



Images are from: R.L. Hsu, M.A. Mottaleb, A.K. Jain, “Face Detection in Color Images”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 696-706, 2002.

Algorithm A: eye localization (1)

The algorithm builds two different eye maps (**chroma** and **luma**)

- *Chrominance map*: its creation relies on the observation that the region around eyes is characterized by high values of C_b and low values of C_r:

$$EyeMapC = \frac{1}{3}\left\{ \left(C_b^2\right) + \left(\tilde{C}_r^2\right) + \left(\frac{C_b}{C_r}\right) \right\} \qquad \tilde{C}_r = 255 - C_r$$

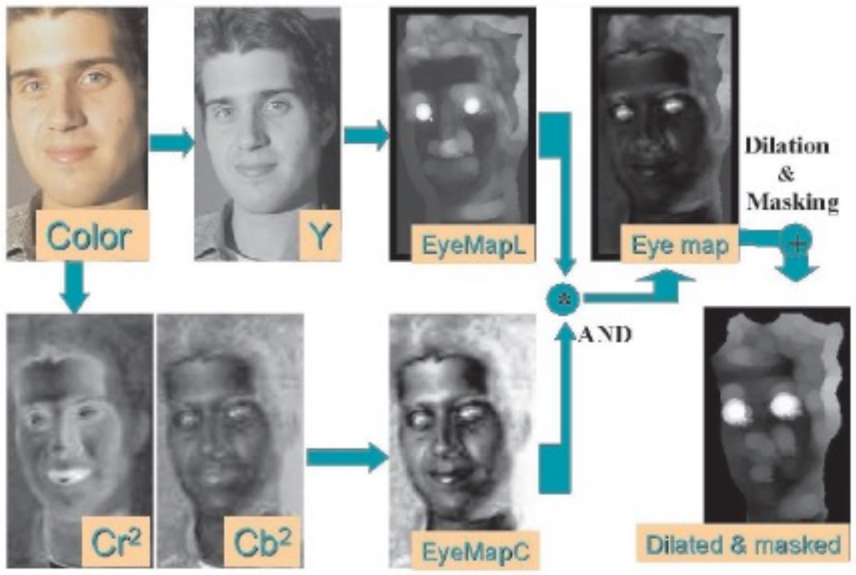
where C_b^2, $(\tilde{C}_r)^2$, C_b/C_r all are normalized to the range $[0, 255]$

- *Luminance map*: eyes usually contain both light and dark zones that can be highlighted by morphological operators (dilation and erosion with hemispheric structuring elements)

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1}$$

Algorithm A: eye localization (2)

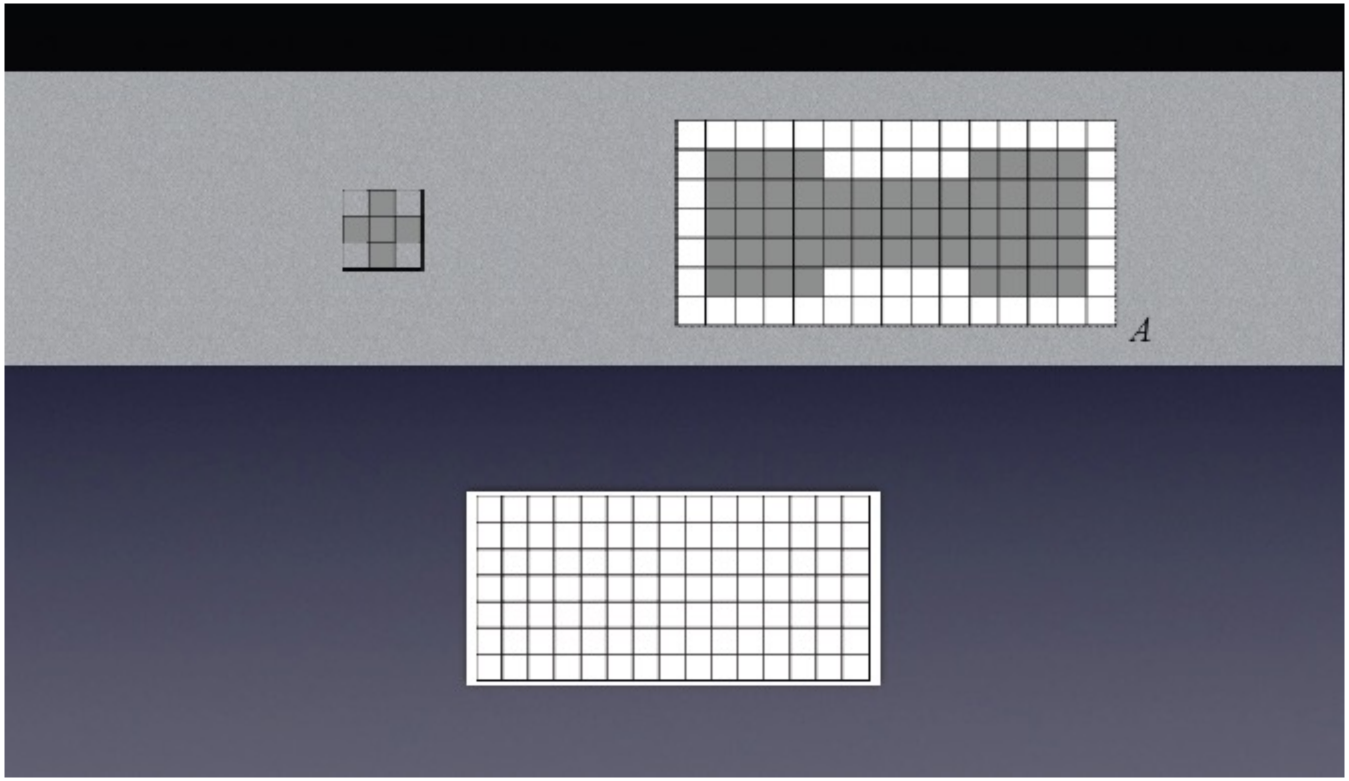
- Chroma map is enhanced by histogram equalization
- The two maps are combined through AND operator
- The resulting map undergoes dilation, masking and normalization to discard the other face regions and brighten eyes.
- Further operations allow to refine this map.



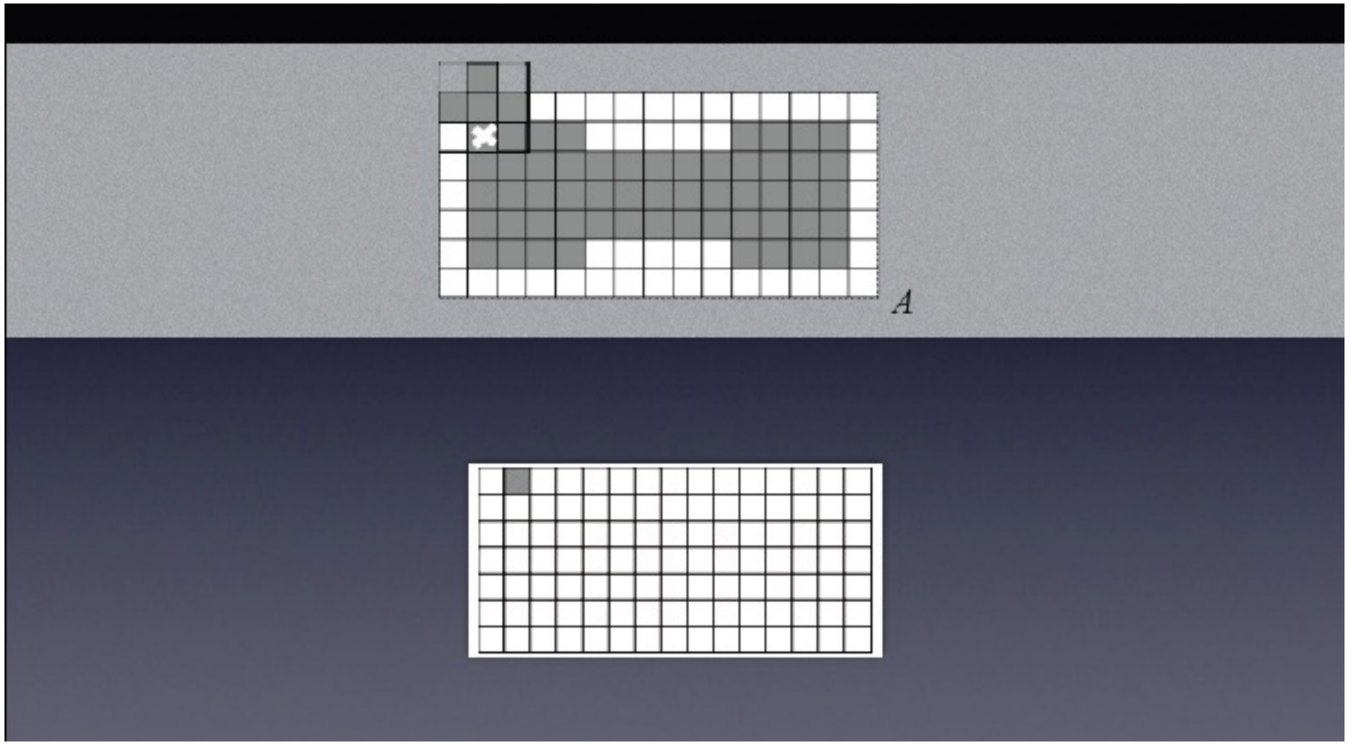
Note: Dilation (1)

- The dilation operator takes two pieces of data as inputs. The first is the image which is to be dilated. The second is a (usually small) set of coordinate points known as a structuring element (also known as a *kernel*). It is this structuring element that determines the precise effect of the dilation on the input image.
- The mathematical definition of dilation for *binary* images is as follows:
- Suppose that X is the set of Euclidean coordinates corresponding to the input binary image, and that K is the set of coordinates for the structuring element. Let Kx denote the translation of K so that its center is at x.
- Then the dilation of X by K is simply the set of all points x such that the intersection of Kx with X is non-empty.

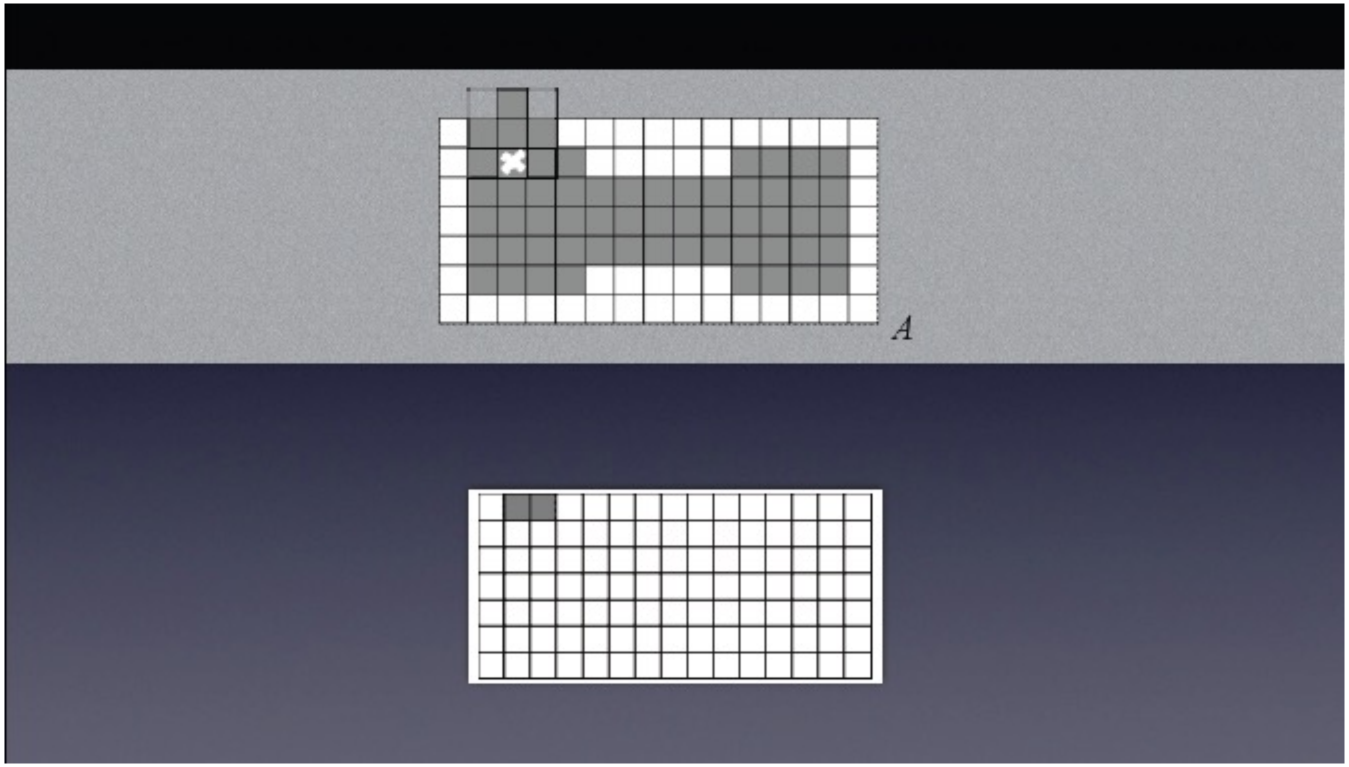
Note: Dilation (2)



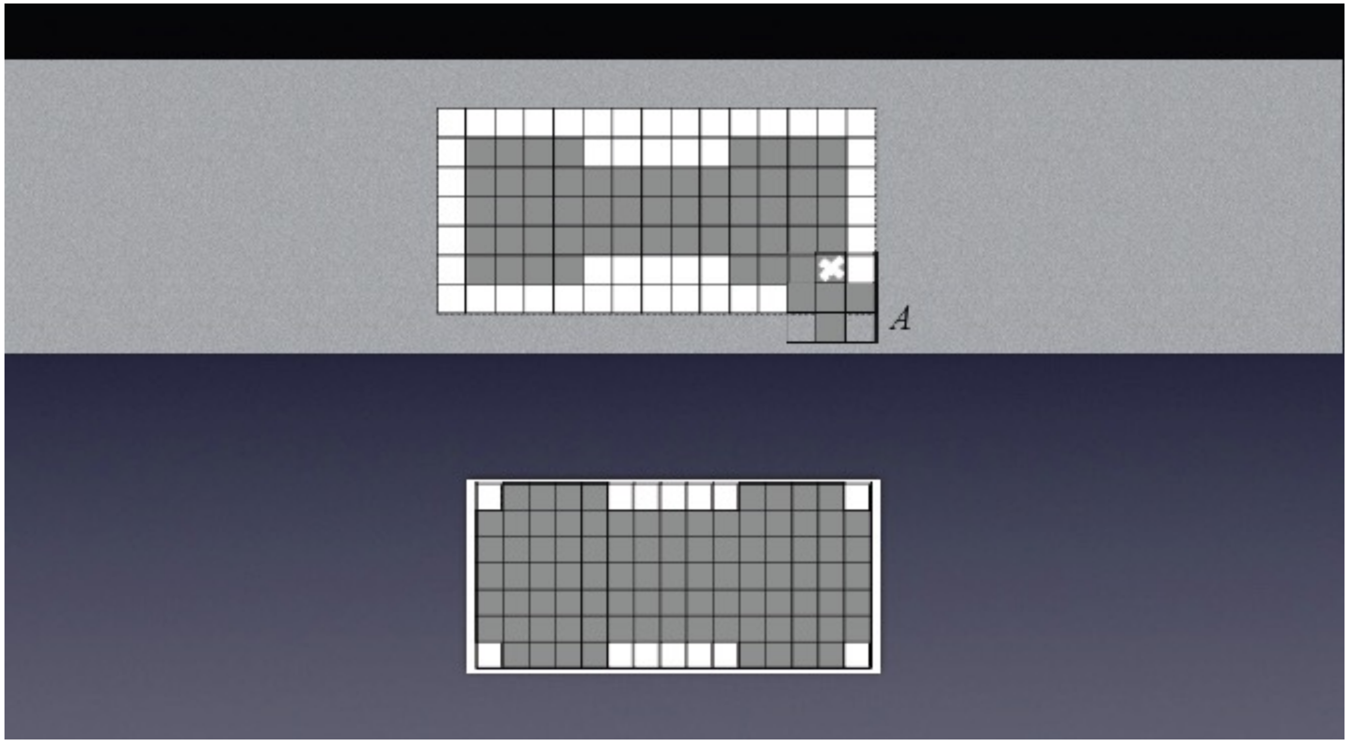
Note: Dilation (2)



Note: Dilation (2)



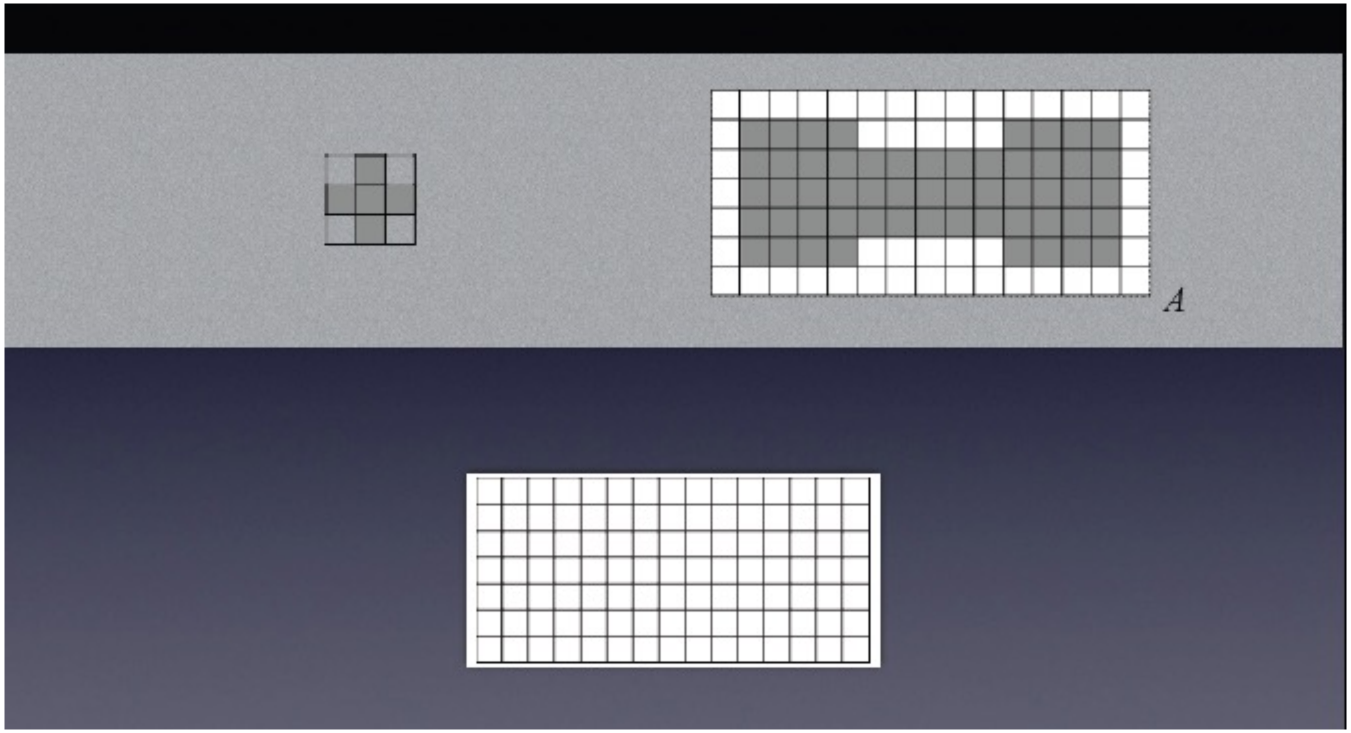
Note: Dilation (2)



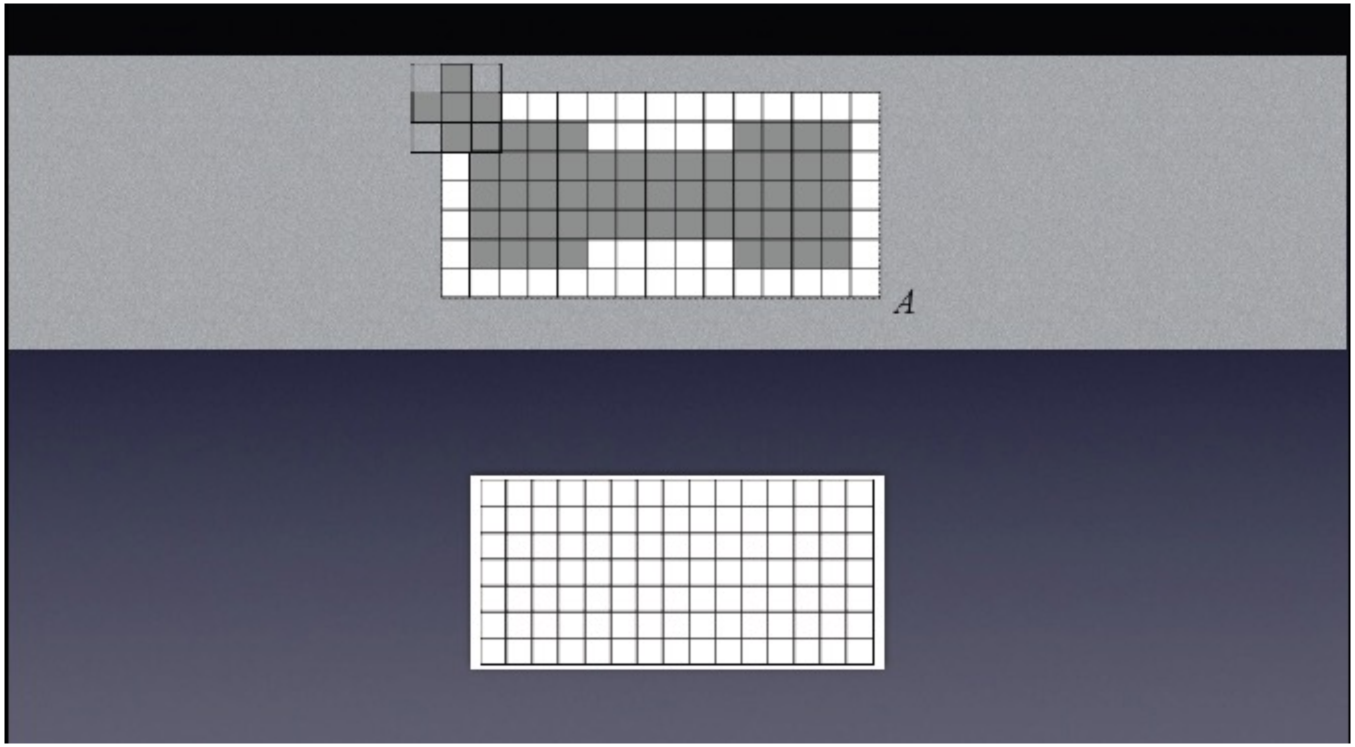
Note: Erosion (1)

- The erosion operator takes two pieces of data as inputs. The first is the image which is to be eroded. The second is a (usually small) set of coordinate points known as a structuring element (also known as a *kernel*). It is this structuring element that determines the precise effect of the erosion on the input image.
- The mathematical definition of erosion for *binary* images is as follows:
- Suppose that *X* is the set of Euclidean coordinates corresponding to the input binary image, and that *K* is the set of coordinates for the structuring element. Let *Kx* denote the translation of *K* so that its origin is at *x*.
- Then the erosion of *X* by *K* is simply the set of all points *x* such that *Kx* is a subset of *X*.

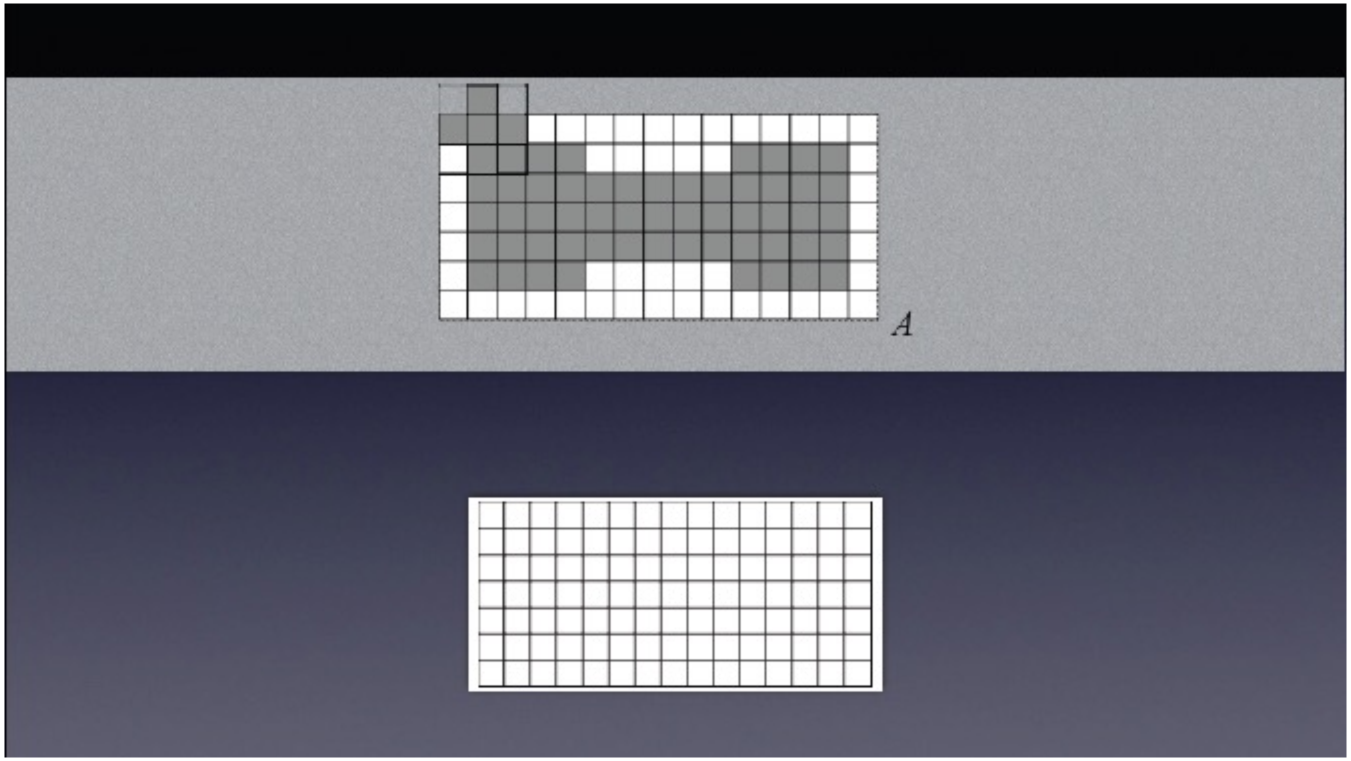
Note: Erosion (2)



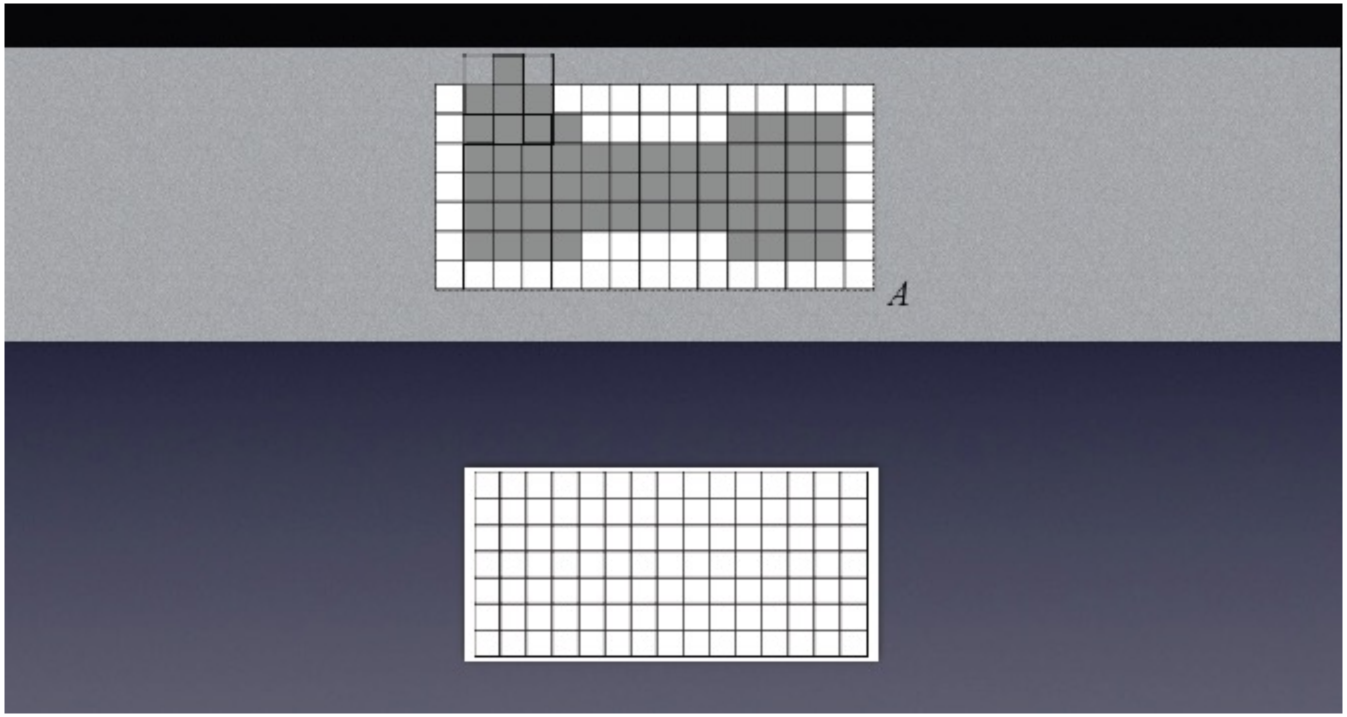
Note: Erosion (2)



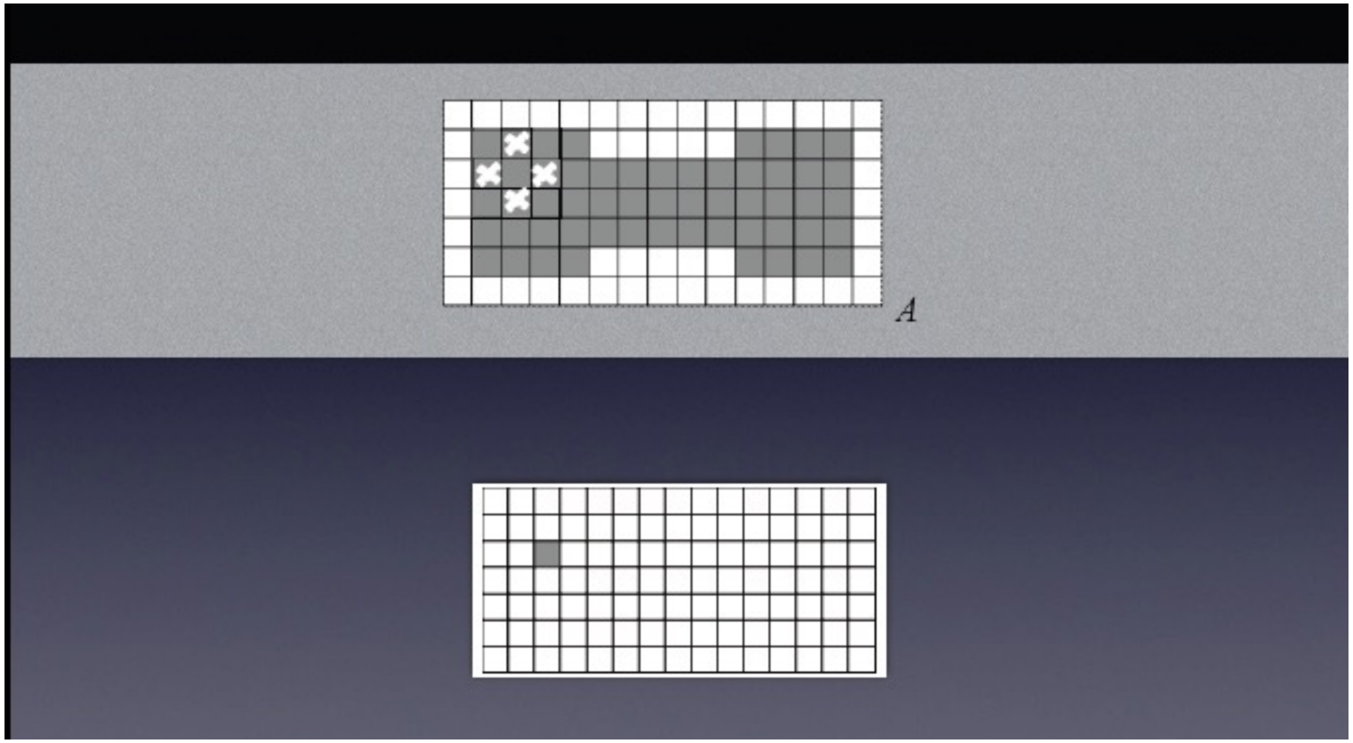
Note: Erosion (2)



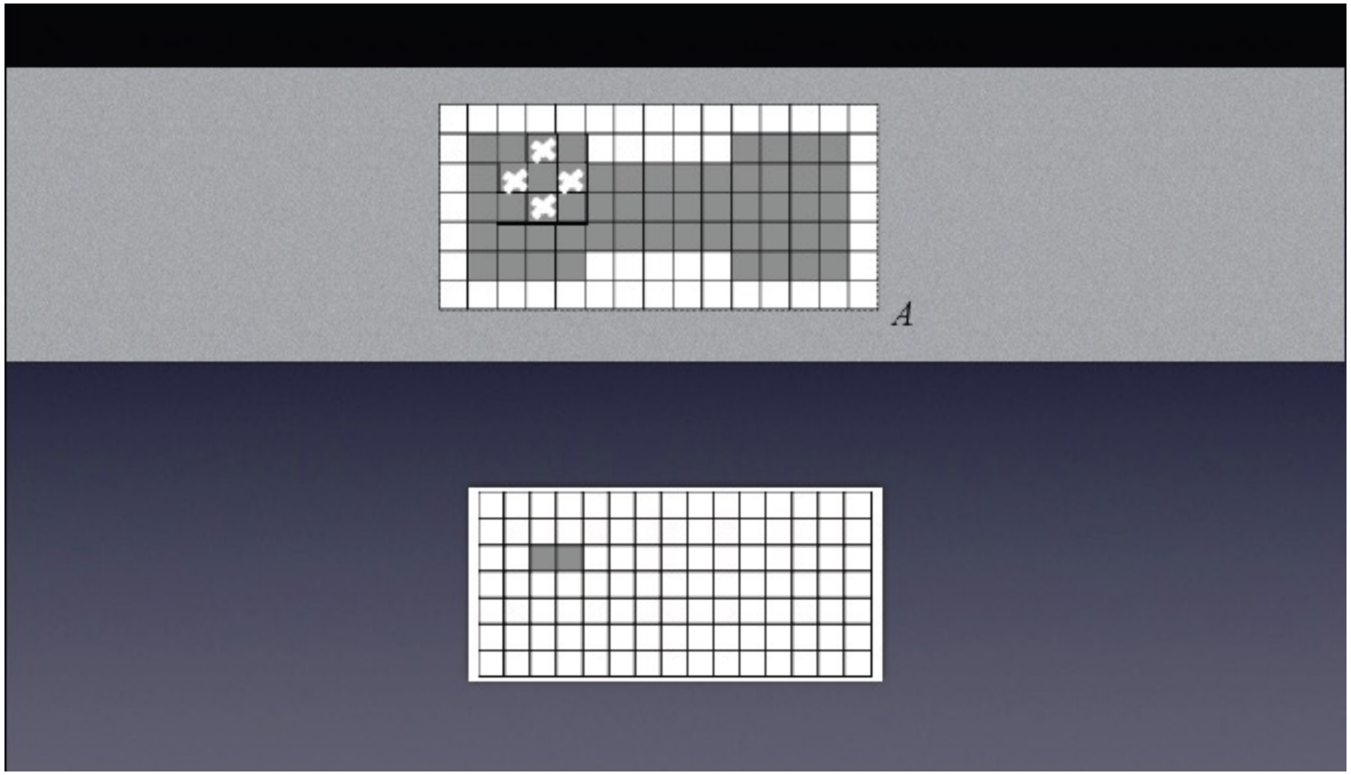
Note: Erosion (2)



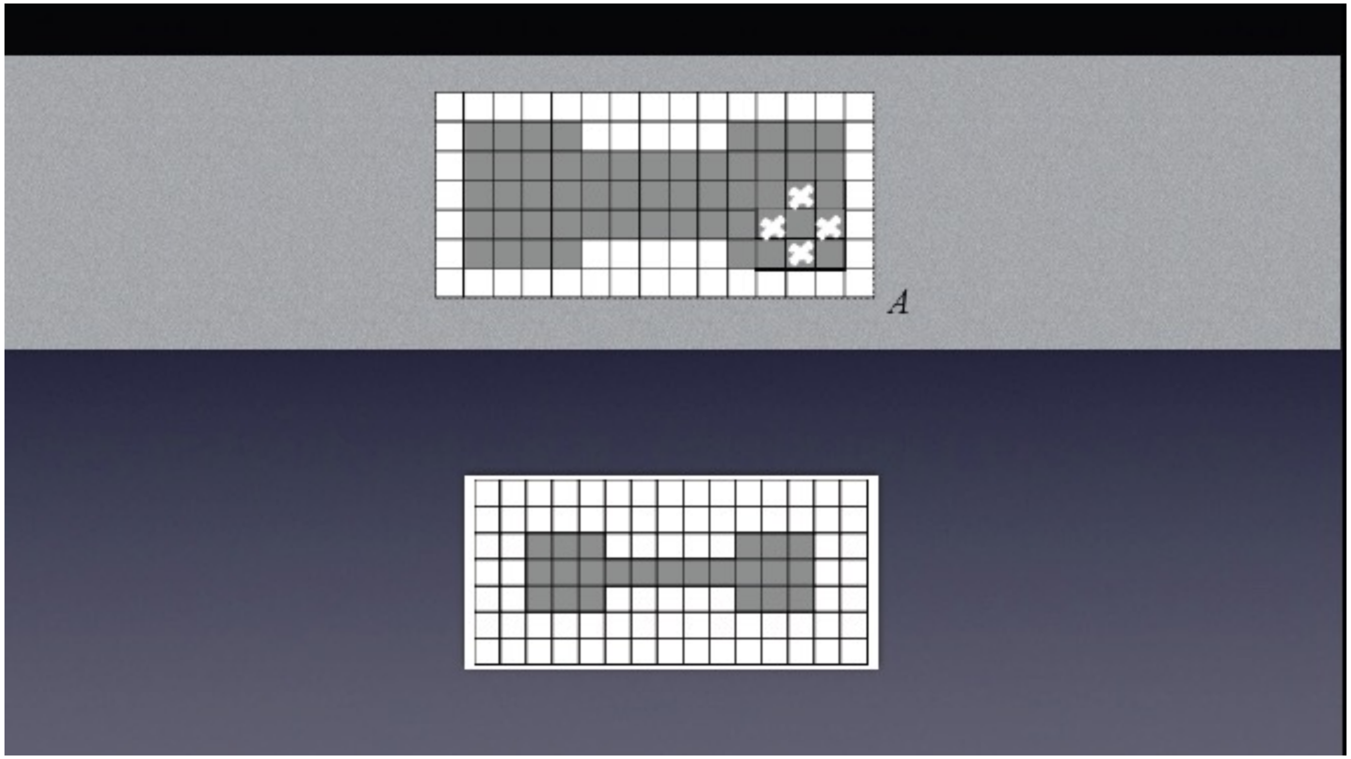
Note: Erosion (2)



Note: Erosion (2)



Note: Erosion (2)



Algorithm A: mouth localization (1)

In the mouth region the C_r component is higher than the C_b one; the response to C_r/C_b is low, while the response to C_r^2 is high:

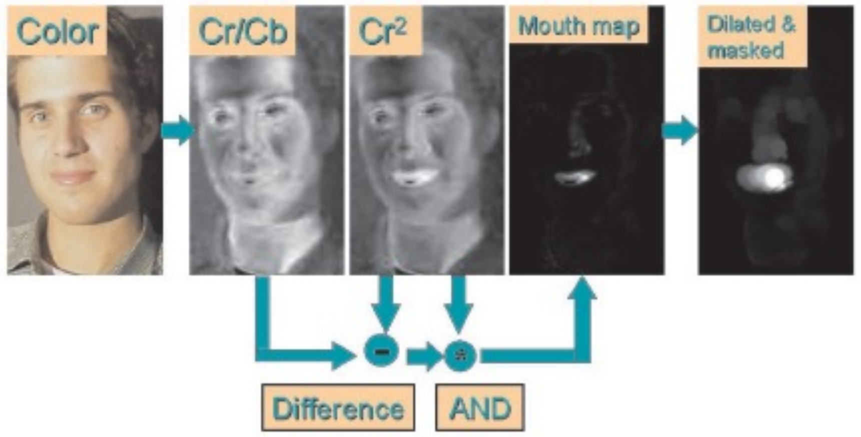
$$MouthMap = C_r^2 \cdot \left(C_r^2 - \eta \cdot C_r/C_b\right)^2$$

$$\eta = 0.95 \cdot \frac{\frac{1}{n} \sum_{(x,y) \in FG} C_r(x,y)^2}{\frac{1}{n} \sum_{(x,y) \in FG} C_r(x,y)/C_b(x,y)}$$

C_r/C_b and C_r^2 are normalized in [0, 255]

n is the number of pixels in the face mask

Algorithm A: mouth localization (2)



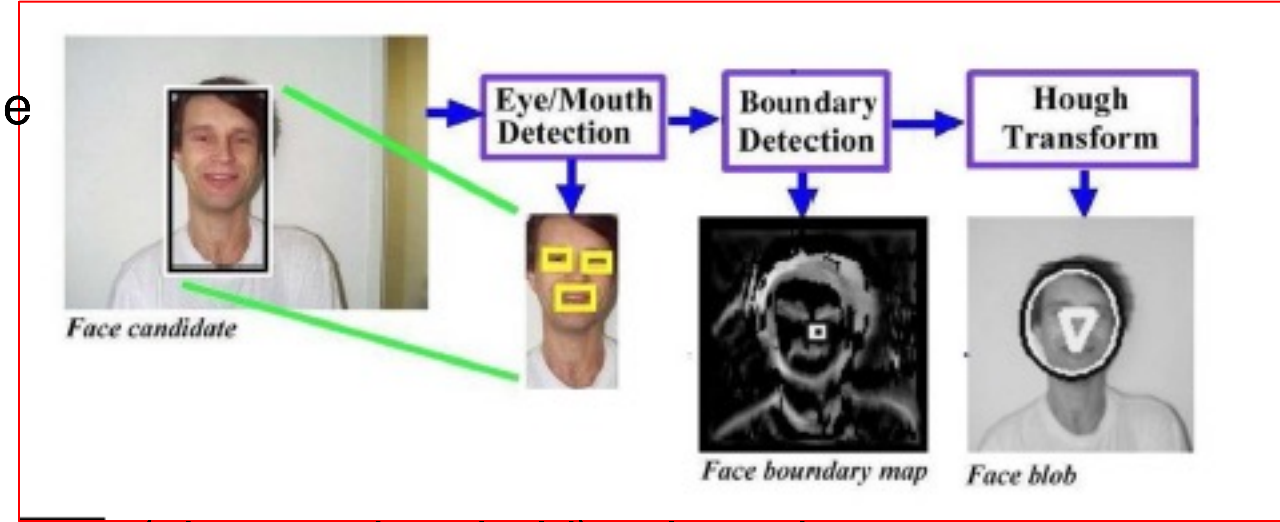
Algorithm A: face contour

The algorithm analyzes all the triangles composed by two candidate eyes and a candidate mouth.

Each triangle is verified by cheking:

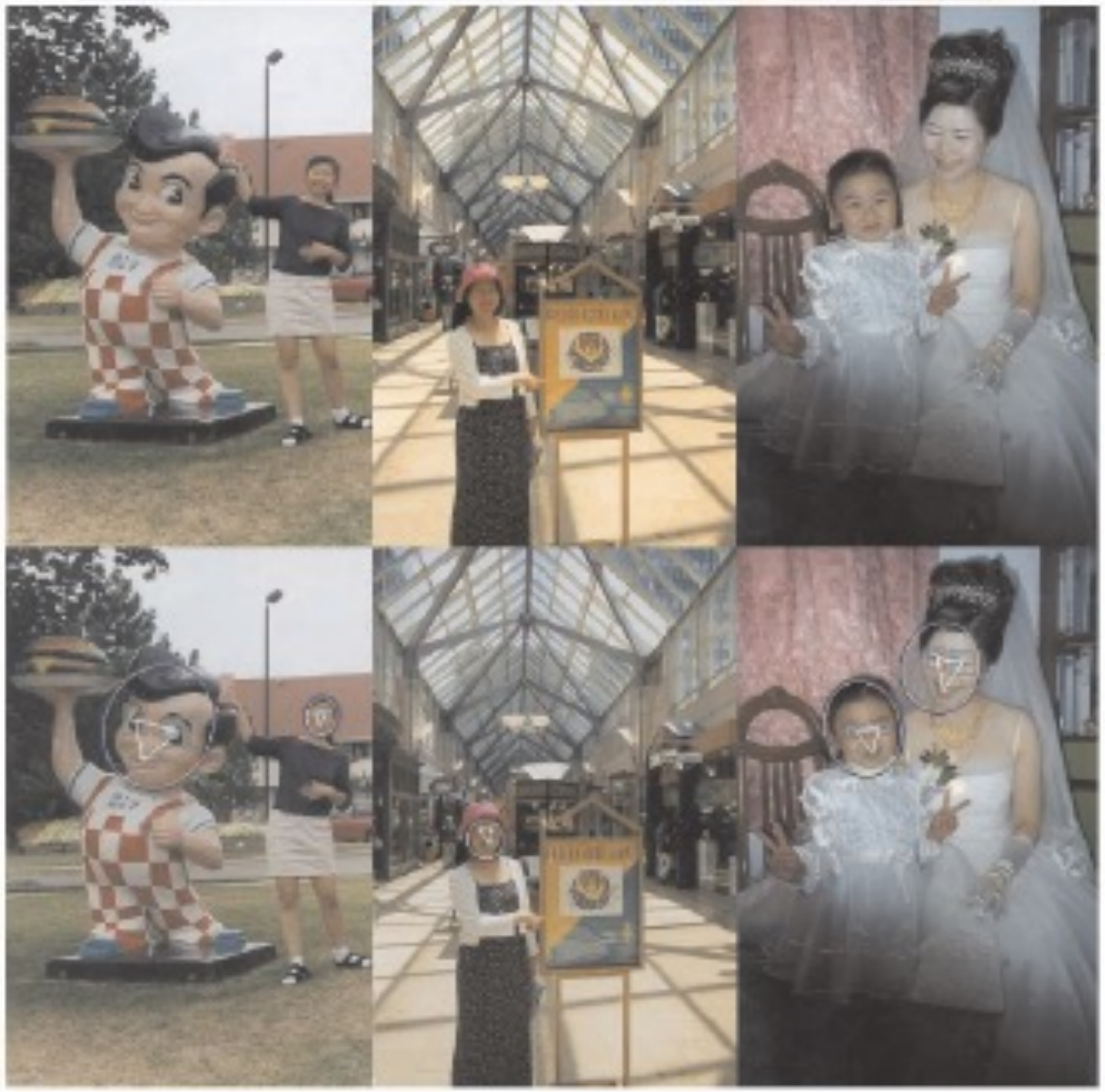
- Luma variations and average of the orientation gradient of the blobs containing eyes and mouth
- Geometry and orientation of the triangle
- Presence of a face contour around the triangle

A score is assigned to each triangle satisfying the conditions, which also considers preference for upright orientation and simmetry.



The triangle with highests score (above a threshold) selected.

Algorithm A: some results



Algorithm B: Viola-Jones (2004) based on a machine learning technique

- Paul Viola and Michael Jones proposed one of the most successful (so far) approaches to object localization (included in OpenCV) (2001).

- The algorithm is image based and can be applied to face detection (but also to eye and mouth detection in a hierarchical strategy).

- The algorithm requires to create a classifier that is initially trained using multiple instances of the class to identify (positive examples), and several instances of images that do not contain any object of the class but may cause an error (negative examples).

- Training is designed to extract several features from the examples and to select the most discriminating ones. The statistical model which is built incrementally contains such information.

- Misses (a present object is not detected) or false alarms (an object is detected but it is not present) can be decreased by retraining adding new suited examples (positive or negative).

Algorithm B: Viola-Jones

- The face classifier is able to associate an input pattern with one of two classes face / non face.

- Training is slow, but detection is very fast
- Key ideas:
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Multiscale detection*

- Localization is performed by a search sliding window (with varying size) over the image. The window is classified as face/non face according to the features extracted from it.

Note: AdaBoost learning procedure

- Boosting is a classification scheme that works by combining M *weak learners (linear classifiers)* to obtain a *strong* one $H_M(x)$ (nonlinear ensemble classifier)
- *Weak learner*: classifier with possibly very low accuracy (just better than chance)

$$H_M(x) = \frac{\sum_{i=1}^{M} \alpha_i h_i(x)}{\sum_{i=1}^{M} \alpha_i}$$

x is a pattern to classify

$h_i(x) \in \{-1,+1\}$ are the weak classifiers

$\alpha_i \geq 0$ are the relative weights

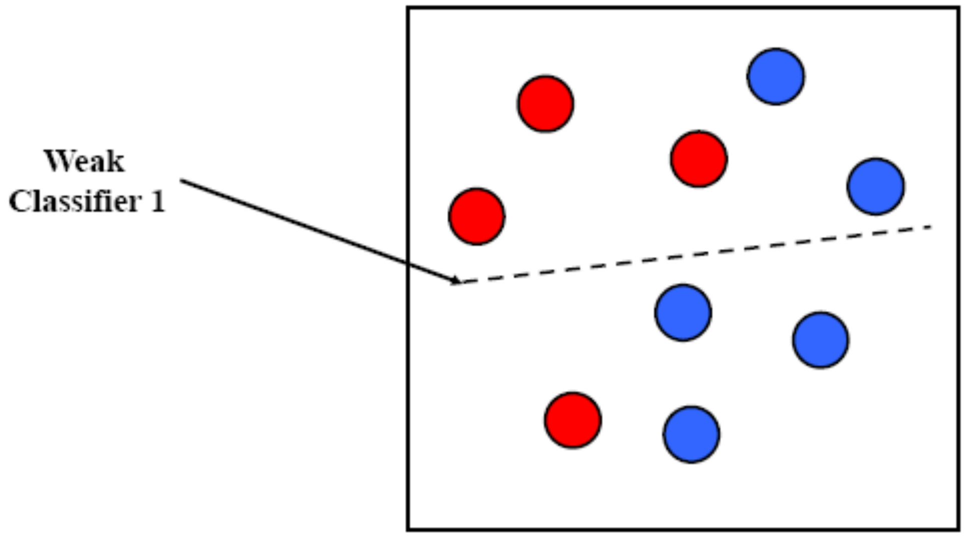
$\sum_{i=1}^{M} \alpha_i$ is a normalization factor

$$H_M(x) = \frac{\sum_{i=1}^{M} \alpha_i h_i(x)}{\sum_{i=1}^{M} \alpha_i}$$

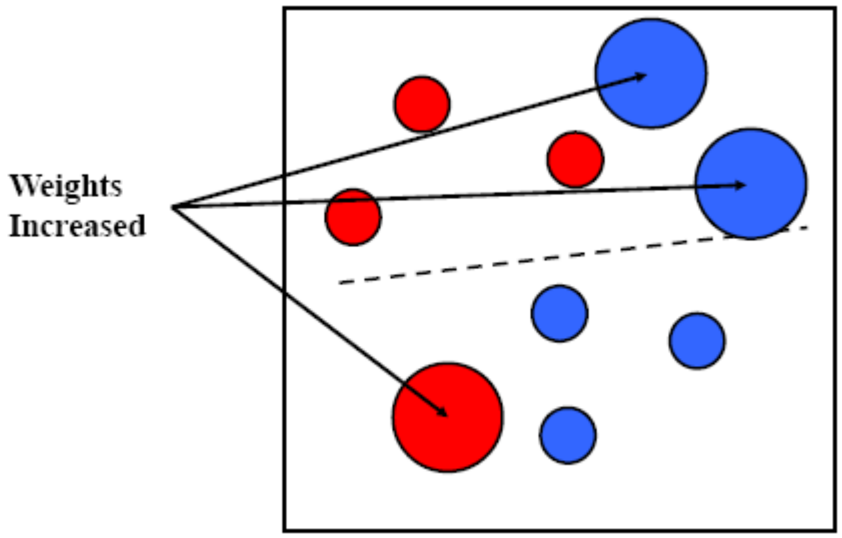
Note: AdaBoost learning procedure

- AdaBoost (Adaptive Boost) is a training technique that has the purpose of learning the best sequence of weak classifiers and their corresponding weights.
- It requires a set of training patterns $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where $y_i \in \{-1, +1\}$ is the class label associated with the pattern $x_i \in \Re^n$.
- During learning a distribution of weights $[w_1, w_2, ..., w_N]$ associated with the training patterns $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ is calculated and updated.
- At the beginning all samples have the same weight (e.g., 1)
- After iteration m, a higher weight $w_i^{(m)}$ is assigned to the patterns that resulted more difficult to classify, so that in the next m+1-th iteration such patterns will receive greater attention.
- Weight updating rules must be devised.
- AdaBoost assumes to have a procedure for learning a weak classifier from a
  set of training patterns, given the distribution $[w_i^{(m)}]$.

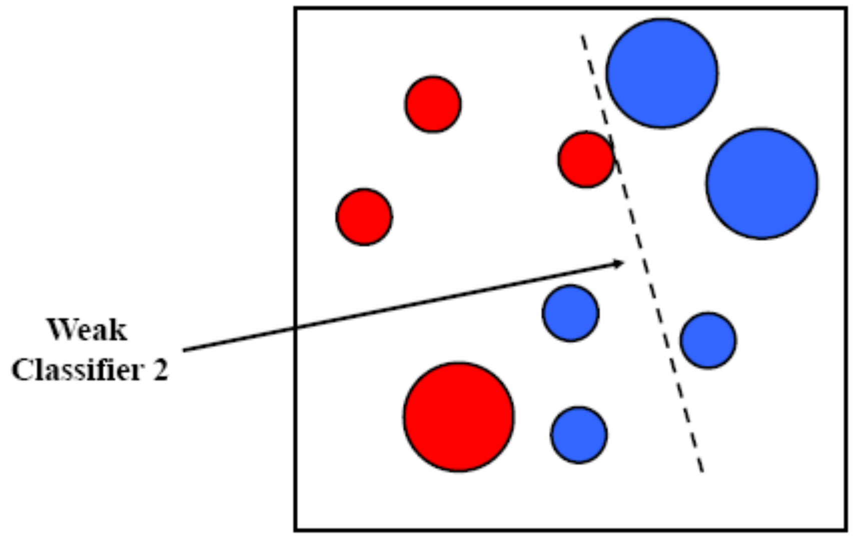
Note: example of AdaBoost



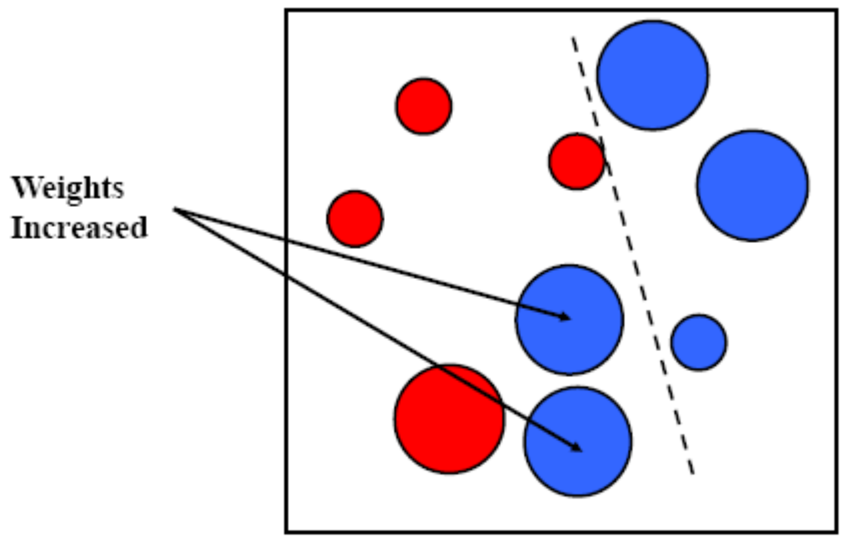
Note: example of AdaBoost



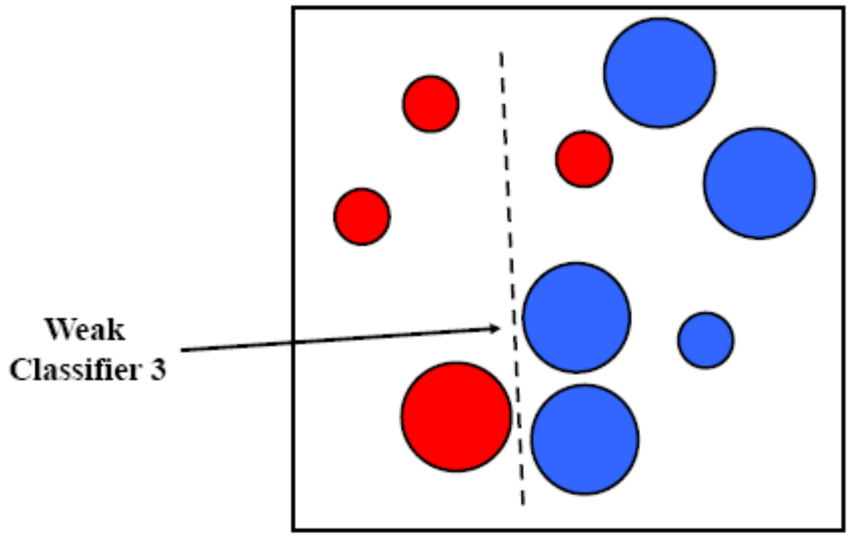
Note: example of AdaBoost



Note: example of AdaBoost

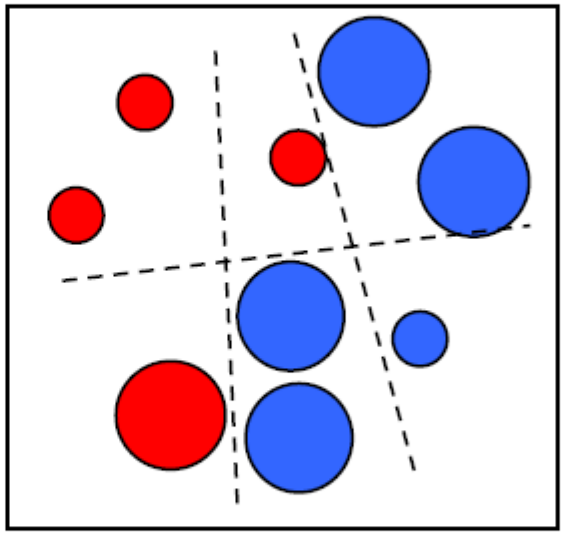


Note: example of AdaBoost



Note: example of AdaBoost

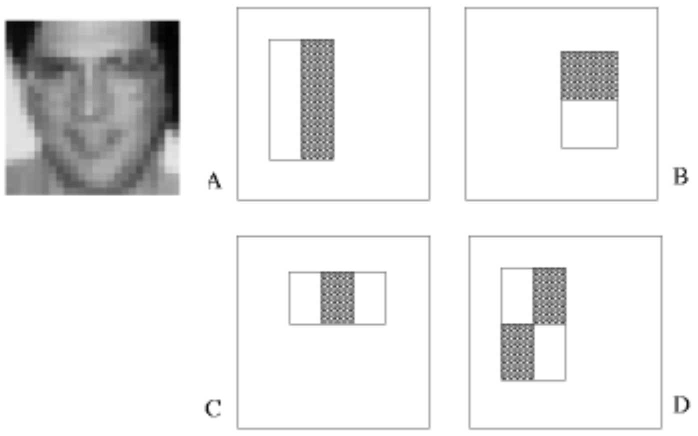
Final classifier is
linear combination of
weak classifiers



Algorithm B: AdaBoost learning procedure for images

- As for images we can define weak learners based on simple (rectangular) features: Haar features

"Rectangle filters"



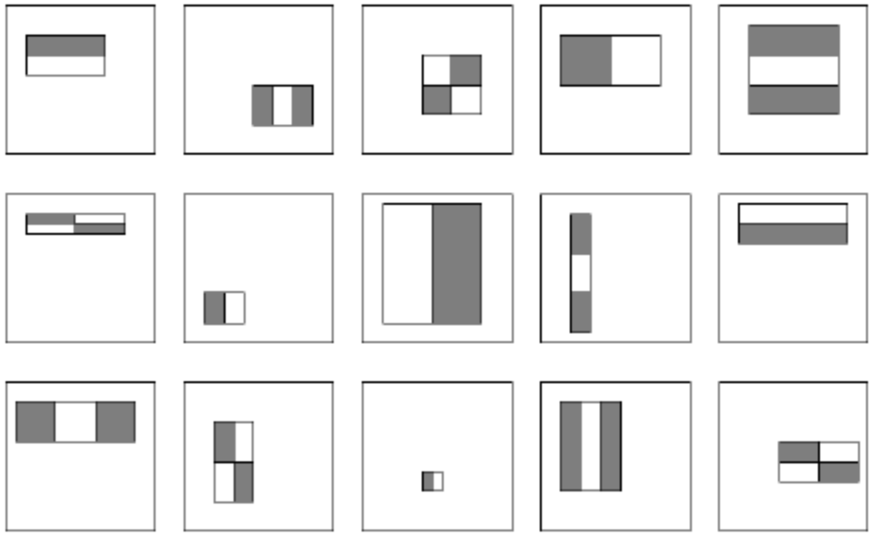
Value =

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

- Each feature is located in a subregion of a subwindow of the image
- Features are applied by modifying their size, shape, and position in the subwindow

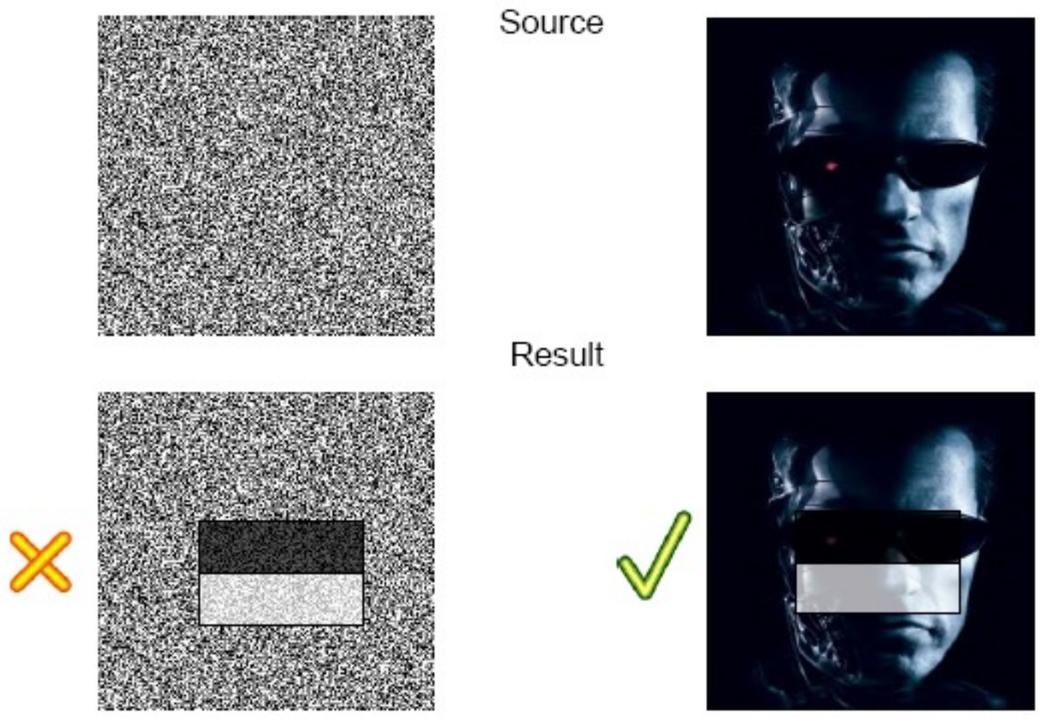
Algorithm B: AdaBoost learning procedure for images

- For a 24x24 detection region, the number of possible rectangle features is ~180.000!



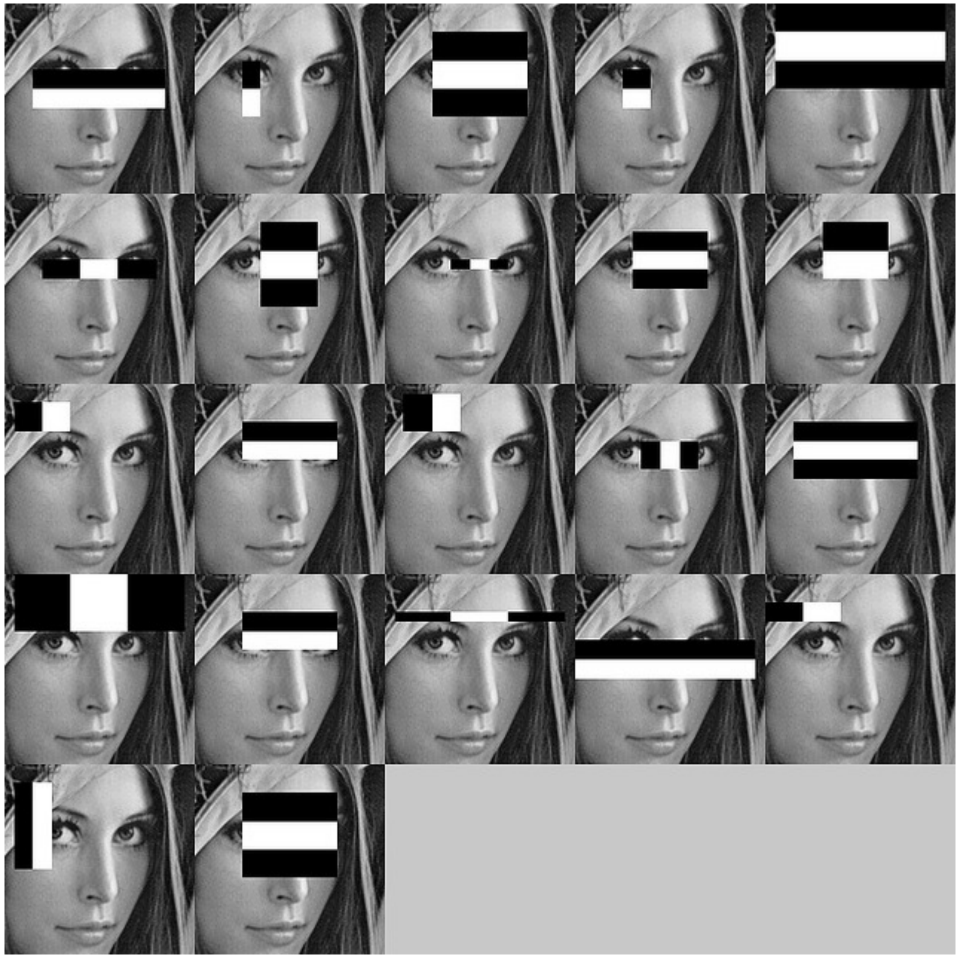
- At test time, it is impractical to evaluate the entire feature set
- We can use AdaBoost to select a small subset of all possible features to build a good classifier.

Example use of rectangular features



From: The Viola/Jones Face Detector. By Robert Fergus

Example use of rectangular features: interesting page



An example of an early stage in the Haar cascade. Each black and white patch represents a feature that the algorithm hunts for in the image.

https://github.com/atduskgreg/Makematics/blob/master/research/haar_detection/index.md

Algorithm B: AdaBoost learning procedure for images

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

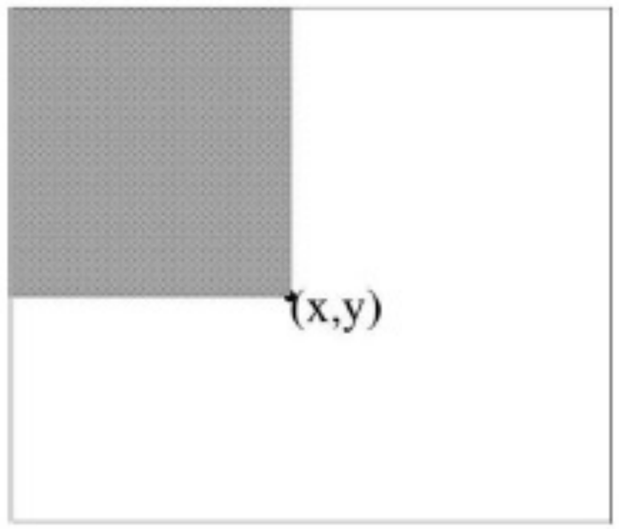
Computational complexity of learning: $O(MNT)$

M filters, N examples, T thresholds

Algorithm B: How to compute Haar features

- Rectangular features can be evaluated through integral images
- The integral image in (x,y), denoted as II(x,y), is the sum of the values of pixels above and on the left of (x,y) computed

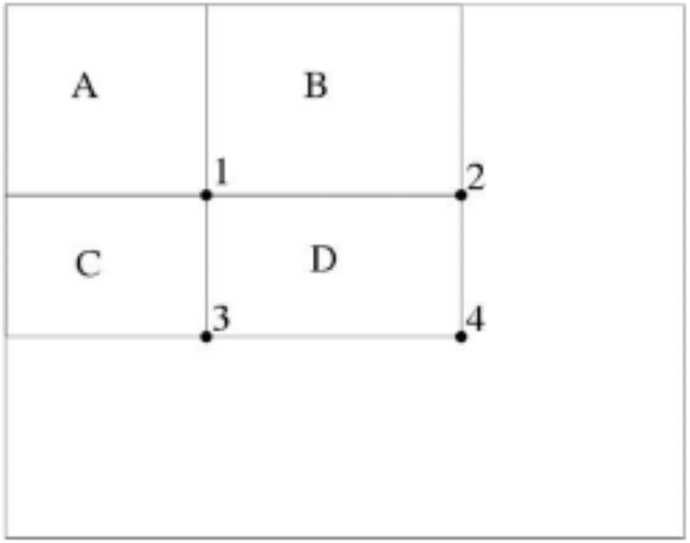
$$II(x,y) = \sum_{x' \le x, y' \le y} I(x',y')$$



$$II(x,y) = \sum_{x' \le x, y' \le y} I(x',y')$$

Algorithm B: How to compute Haar features

- Using integral image it is possible to compute the sum of the values of pixels in any rectangle.
- Example :



Sum of pixels in rectangle D

II(4)– II(2) – II(3) + II(1)

Algorithm B: the weak classifier

- The simplest classifier id a decision tree with one node
- Let's suppose we have built M-1 weak classifiers $\{h_m(x)|m=1,..,M-1\}$ and that we want to build $h_M(x)$.

- This new classifier compares the value of a feature z_{k^*} with a fixed threshold t_{k^*} and assigns the values +1 or -1 accordingly:

$$
\begin{aligned}
h_M(x) &= +1 \text{ if } z_{k^*} > t_{k^*} \\
&= -1 \text{ otherwise}
\end{aligned}
$$

- $h_M(x)$ depends from the two parameters z_{k^*} and t_{k^*}

Algorithm B: the weak and strong classifiers

- $h_M(x)$ depends from the two parameters z_{k^*} and t_{k^*}
- The two parameters can be fixed based on the minimum classification error.

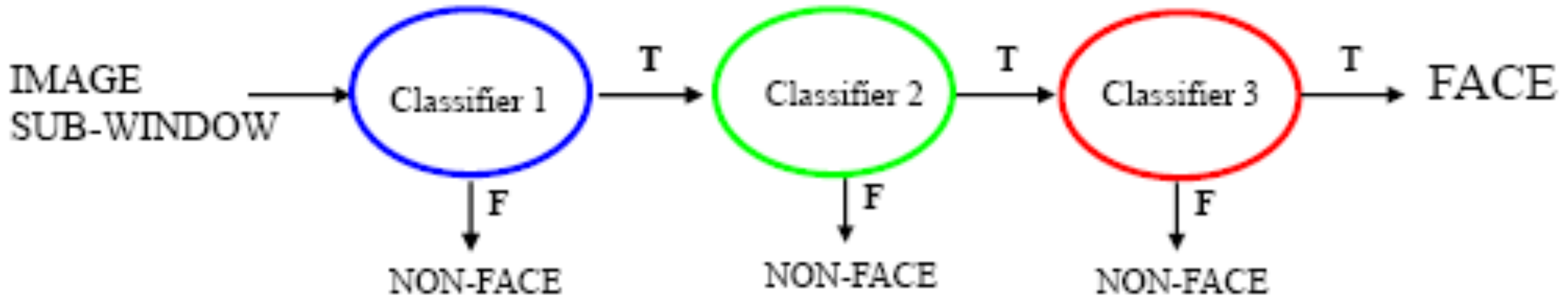
- For each feature z_k we choose the threshold value t_k which minimizes the classification error

- The feature $\mathbf{z}_{k^*}$ which is chosen for the current classfier is the one allowing to obtain the lower error

- AdaBoost learns a sequence of weak classifiers h_m and combines them in a robust classifier H_M, by minimizing the upper bound for the classification error of H_M

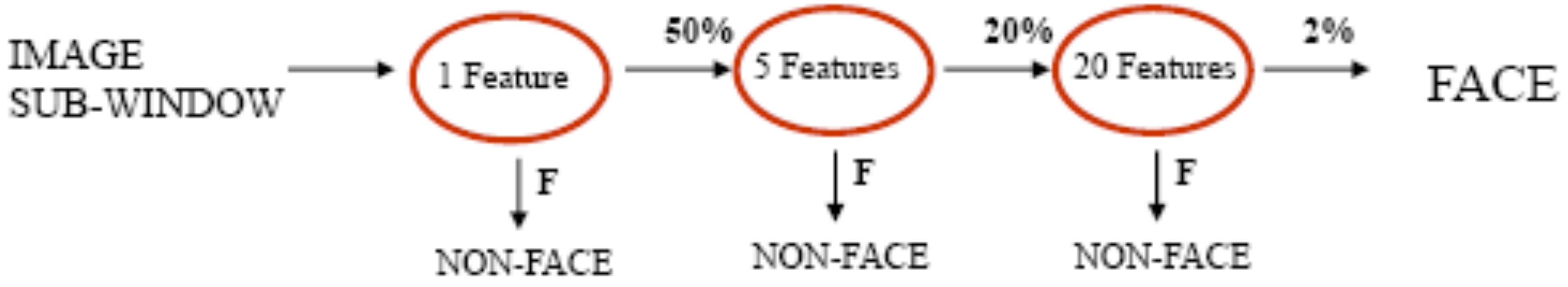
Algorithm B: Cascading classifiers

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive results 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



Algorithm B: how to train the cascade

- Adjust weak learner threshold to minimize *false negatives* (as opposed to total classification error)
- Each classifier trained on false positives of previous stages
- Assume that a single-feature classifier achieves 100% detection rate and about 50% false positive rate
- A five-feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
- A 20-feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

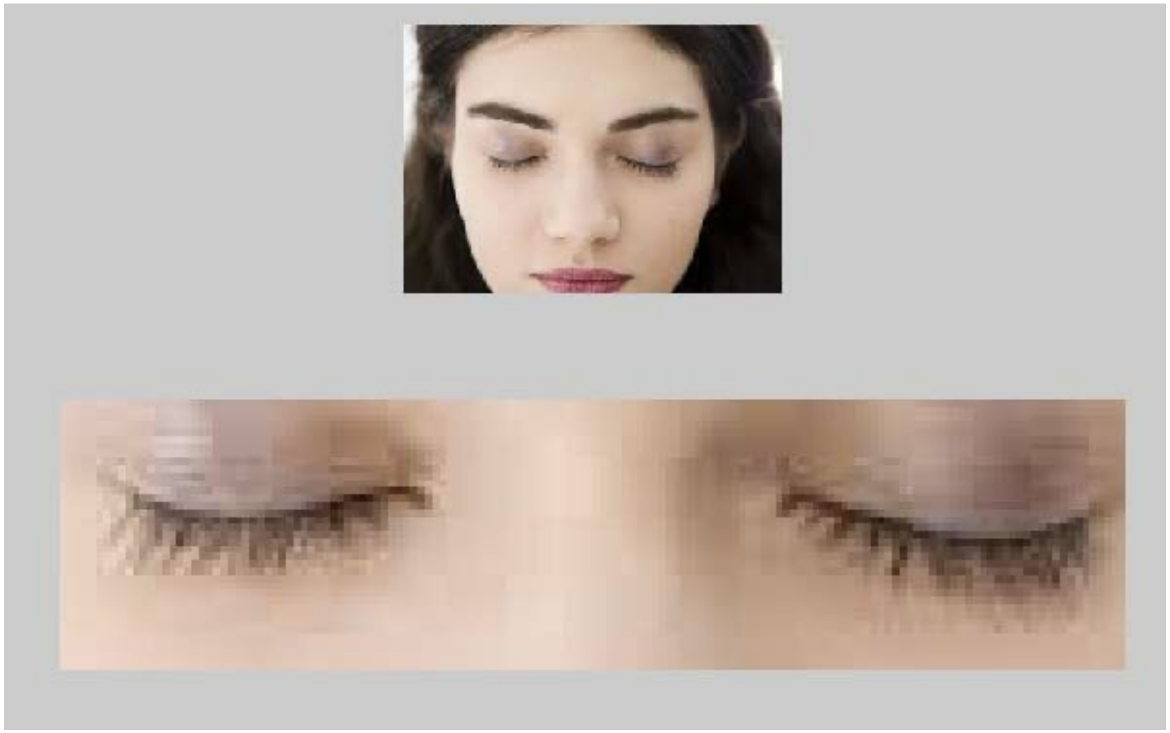


Algorithm B: how it works

- The localization of faces is done by analyzing consecutive sub-windows (overlapping) of the image as input and evaluating for each of them if it belongs to the class of faces.
- The locator of Viola and Jones has been one of the most robust and efficient in the past state of the art
  - The training is very slow (it can take days)
  - The locating procedure is very efficient (real-time operation).
- It is still available in OpenCV

Algorithm B: hierarchical region detection

- It is possible to run the algorithm with models trained on eyes, nose and mouth to detect the single face regions after having identified the face ROI (Region of Interest)



Recent advances: deep approaches

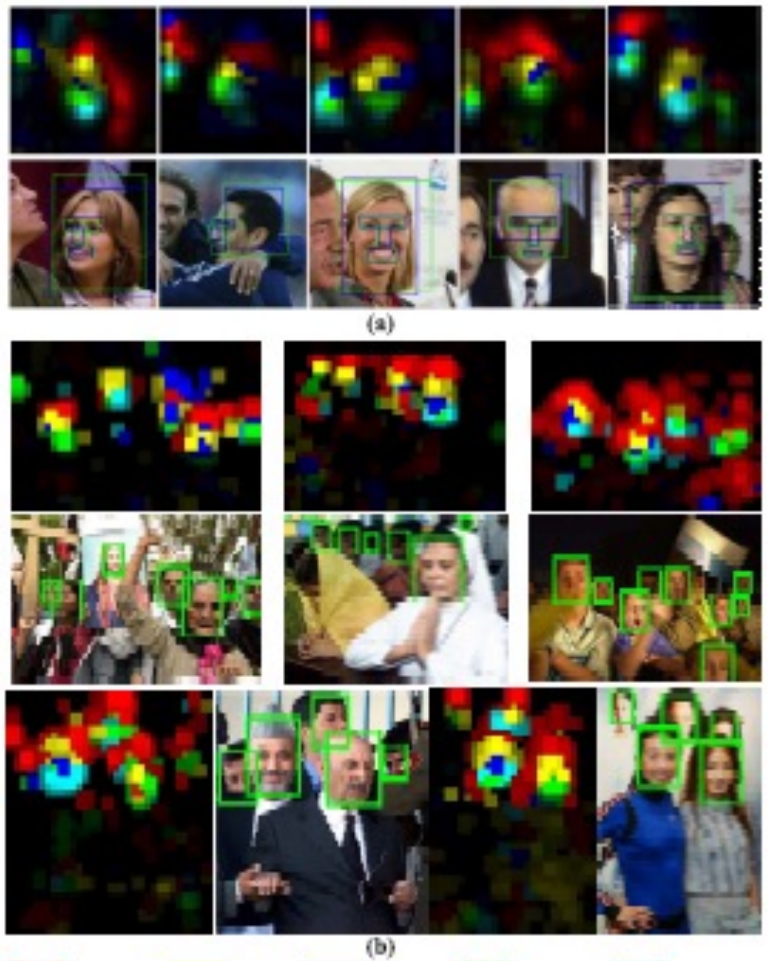

🟥 Hair 🟨 Eye 🟦 Nose 🟩 Mouth 🟦 Beard

Figure 6. (a) The top row depicts the response maps generated by the proposed approach on each part. The second row shows the part localization results. Ground truth is depicted by the blue bounding boxes, whilst our part proposals are indicated in green. (b) Face detection results on FDDB images. The bounding box in green is detected by our method. We show the partness maps as reference.

"From facial parts responses to face detection: A deep learning approach.,"
S. Yang, P. Luo, C. Loy, X. Tang, X.
Proceedings of the IEEE International Conference on Computer Vision 2015
(pp. 3676-3684).

Deep approaches

- The key of the approach is a new mechanism for scoring face likeliness based on deep network responses on local facial parts.
- The part-level response maps ('partness' map) generated by the deep network give a full image without prior face detection.

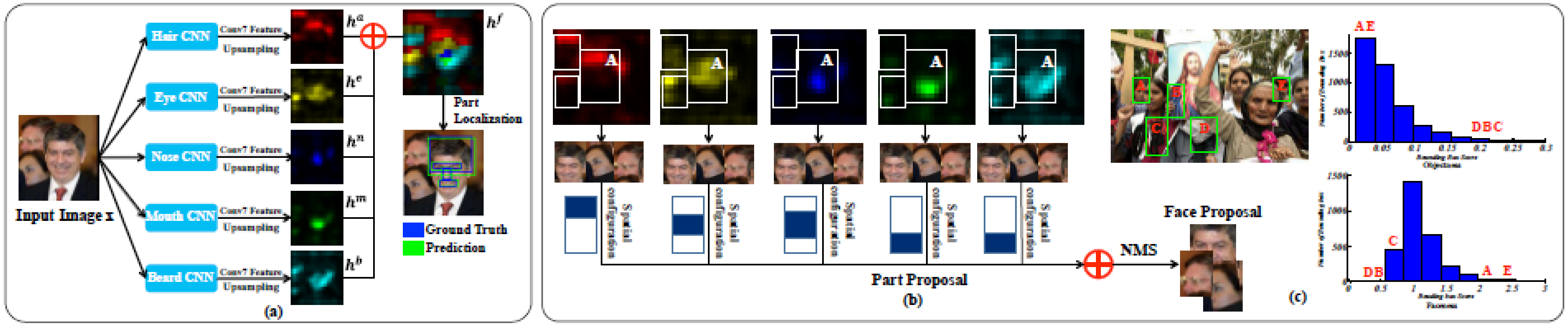


Figure 2. (a) The pipeline of generating part response maps and part localization. Different CNNs are trained to handle different facial parts, but they can share deep layers for computational efficiency. (b) The pipeline for generating face proposals. (c) Bounding box reranking by face measure (**Best viewed in color**).

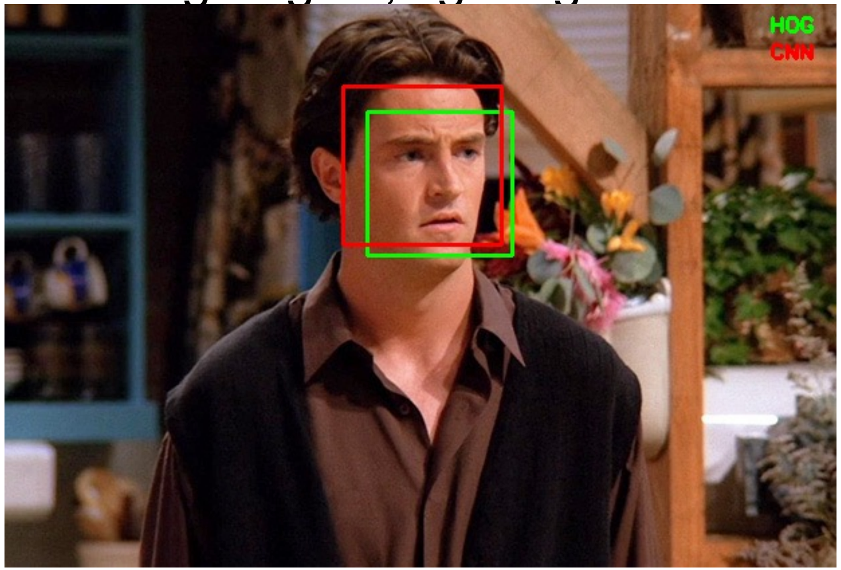
FROM LOCAL TO GLOBAL

Face detection with dlib (HOG and CNN)

dlib includes two face detection methods built into the library:

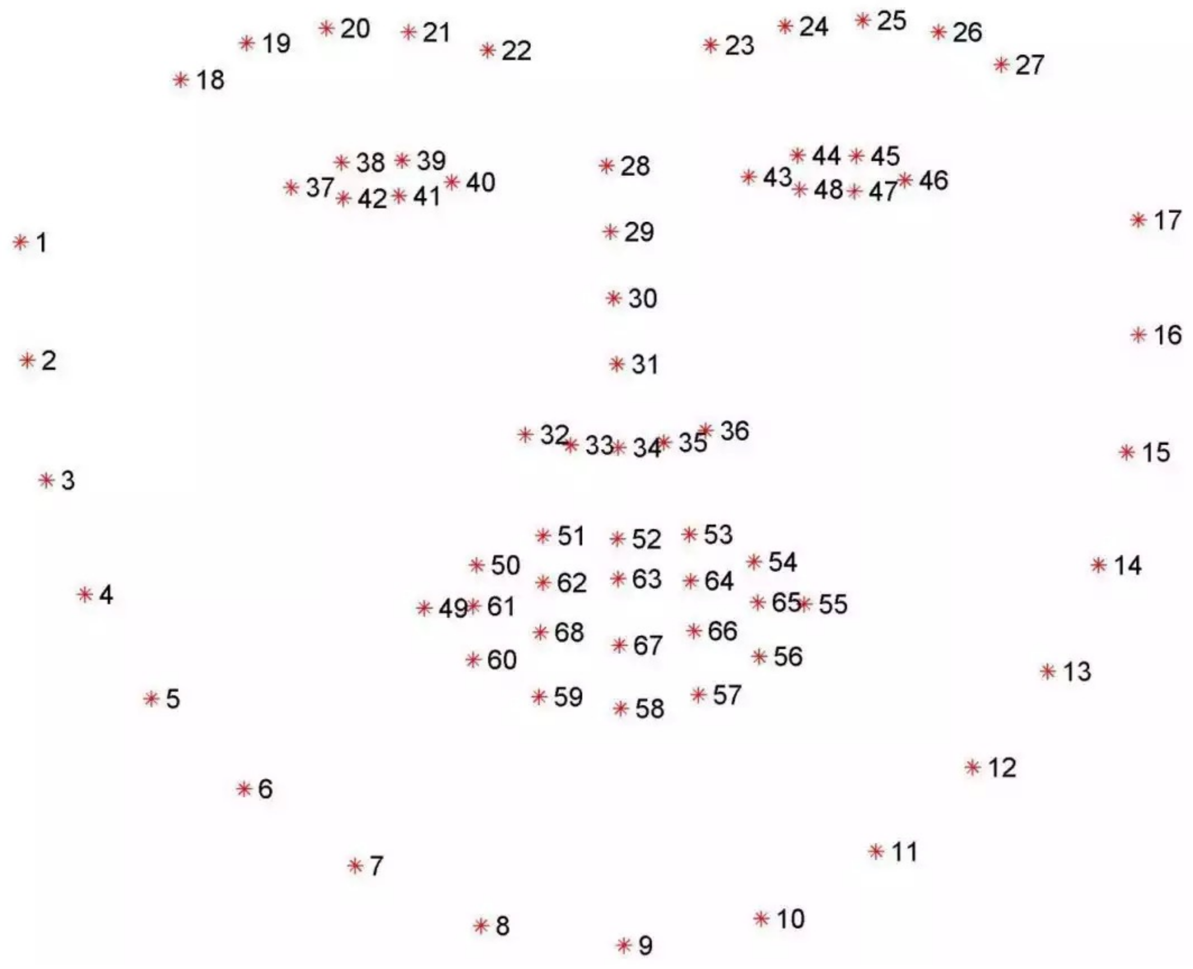
- A HOG (Histogram of Gradients)+ Linear SVM face detector that is accurate and computationally efficient.
  - Although the detector seemed to perform on faces which were bigger and front facing, it suffered when the orientation of the face changed, had any coverings, or if the face appeared very small in the image/video frame.

- A Max-Margin (MMOD) CNN face detector that is both highly accurate and very robust, capable of detecting faces from varying viewing angles, lighting conditions, and occlusion.



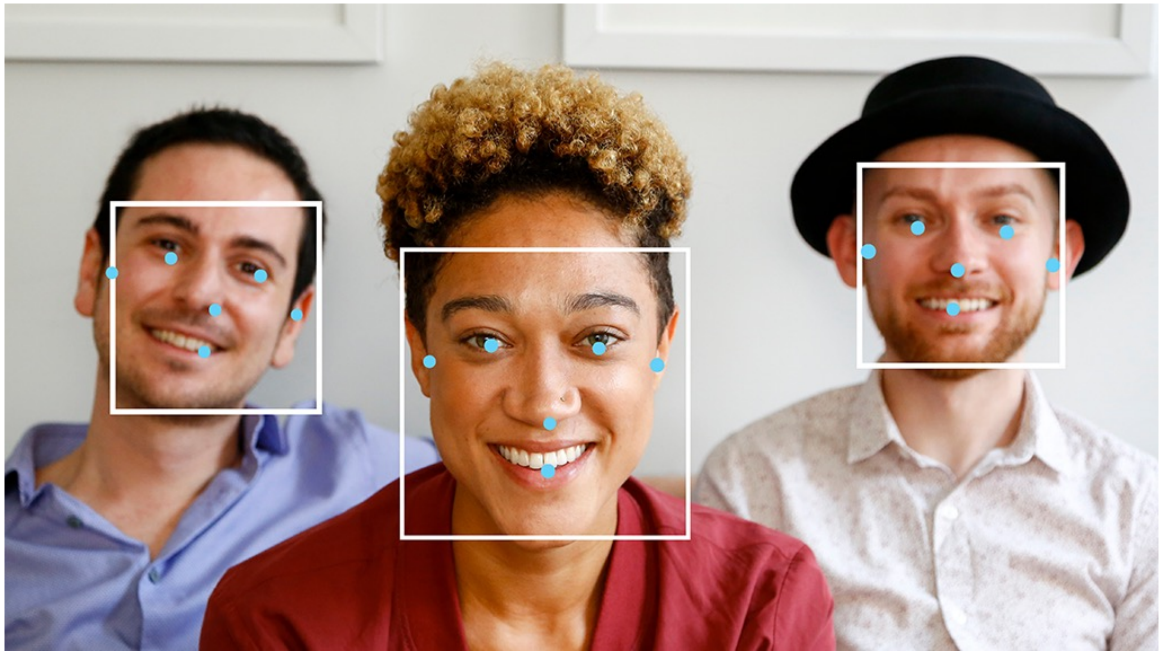
Face landmarks with dlib

- Dlib provides a pre-trained facial landmark detector that can detect 68 points on a face.



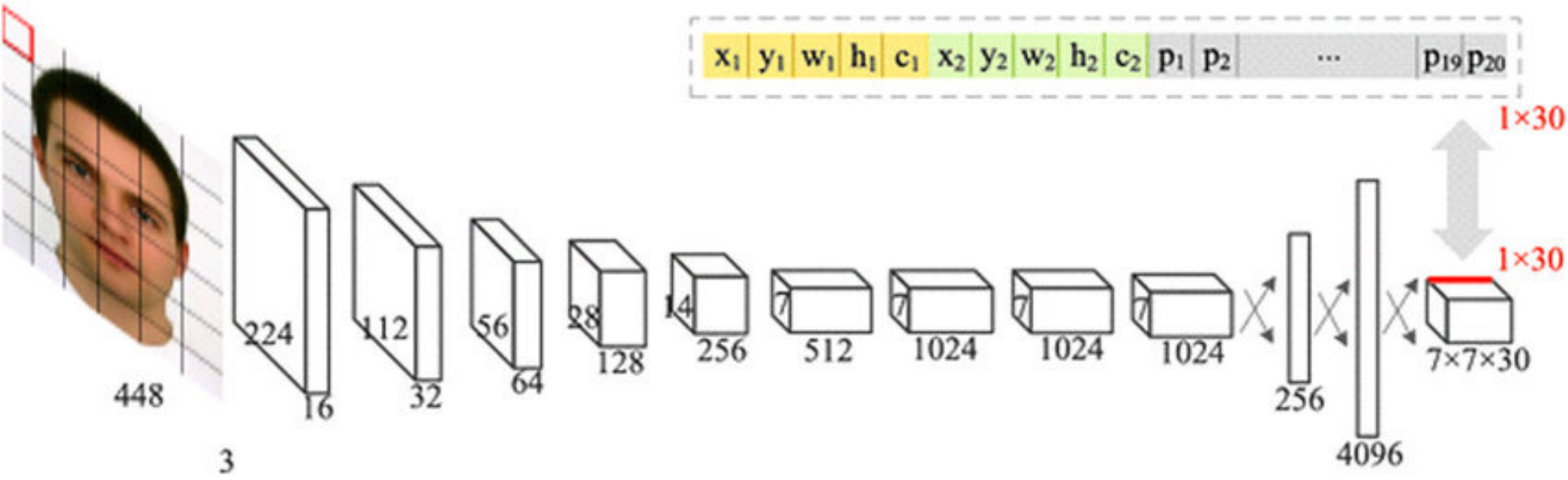
Face detection with MediaPipe (by Google)

- The MediaPipe Face Detector detects faces in an image or video.
- It is possible to locate faces and facial features within a frame.
- This task uses a machine learning (ML) model that works with single images or a continuous stream of images.
- The task outputs face locations, along with the following facial key points: left eye, right eye, nose tip, mouth, left eye tragion, and right eye tragion.



Face detection with YOLO

- YOLO, or You Only Look Once, is a neural network architecture for identifying objects in images.



x_1 y_1 w_1 h_1 c_1 x_2 y_2 w_2 h_2 c_2 p_1 p_2 ... p_{19} p_{20}
1×30
1×30
448
3
224
16
112
32
56
64
28
128
14
256
7
512
7
1024
7
1024
7
1024
256
4096
7×7×30

How to evaluate localization

- **False positives**
  Percentage of windows classified as faces that do not contain any face
- **Not localized faces**
  Percentage of faces that have not been identified
- **C-Error**
  Localization error: Euclidean distance between the real center of the face and the one estimated by the system, normalized with respect to the sum of the axes of the ellipse containing the face.

- Face detection home page:
  http://www.facedetection.com/

Some references

- R.L. Hsu, M.A. Mottaleb, A.K. Jain, “Face Detection in Color Images”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696-706, 2002.
- Nie, F., Wang, Y., Pan, M., Peng, G., & Zhang, P. (2013). Two-dimensional extension of variance-based thresholding for image segmentation. *Multidimensional Systems and Signal Processing*, *24*(3), 485-501.
- P. Viola, M. J. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, in proc. IEEE International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 511-518, 2001.
- Paul Viola, Michael Jones, Robust Real-Time Face Detection, International Journal of Computer Vision 57(2), 137–154, 2004
- Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.
- Qi, D., Tan, W., Yao, Q., & Liu, J. (2022, October). YOLO5Face: why reinventing a face detector. In European Conference on Computer Vision (pp. 228-244). Cham: Springer Nature Switzerland.