Interfețe Vizuale Om-Mașină
IVOM

Cosmin Țoca	
Mail.	cosmin.toca@gmail.com

IVOM

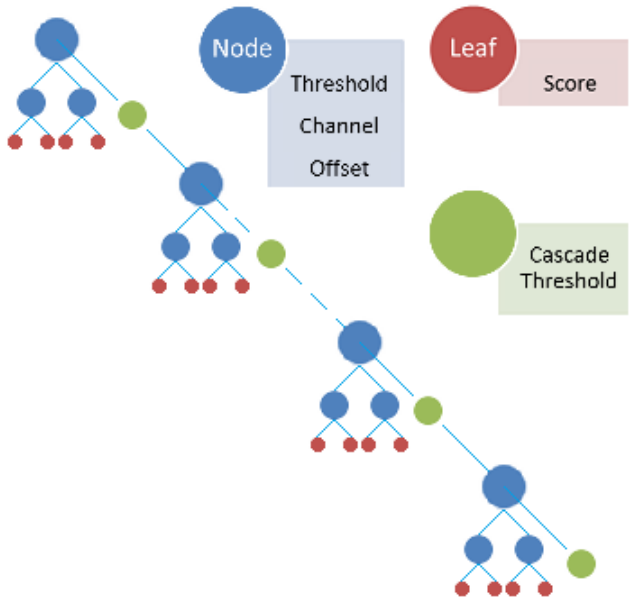
Overview

1. Haar-Like Features
2. Integral Image
3. AdaBoost Training

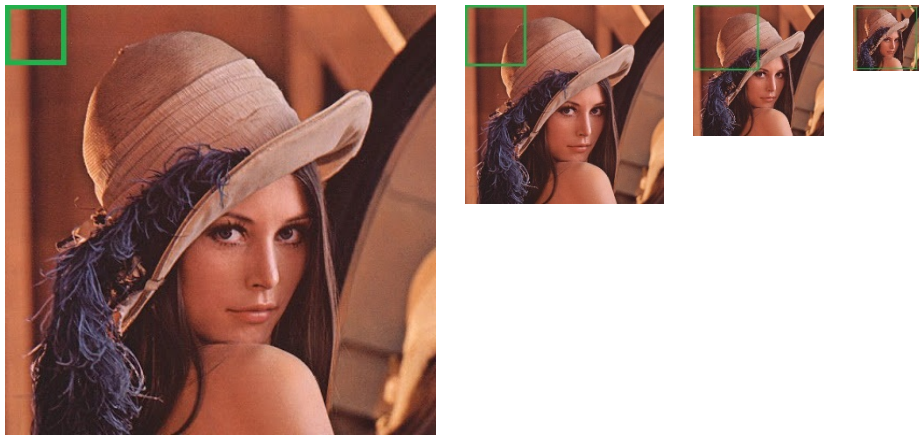
Tasks

P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, 2004

TRAINING (A CASCADE OF WEEK-CLASSIFIERS)



TESTING / INFERENCE (SCAN IMAGES AT VARIOUS SCALES)



Week Classifiers – Haar-Like Features

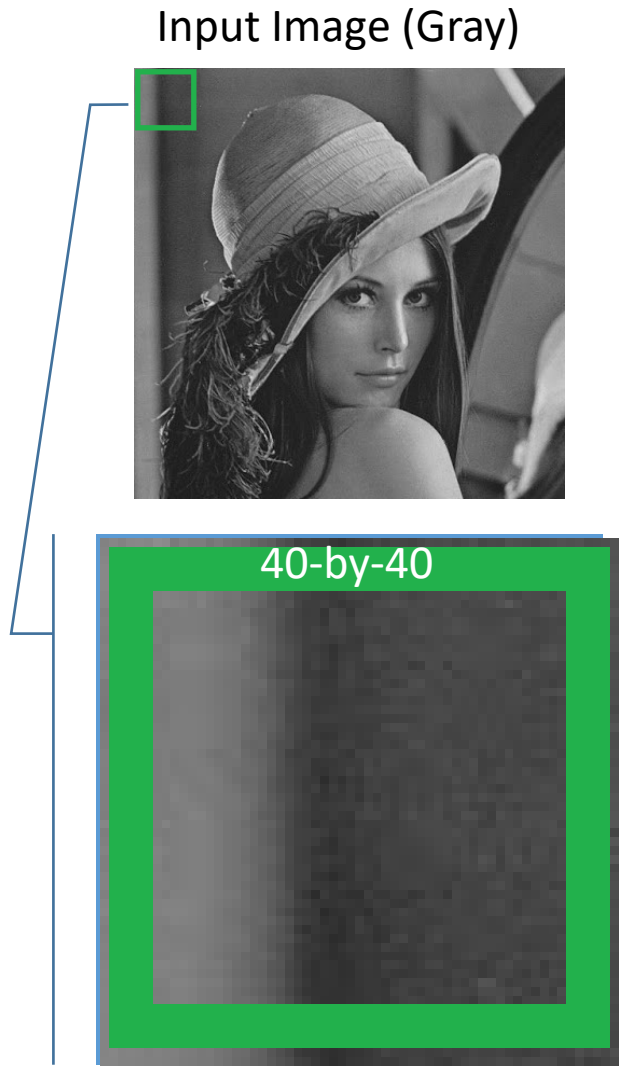
- Let's consider an input image **J** of size 400-by-400 pixels

Input Image (Gray)



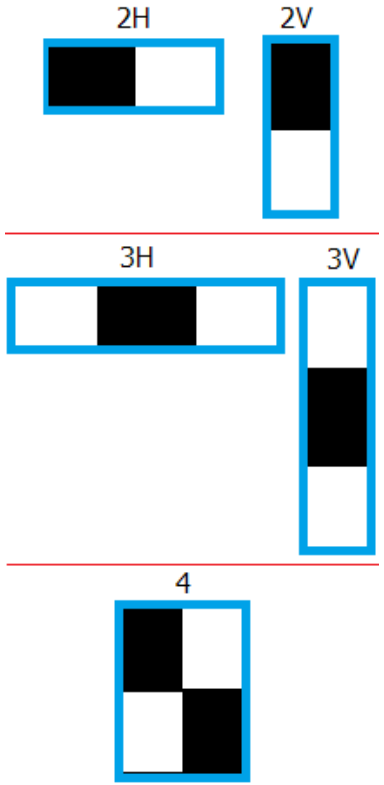
Week Classifiers – Haar-Like Features

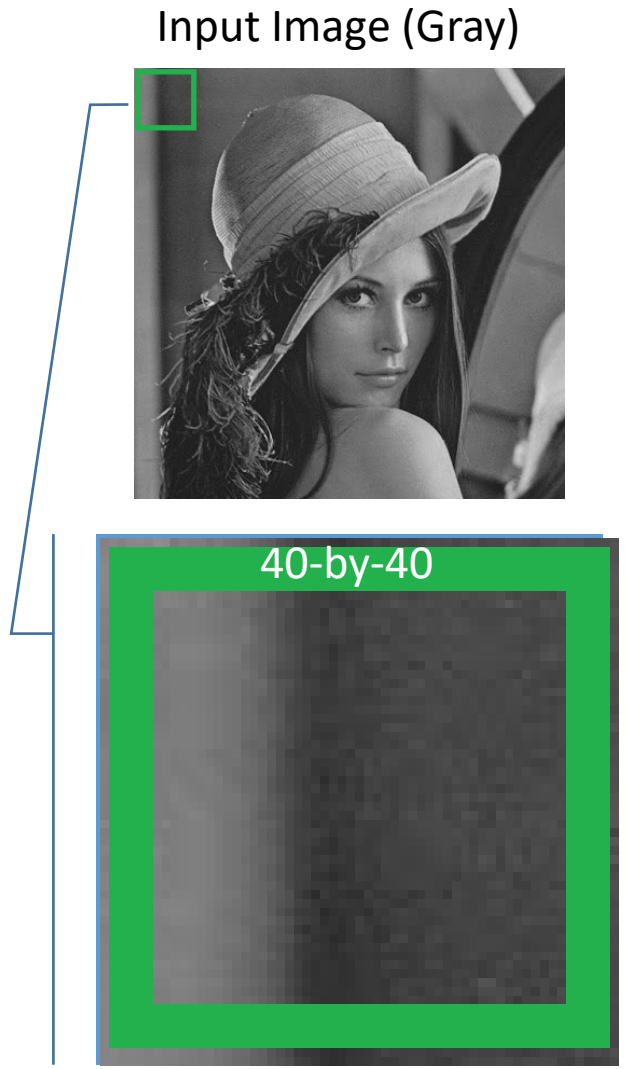
- Let's consider an input image **J** of size 400-by-400 pixels
- We define a sliding window **W** of size 40-by-40 pixels



Week Classifiers – Haar-Like Features

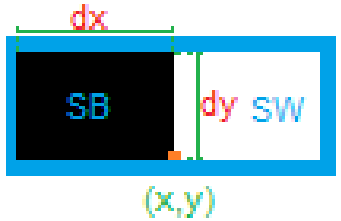
- Let's consider an input image **J** of size 400-by-400 pixels
- We define a sliding window **W** of size 40-by-40 pixels
- Haar-Like Feature Types:
  - 2H (x, y, dx, dy)
  - 2V (x, y, dx, dy)
  - 3H (x, y, dx, dy)
  - 3V (x, y, dx, dy)
  - 4 (x, y, dx, dy)

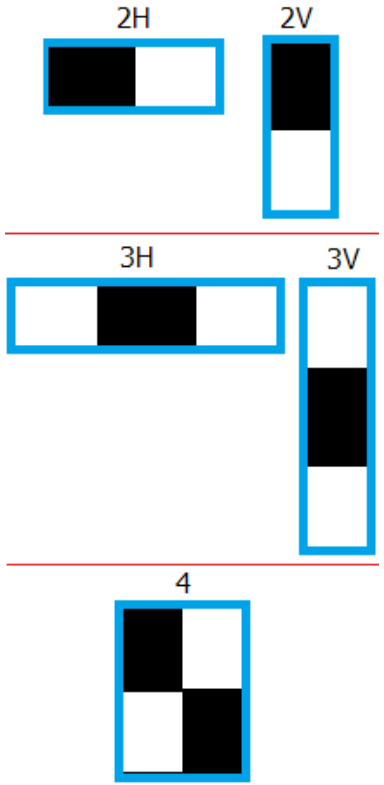




Week Classifiers – Haar-Like Features

- Let's consider an input image **J** of size 400-by-400 pixels
- We define a sliding window **W** of size 40-by-40 pixels
- Haar-Like Feature Types:
  - 2H (x, y, dx, dy)
  - 2V (x, y, dx, dy)
  - 3H (x, y, dx, dy)
  - 3V (x, y, dx, dy)
  - 4 (x, y, dx, dy)
- Features are defined by position and size within the sliding window





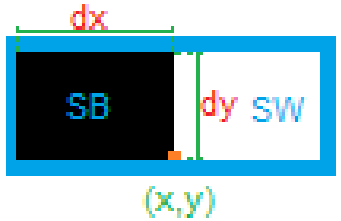
Input Image (Gray)



IVOM

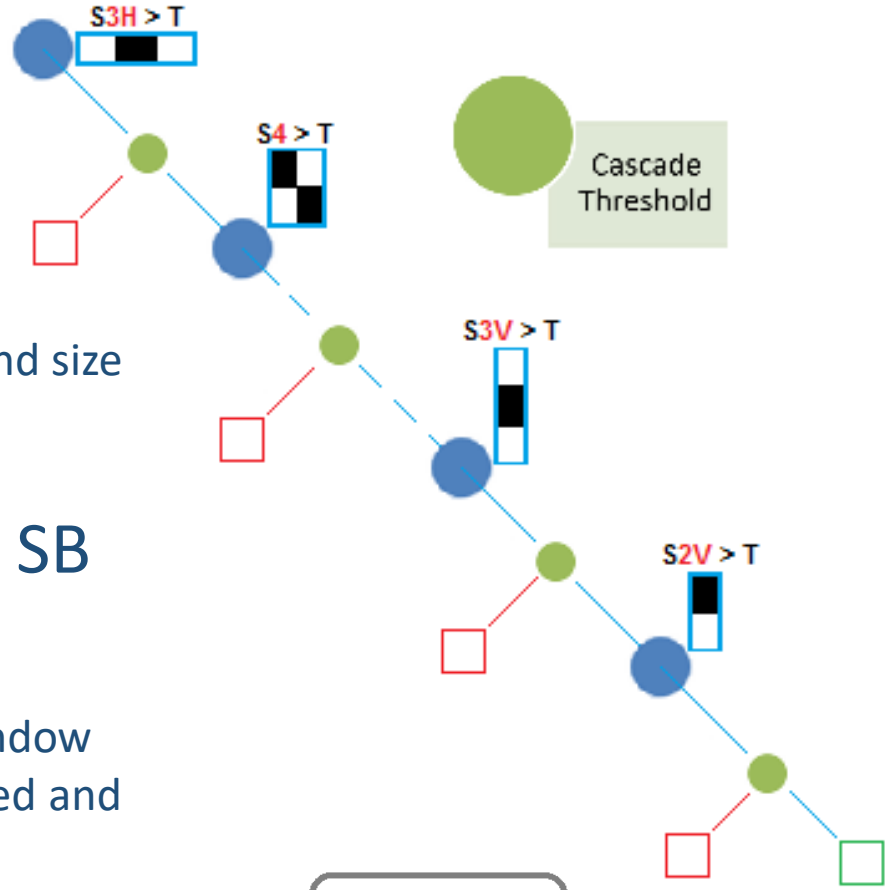
Week Classifiers – Haar-Like Features

- Let's consider an input image **J** of size 400-by-400 pixels
- We define a sliding window **W** of size 40-by-40 pixels
- Haar-Like Feature Types:
  - 2H (x, y, dx, dy)
  - 2V (x, y, dx, dy)
  - 3H (x, y, dx, dy)
  - 3V (x, y, dx, dy)
  - 4 (x, y, dx, dy)
- Features are defined by position and size within the sliding window



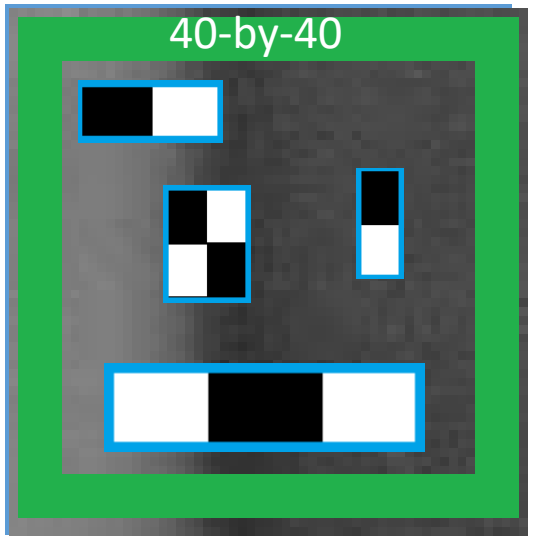
$$S2H = SW - SB$$

- For each position of the sliding window various Haar Features (**S**) are computed and compared to a threshold (**T**).



Input Image (Gray)





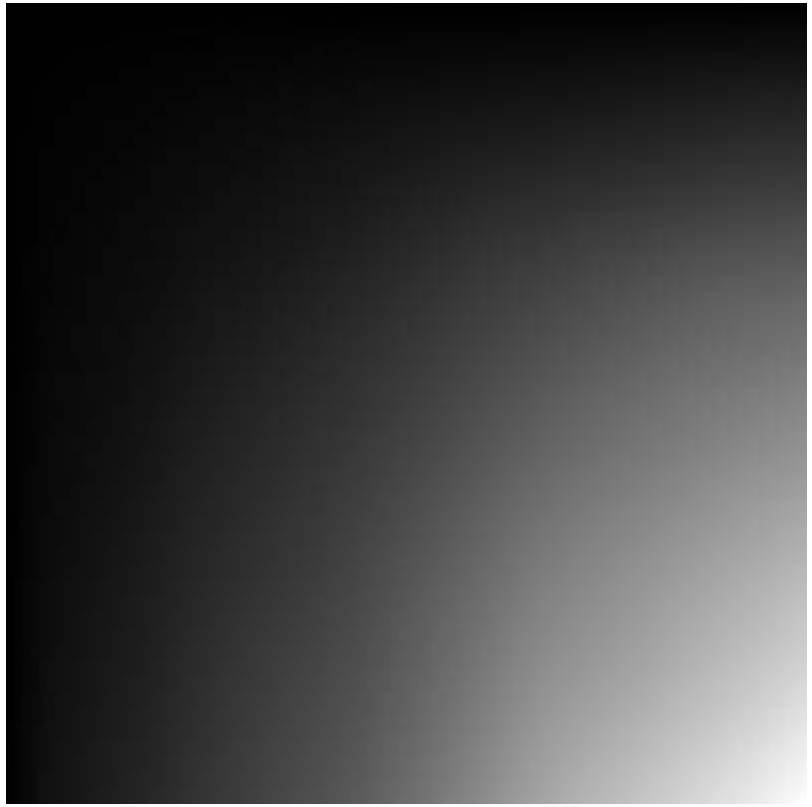
IVOM

Week Classifiers – Haar-Like Features



Question: How do you think the right side image has been obtained?





Integral Image (Summed Area Table)

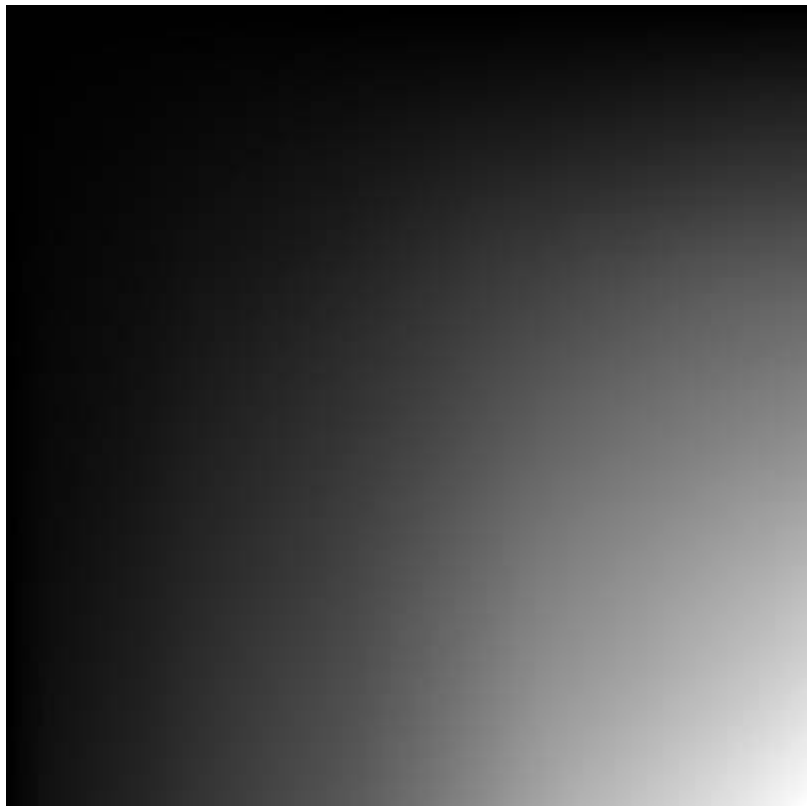
- Let's compute the integral image of **J**, and denote it by **I**

Input Image (Gray)



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

Integral Image



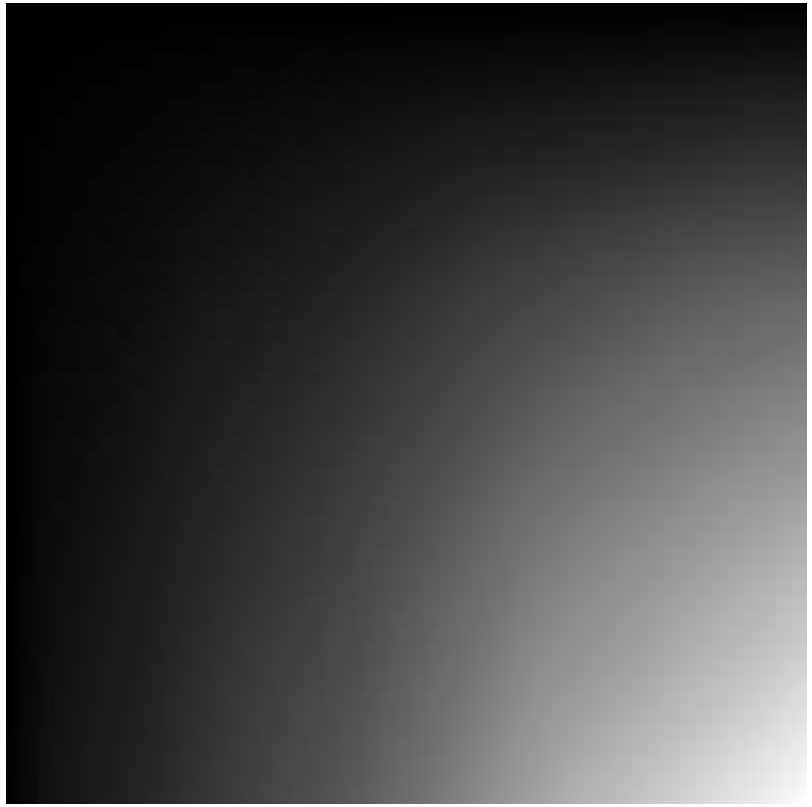
Question: For what do you think we use it?

Input Image (Gray)



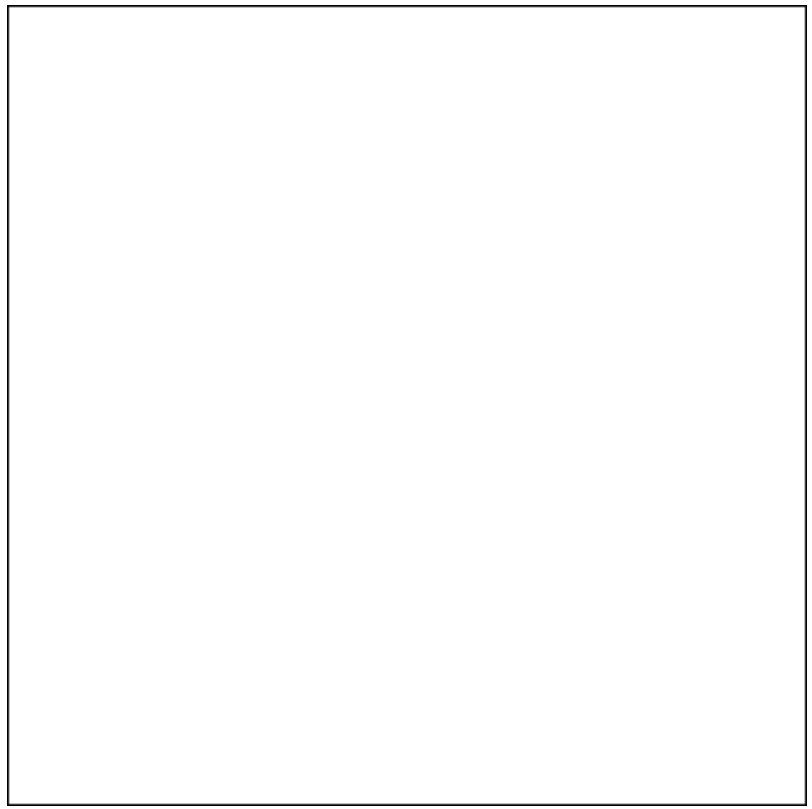
$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

Integral Image



Question: If the input image is white, what is the value of the bottom-right corner of the integral image?

Input Image (Gray)



400-by-400

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

Integral Image



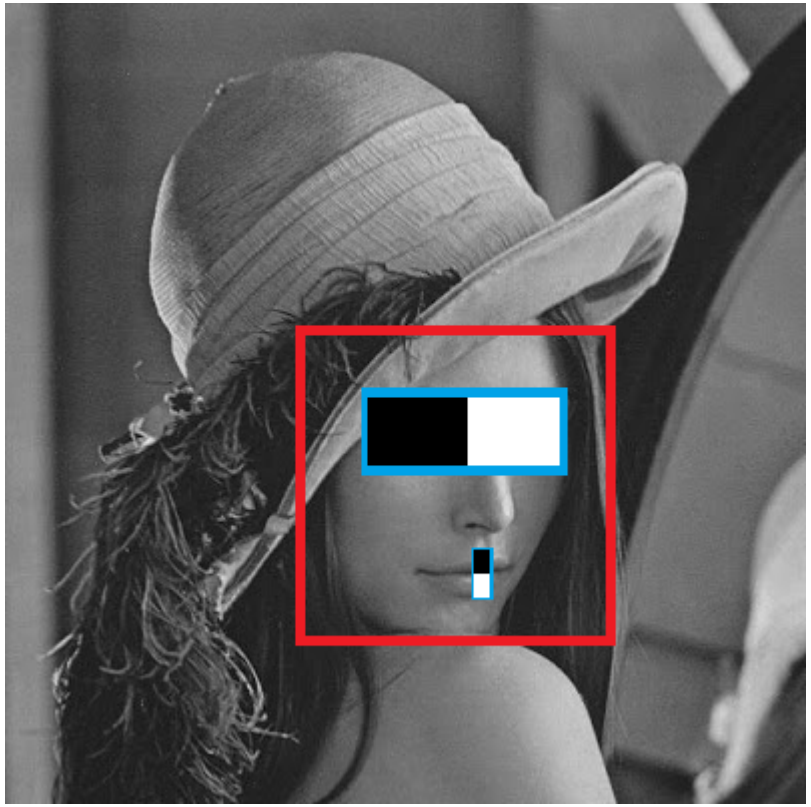
400-by-400

IVOM

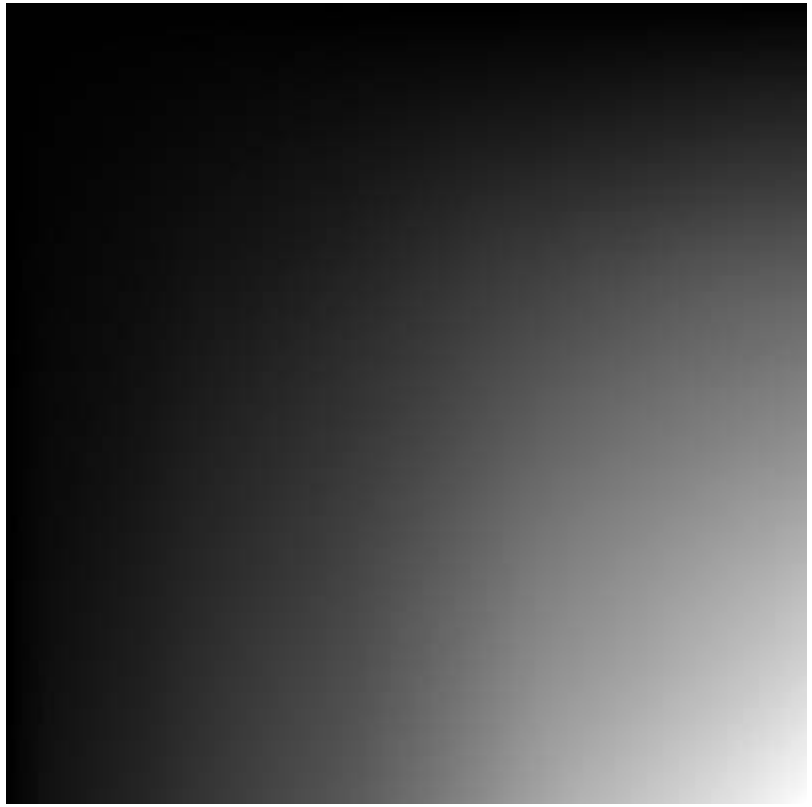
Integral Image (Summed Area Table)

- We need to compute the sum within several Haar-Like Features

Input Image (Gray)



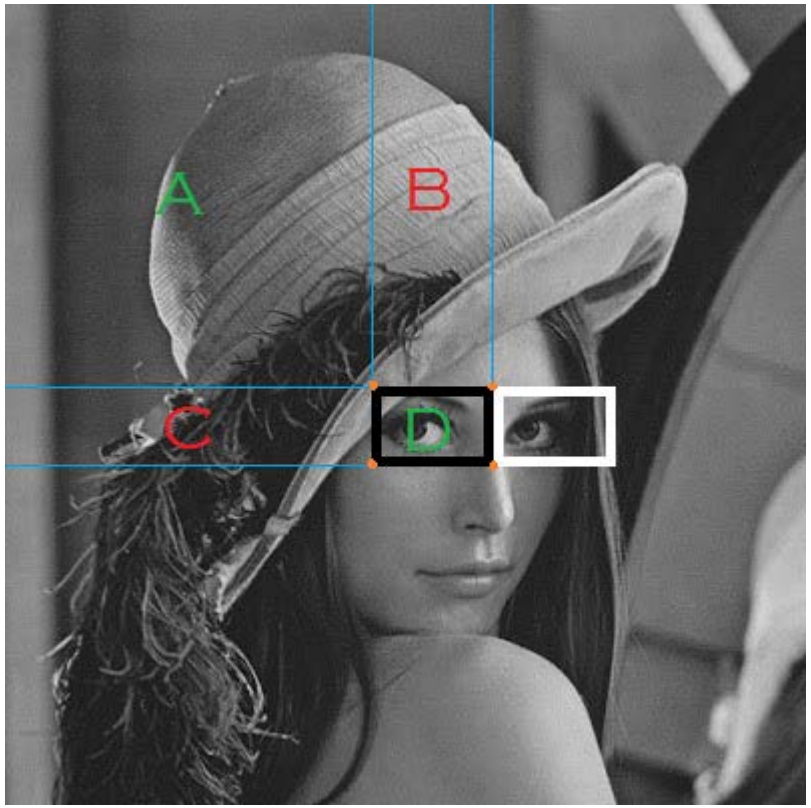
Integral Image



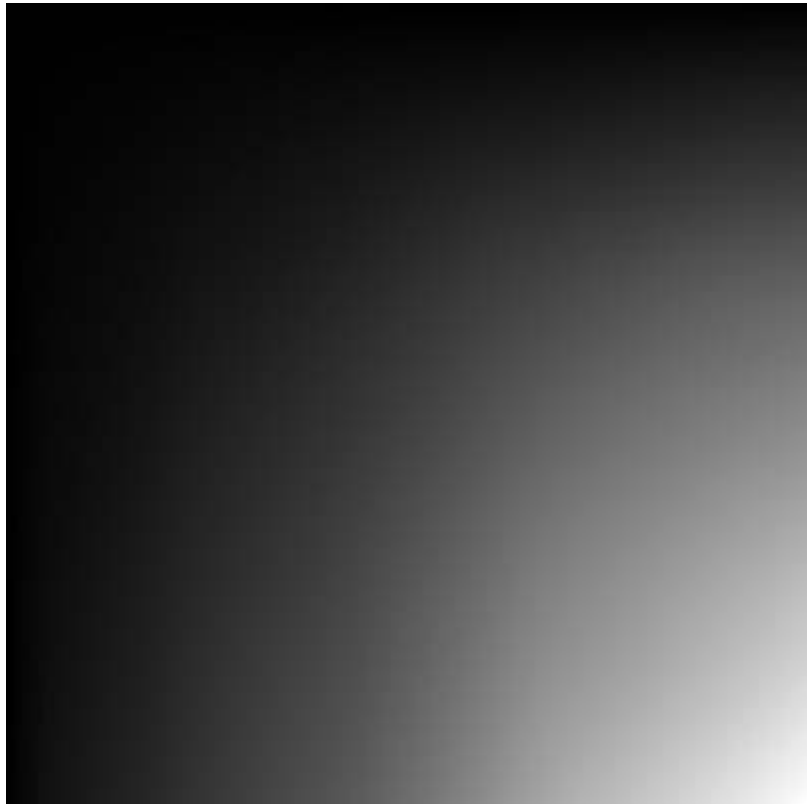
Integral Image (Summed Area Table)

- We want to compute the sum of the pixels inside the black rectangle below

Input Image (Gray)



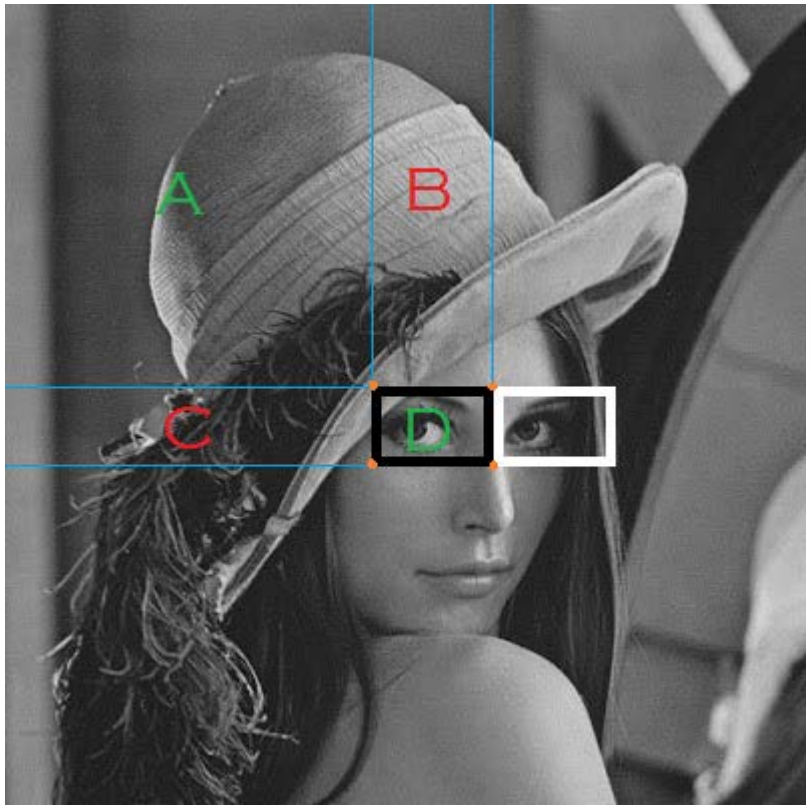
Integral Image



Integral Image (Summed Area Table)

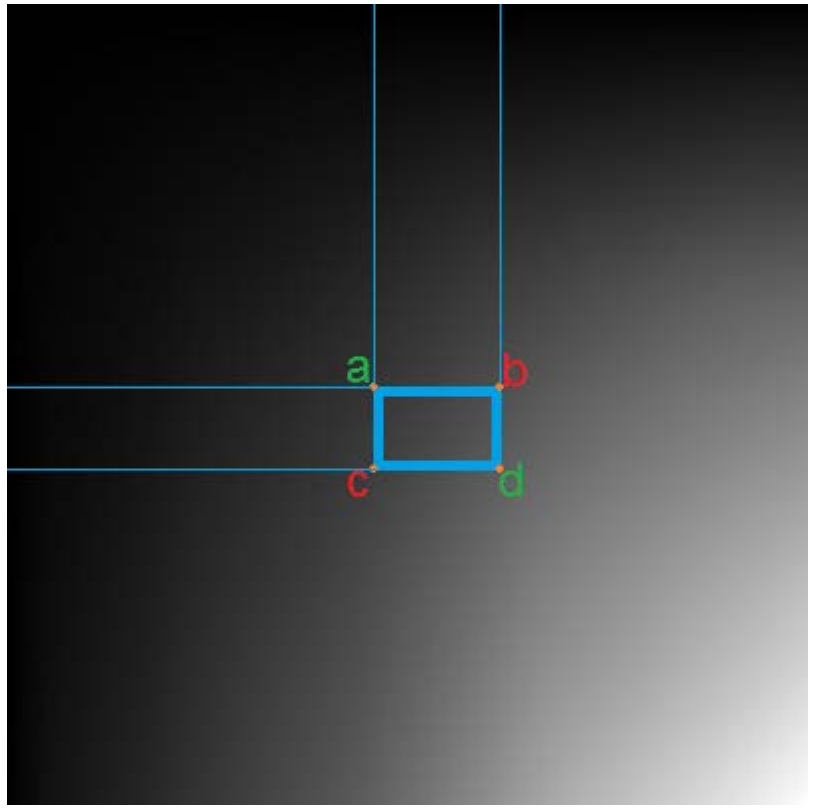
- We want to compute the sum of the pixels inside the black rectangle below

Input Image (Gray)



$S = I(d)+I(a)-I(b)-I(c)$

Integral Image



Exercise: Write a C Function that computes the integral image!

```
void computeIntImg ( unsigned char* in_img, int width, int height, unsigned int* out_img );
```

- Input:
  - Pointer to the input gray scale image
  - Input width
  - Input height
- Output
  - Pointer to the output image

- Observations:
  - Consider that both the input and the output images are externally allocated
  - Each element of the output image should retain the sum over all pixels located in the up-left side region in the original image.

Integral Image (Summed Area Table)

- Squared integral image (I^2)
- **Variance ($E[I^2] - (E[I])^2$)**

Exercise: Write a C Function that computes the integral image!

```
void computeIntImg ( unsigned char* in_img, int width, int height, unsigned int* out_img );
```

- Input:
  - Pointer to the input gray scale image
  - Input width
  - Input height
- Output
  - Pointer to the output image

- Observations:
  - Consider that both the input and the output images are externally allocated
  - Each element of the output image should retain the sum over all pixels located in the up-left side region in the original image.

Learning Algorithm – AdaBoost

Given a set of *m* positive and *l* negative training samples (*m* + *l* = *N*), with their labels :

(x_i, y_i)

where $y_i = 1$, for positives , $y_i = 0$, for negatives

1. Assign a weight to each input image (positives and negatives):

$$w_{1,i} = \frac{1}{2m} \qquad w_{1,i} = \frac{1}{2l}$$



Learning Algorithm – AdaBoost

2. For each $k = 1,\ldots,T$ (no. classifiers) :
   - Normalize the weights such that they are stochastic (sum to one).

$$w_{k,i} = \frac{w_{k,i}}{\sum_{j=1}^{N} w_{k,j}}$$

   - Apply the feature *t* to each image in the training set, then find the optimal threshold and polarity that minimizes the weighted classification error.

$$\epsilon_t = \sum_{i=1}^{N} w_{k,i} |h_t(x_i, p_t, \theta_t) - y_i|$$

   - Choose the classifier with the lowest error.
   - Adjust the weights for the next iteration, weights are reduced for the images that were correctly classified.

$$w_{k+1,i} = w_{k,i} \beta_k^{1-e_i} \qquad \beta_k = \frac{\epsilon_{t,\min}}{1-\epsilon_{t,\min}}$$

where $e_i = 1$, if image *i* has been correctly classified , $y_i = 0$, otherwise

Learning Algorithm – AdaBoost

3. Set the final classifier *h(x)* to 1 if:

$$\sum_{k=1}^{T} \alpha_k h_{t,k}(x) \geq \frac{1}{2} \sum_{k=1}^{T} \alpha_k$$

$$\alpha_k = \log \frac{1}{\beta_k}.$$

and to 0, otherwise.

Q&A