Object Detection and Feature Extraction

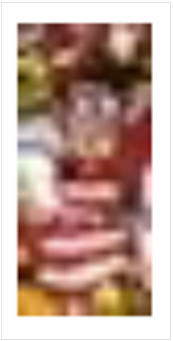
A.M. Tomé

[1]DETI / IEETA
Universidade de Aveiro, Portugal

November 22, 2016

Outline

Where is Wally?



Template

Template Matching

In template matching objects can be recognized by storing sample images (the template)

- **Requirements**
  - an example image of the object to be looked for.
  - a measure of similarity.

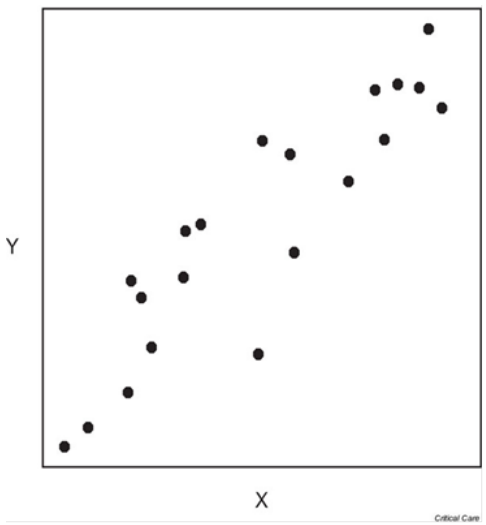
The image has the following cup?

If so. **Where is it?**

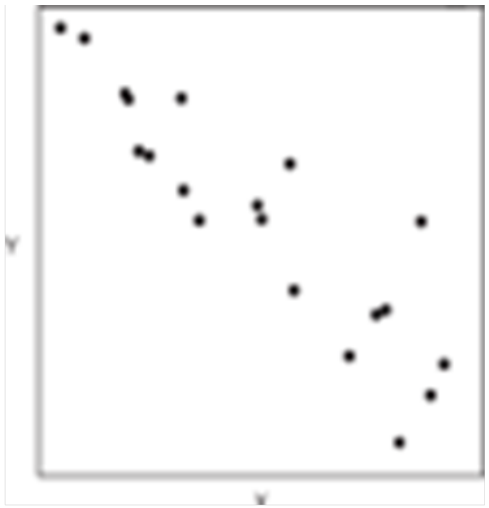
- **General strategy**
  - the template (a smaller image) is compared to all possible subregions of a larger image

Similarity measure: Correlation coefficient

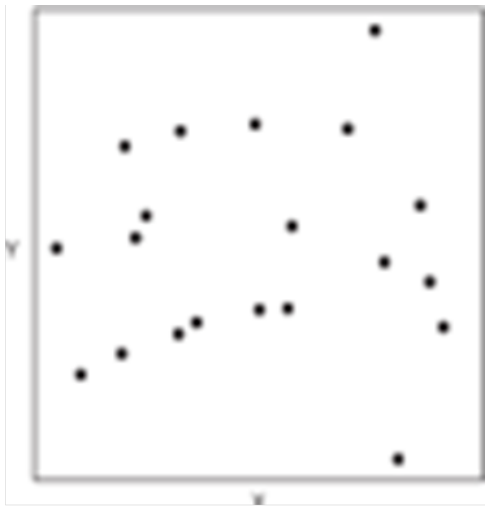
Scatter plot of two varibles x and y



ρ=0.9



ρ=−0.9



ρ=0.04

Statistic Measure within $[-1 \quad 1]$

- Close to 1 strong linear relation between variables.
- Close to 0 no relation between the two variables.

Correlation Coefficient (normalized correlation)

The measure of similarity based on 2D correlation

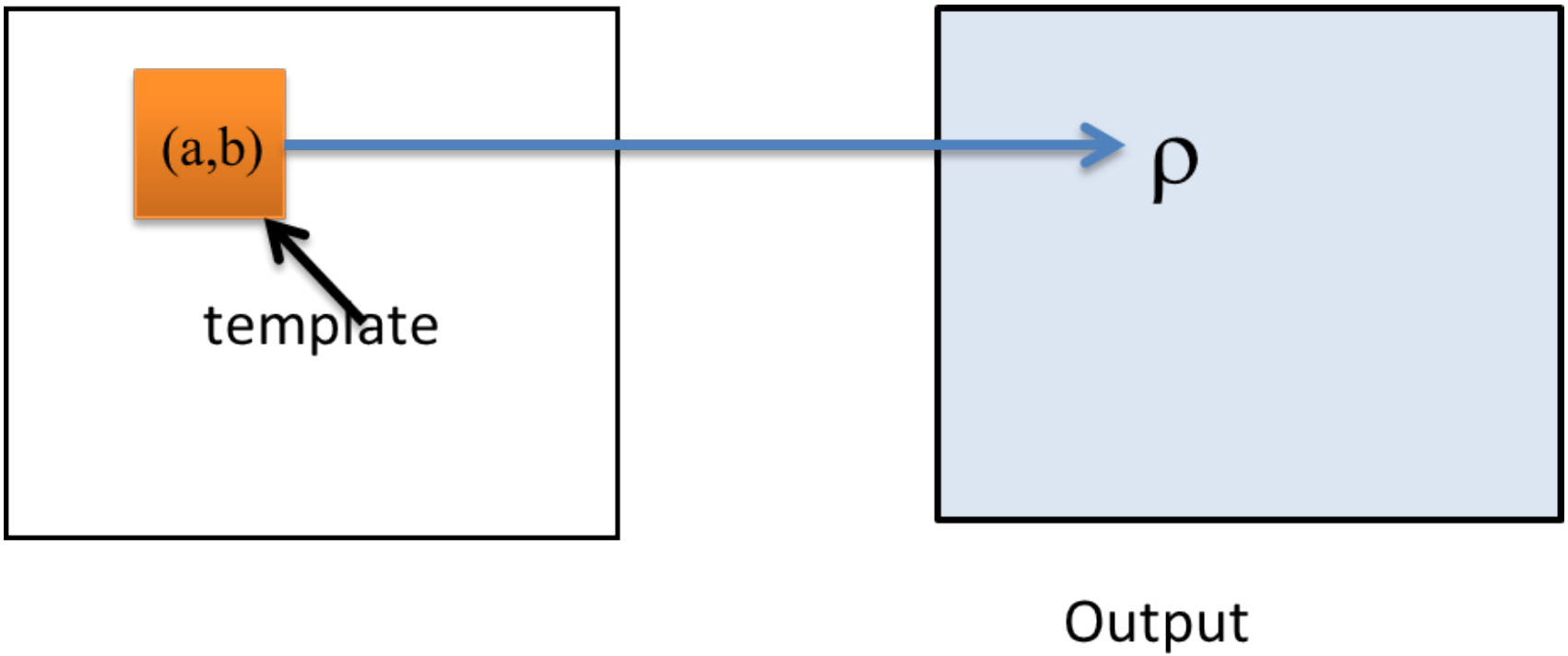
$$\rho(a,b) = \frac{\sum_{x=0}^{W}\sum_{y=0}^{H}(I_S(x+a,y+b)-\bar{I}_S)(I_T(x,y)-\bar{I}_T)}{\sqrt{\sum_{x=0}^{W}\sum_{y=0}^{H}(I_S(x+a,y+b)-\bar{I}_S)^2(\sum_{x=0}^{W}\sum_{y=0}^{H}(I_T(x,y)-\bar{I}_T)^2}}$$

Where

- $I_T(x,y)$ is a template with size $W \times H$
- $I_S(x+a, b+y)$ is the sub-region of the source image I_S at displacement $[a,b]$
- $\bar{I}_T$ is the mean value of the pixels in template
- $\bar{I}_S$ is the mean values within the region of size $W \times H$

Template Matching: method

The template is moved to all possible positions (a, b) source image and computes $\rho(a, b)$ that indicates how well the template matches the image in that position.

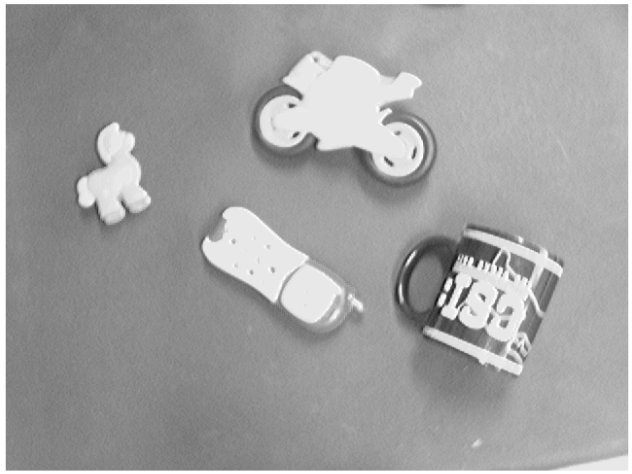


Decision: object is in on the image in position $[a, b]$ if

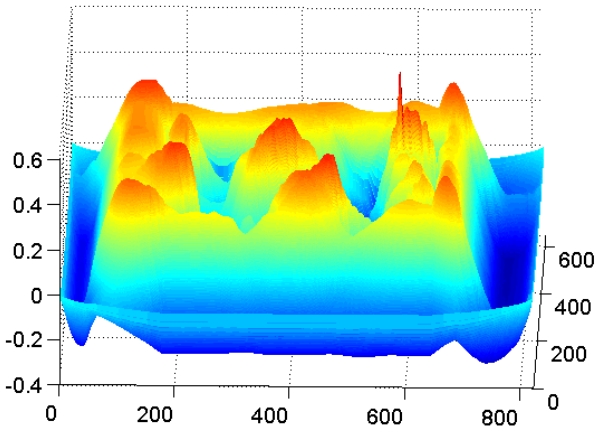
$$\rho(a, b) > thereshold$$

Example: normalized correlation function

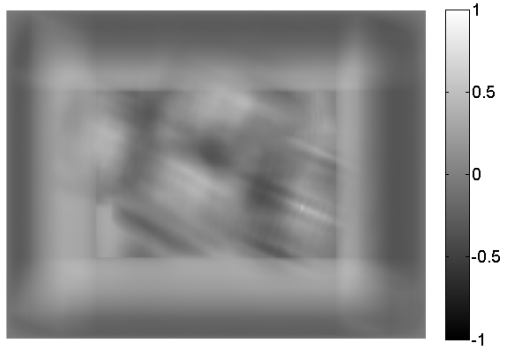
Given the template and image



The correlation coefficient (3D- view) and (2D-view)



- correlation coeficients: $[-0.38\ 0.59]$
- several peaks



Template Matching: main characteristics

Correlation coefficient (also called normalized correlation in object recognition works)

- is in range $[-1,\ 1]$ where 0 means non-correlated, a large positive value means high correlation.
- insensitive to **linear scaling** of contrast

Easy to implement but correlation coefficient

- is **not invariant to scaling or rotation** of the objects.
- sensitive to clutter and occlusion

Computationally demanding with exhaustive search but

- sub-optimal heuristics are often used.

Computational complexity

- Template image size: 40×36
- Image size: 200×236
- Assumption: template image is inside the source image.
- Correlation (search) matrix size: 160×200

Number of operations (product and add):
$40 \times 36 \times 160 \times 200 = 4608 \times 10^4$

Scaling and rotation problem

Application of template: different scales and rotations

Different rotations

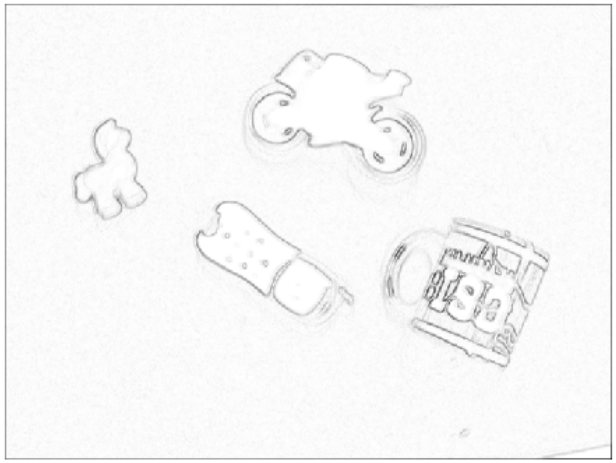
Different scales

Template Matching: Variant I

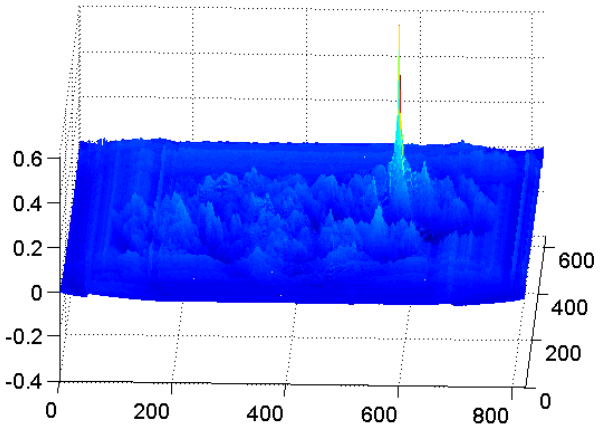
- Include a pre-processing step: by applying an edge detection filter (Canny filter) to both template and image.
- 2D correlation coefficients are computed on filtered image.

Example: after edge detection

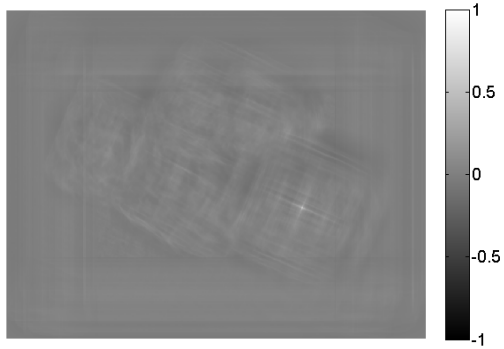
Gradient of template and image [1]



The correlation coefficient (3D- view) and (2D-view)



- correlation coeficients: $[-0.12\ 0.79]$
- well defined peak

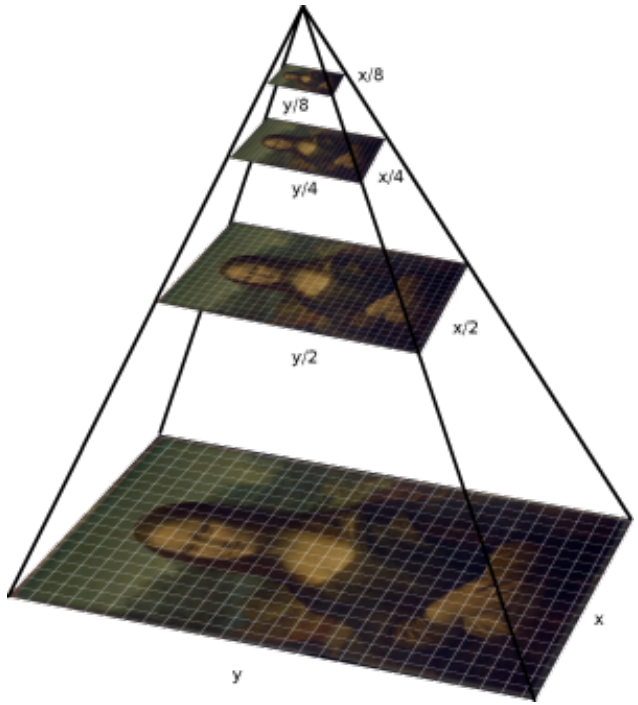


[1] visualization negative images

Template Matching: Variant II

To speed up use image pyramidal images (hierarchical down-sampling of an image)

- Start by the low-resolution image and get **coarse** object **location** hypothesis
- In higher resolution scales only the **neighborhood of the coarse locations** of previous are analyzed.



The obvious **advantage** is **avoiding** to an **exhaustive search** in the original (higher resolution) image.

- Computational saving depends on the number of levels of the pyramid
- Efficient algorithms but increased memory requirements

Template Matching: Variant III

Phase -Only Correlation (POC): correlation is performed on the **Fourier Transform domain** Given the Fourier transforms of both template and regions of original image

- Region of source image : $F_S = A_S(u,v)e^{j\theta_S(u,v)}$
- Template:$F_T = A_T(u,v)e^{j\theta_T(u,v)}$

The normalized cross-correlation of spectrum

$$R(u,v) = \frac{F_S(u,v)F_T^*(u,v)}{|F_S(u,v)F_T^*(u,v)|} = e^{j(\theta_S(u,v)-\theta_T(u,v))}$$

The inverse Fourier of the phase is POC, is easily computed as

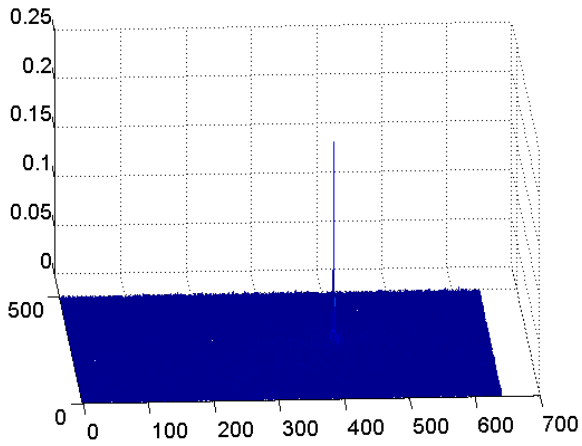
$$r(x,y) = \mathcal{F}^{-1}(R(u,v))$$

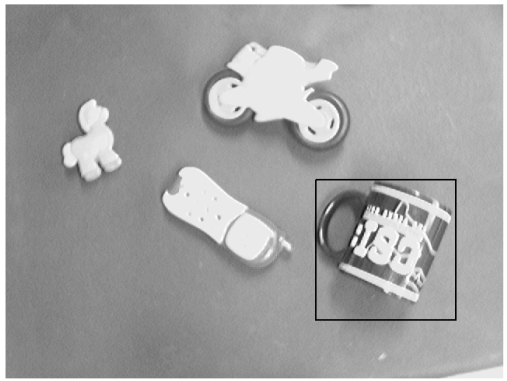
Template Matching: POC implementation issues

The Fourier transforms should be implemented with the same number of points:

1. Zero padding the image of template
2. Computing POC of subregions (with the size of the template) of the image.

Zero-padding template: $r(x, y)$ has the size of image



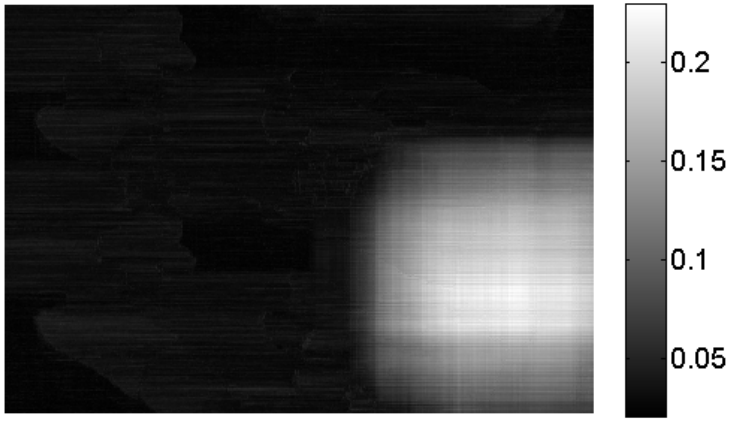


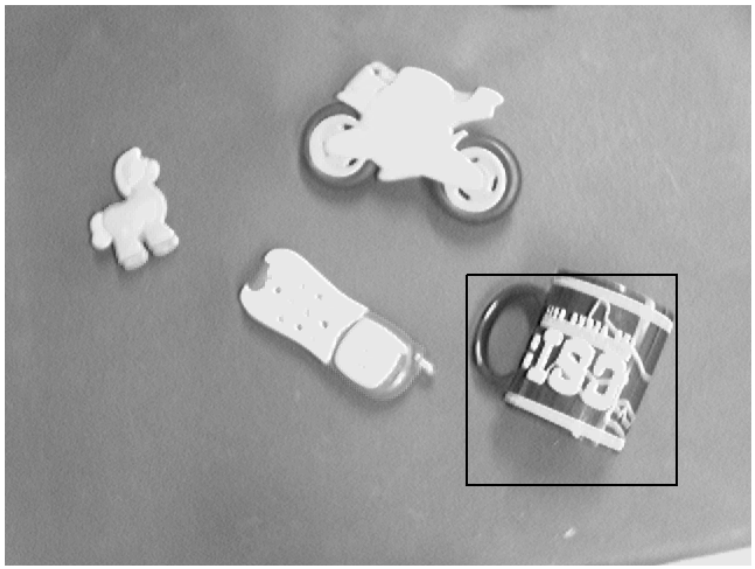
Complexity: 3 2D-FFT are computed: two direct and one inverse.

Template Matching: POC implementation issues

Computing POC in sub-regions of the image

- How many sub-regions? Exhaustive search: $P = (M - W) \times (N - W)$
- Choose the maximum value of $r(x, y)$ for each p subregion





Complexity: $(2P + 1)$ 2D-FFT. Possible to avoid exhaustive search?
Remarks: FFT involve smaller size images.

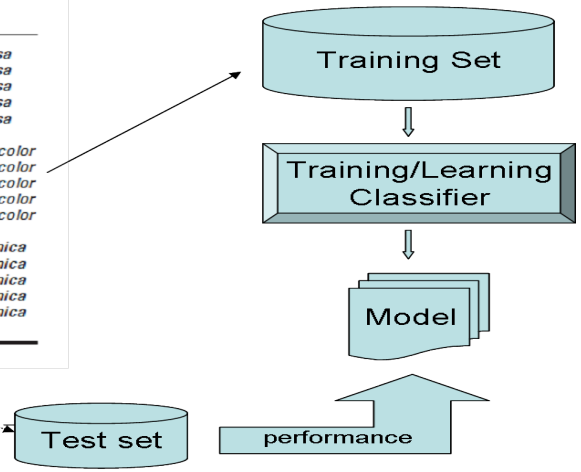
Computer Vision tasks: Detection, recognition and classification

- **Detection**: find a pre-defined object in a static image (or video-frame)
  - A binary classification problem (yes/no object) in sub-regions of image.
- **Recognition**: identify an object as belonging to one of n possible classes. In face recognition the possible sub-tasks
  - identification: tell if the persons is one the n possible persons
  - verification: tell if the person is the one it is claimed.
- Recognition usually involves as pre-processing block the detection step.

Classification



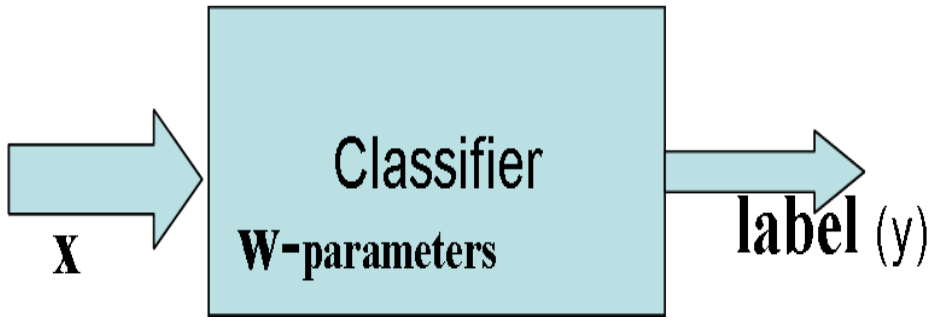
	Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)	Type
1	5.1	3.5	1.4	0.2	*Iris setosa*
2	4.9	3.0	1.4	0.2	*Iris setosa*
3	4.7	3.2	1.3	0.2	*Iris setosa*
4	4.6	3.1	1.5	0.2	*Iris setosa*
5	5.0	3.6	1.4	0.2	*Iris setosa*
...					
51	7.0	3.2	4.7	1.4	*Iris versicolor*
52	6.4	3.2	4.5	1.5	*Iris versicolor*
53	6.9	3.1	4.9	1.5	*Iris versicolor*
54	5.5	2.3	4.0	1.3	*Iris versicolor*
55	6.5	2.8	4.6	1.5	*Iris versicolor*
...					
101	6.3	3.3	6.0	2.5	*Iris virginica*
102	5.8	2.7	5.1	1.9	*Iris virginica*
103	7.1	3.0	5.9	2.1	*Iris virginica*
104	6.3	2.9	5.6	1.8	*Iris virginica*
105	6.5	3.0	5.8	2.2	*Iris virginica*
...					



Classification

An object is described by a set of measurements

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_N \end{pmatrix}$$



Main issues in classification tasks:

- Given a collection of records/objects (training set).
- Learning or Training Phase: Find a model $g(\mathbf{w}, \mathbf{x})$
- Given a test set: study the accuracy of the model (performance of the model)

HOW THE MODEL WORKS?

Linear Classifiers : Two Classes Problems

Linear discriminant functions

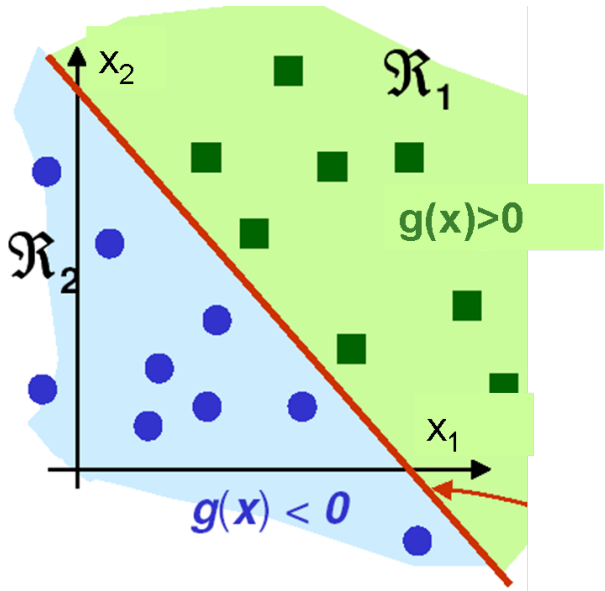
$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \Rightarrow \begin{cases} g(\mathbf{x}) > 0 & \mathbf{x} \in \omega_1 \\ g(\mathbf{x}) < 0 & \mathbf{x} \in \omega_2 \end{cases}$$

Parameters of the classifier: $\mathbf{w}$ and $b \equiv w_0$.

- The linear decision boundary (the hyperplane) the points $\mathbf{x}$ $g(\mathbf{x}) = 0$.

Example of objects described by $N = 2$ features

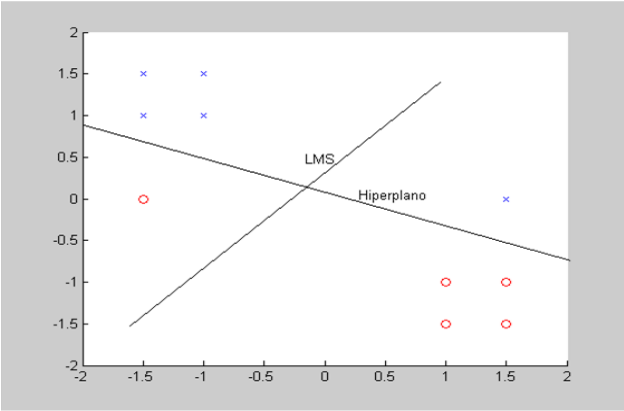
$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

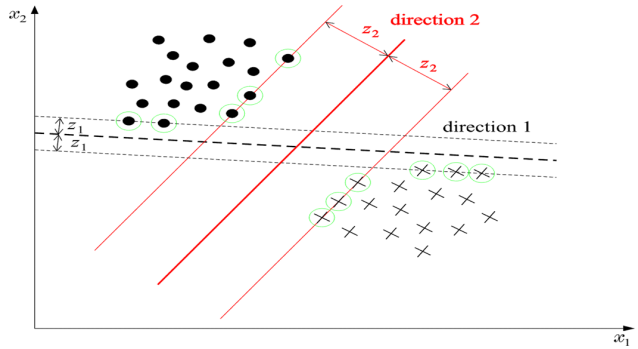


Linear decision boundary (red)

Linear Classifiers and training algorithms and solutions

Examples: perceptron, least mean square (LMS) and support vector machines (SVM)





non-unique solutions

- perceptron converges if the the training set is linearly separable.
- LMS is based on the minimization of the mean square error.

unique solution

- SVM is the maximum margin classifier. The hyperplane that leaves the maximum margin from both classes

NOTE: MODELS able to find non-linear decision boundaries are available....

ADABOOST and WeakLearners

Most popular algorithm in the family of boosting algorithms

- Boosting: the performance of simple (weak) classifiers is boosted by combining them iteratively.
- Combination rule

$$g(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x})$$

  where α_t means the importance of classifier f_t
- Simplest framework: binary classification, each $f_t \rightarrow -1, +1$
- The following simplest requirement: each weak classifier f_t should perform better than chance

Adaboost

AdaBoost is an iterative algorithm: train $f_t, t = 1 \ldots T$ given the performance of previous "weak" classifiers

- At each step t
  - **Modify** training **sample distribution** in order to favor difficult examples (according to previous weak classifiers).
  - Train a new weak classifier f_t
  - Select the new weight α_t by optimizing a global criterion
- **Stop** when *impossible to find a weak classifier* satisfying the simplest condition (being better than chance)
- **Final classifier** is the combination of all T classifiers
  - the global classifier can deal with thousands of features in spite of having a few hundred examples to train.
  - Boosting and works very well in practice.

Adaboost: complete algorithm

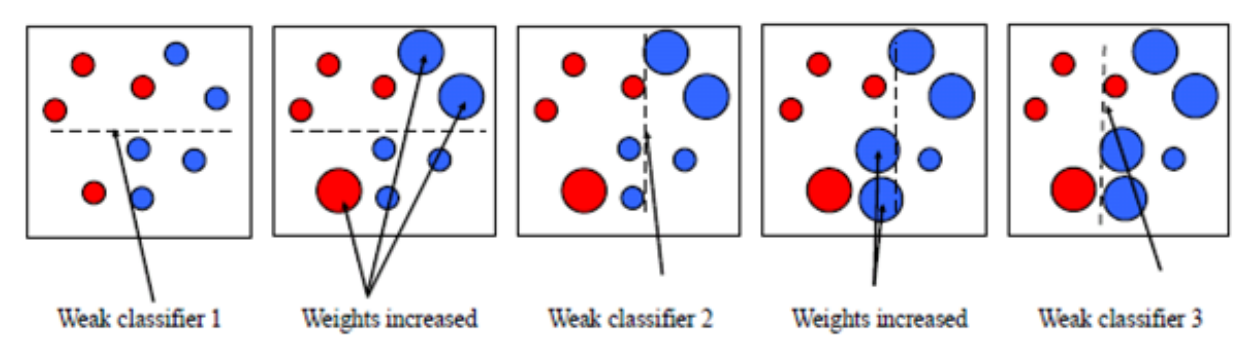
- Original training set: $D_n = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)$
- initialize weights for all objects $w_i^{(1)} = \frac{1}{n}, i = 1 \ldots n$
- for $t = 1 \ldots T$
  1. Train classifier f_t
  2. Calculate performance error in D_n, $\rightarrow \epsilon_i = \frac{1}{n} \sum_{j=1}^{N} w_j I(f_t(\mathbf{x}_j) \neq y_j)$, where $I(p) = 1$, if p is true otherwise 0
  3. calculate α_t of classifier f_t : $\alpha_t = \frac{1}{2} \frac{1-\epsilon_i}{\epsilon_i}$
  4. Update weights of the data, $j = 1, \ldots n$ for next iteration

$$w_j^{(t+1)} = w_j^t \frac{exp(-\alpha_t y_j f_t(\mathbf{x}_j))}{Z_t}$$

     where Z_t is a normalization term $\rightarrow \sum_j w_j^{(t+1)} = 1$
  5. if $\epsilon_t = 0$ or $\epsilon_t > 0.5$ break
- final output $g(\mathbf{x}) = \sum_t \frac{\alpha_t f_t(\mathbf{x})}{\sum \alpha_t}$

Adaboost and data distribution

The weights $w_j, j = 1 \ldots n$ change according to the results of the weeaklearn t



Once

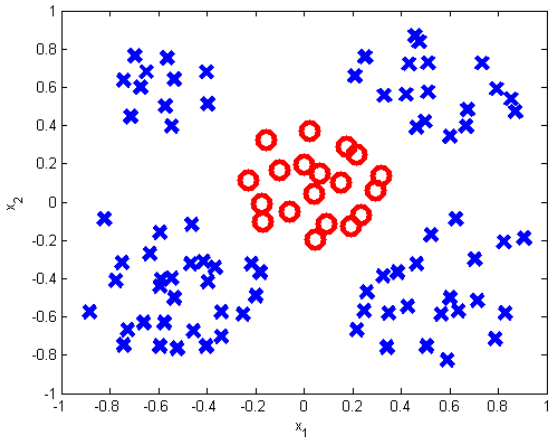
$$w_j^{(t+1)} = w_j^t \frac{exp(-\alpha_t y_j f_t(\mathbf{x}_j))}{Z_t}$$

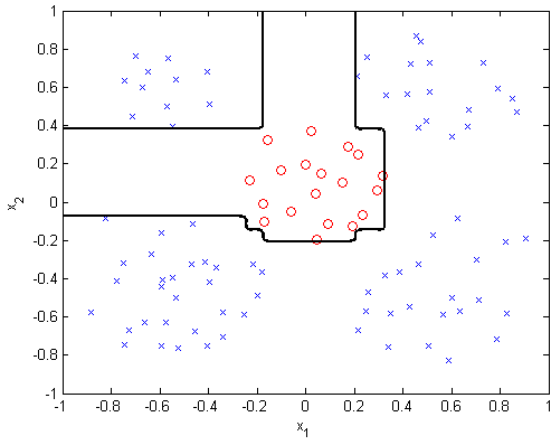
where Z_t is a normalization term $\rightarrow \sum_j w_j^{(t+1)} = 1$

- Right decision exponential has negative argument
- Wrong decision exponential has positive argument

Toy Example: Adaboost and WeakLearner

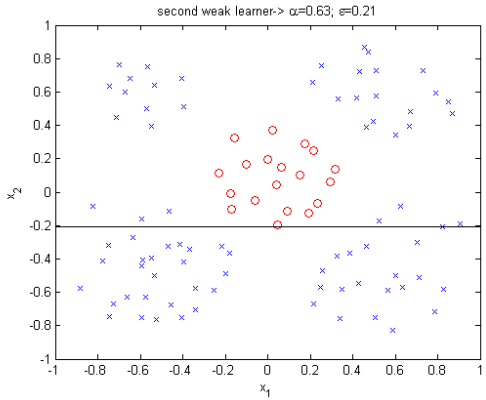
Given the data set

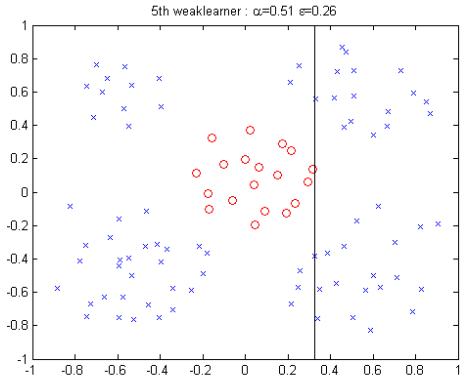




Adaboost solution with T=15 weaklearners

Each learner decides based on a single feature

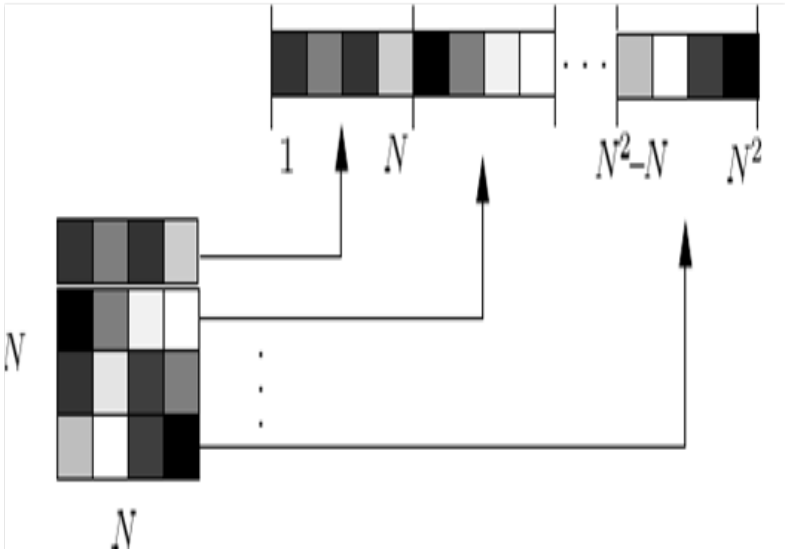


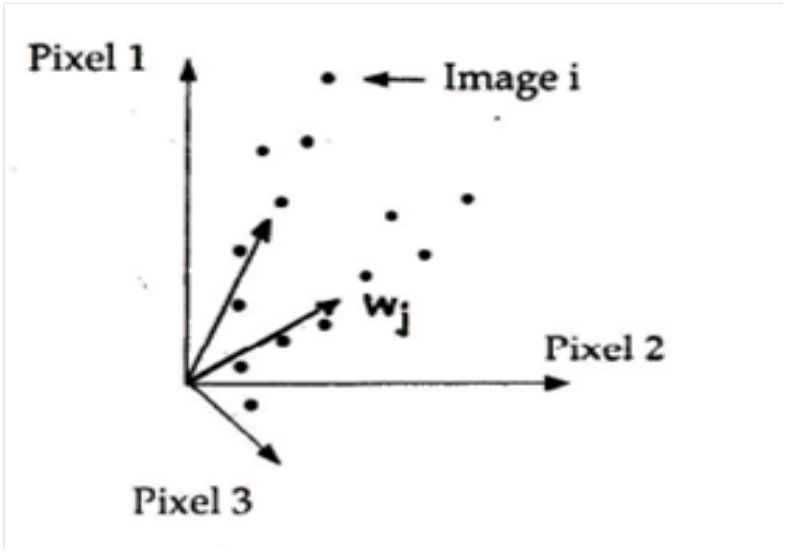


f_2: decision based on x_2 and $\alpha_2 = 0.63$

f_5: decision based on x_1 and $\alpha_5 = 0.51$

Image and multidimensional data





Each image (patch/block of the image) is a point in space of dimension $M = N^2$.

Feature Extraction

Feature Extraction: Measurements taken on the image (or blocks of the image). Examples

- Color Features: Moments (mean, variance and so on) and Histograms
- Linear Transformations
  - **Karhunen-Loève Transform (Principal Component Analysis)**
  - Fourier Transform (FFT)
  - DCT Transform (DCT)
  - **Haar Transform**
- Texture Features:
- and so on.

The application of a feature extraction algorithm leads to **dimension reduction** and better representation of the initial data.

SVD Decomposition

The singular value decomposition (SVD) of any, M by N, real matrix reads

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

Where the matrices

- $\mathbf{U}$: M by M , orthogonal columns (left eigenvectors)

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_M] \Rightarrow \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

- $\mathbf{V}$: N by N , orthogonal columns (right eigenvectors)

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_N] \Rightarrow \mathbf{V}^T\mathbf{V} = \mathbf{I}$$

- $\mathbf{\Sigma}$: M by N, diagonal, $r \leq min(M, N)$ singular values

$$\mathbf{\Sigma} = diag(\sigma_1, \sigma_2 \ldots \sigma_r)$$

PCA and SVD Decomposition

Each column of $\mathbf{X}$ is one image (after removing the *mean image*). With SVD

- Selecting the $L < min(M, N)$ columns of $\mathbf{U}$ related with the largest singular values

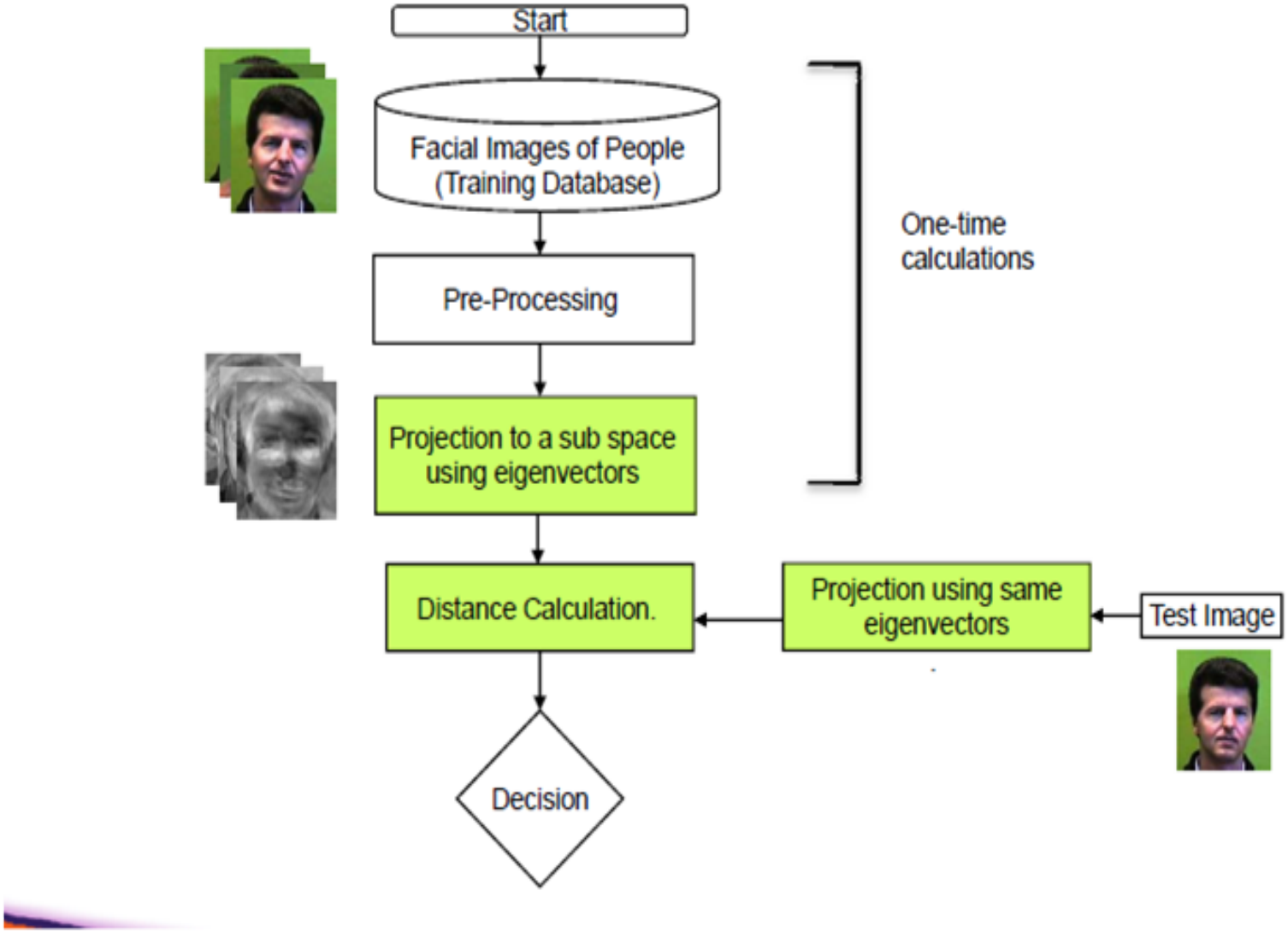
A **new coordinate system** is computed In face recognition is called the **eigenfaces**.

The **new representation** of an image $\mathbf{x}$ by projecting into the eigenvectors

$$\mathbf{z} = \mathbf{U}_L^T \mathbf{x}$$

The new representation has dimension L.

Face Recognition: PCA



Slides of Scott Gibson Clemson University

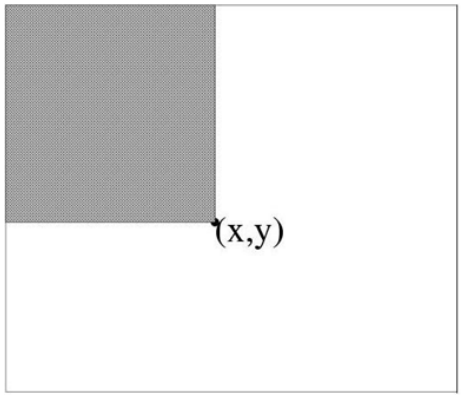
Integral Image

The integral image pixel value $S(x, y)$ at location (x, y)

- sum of the pixels above and to the left of (x, y) inclusive

$$S(x, y) = \sum_{x' < x, y' < y} I(x', y')$$

where $I(x, y)$ pixel value of original image I of size $M \times N$



Computed in one-pass over the image by applying recursively

1: Compute cumulative sum on the row x

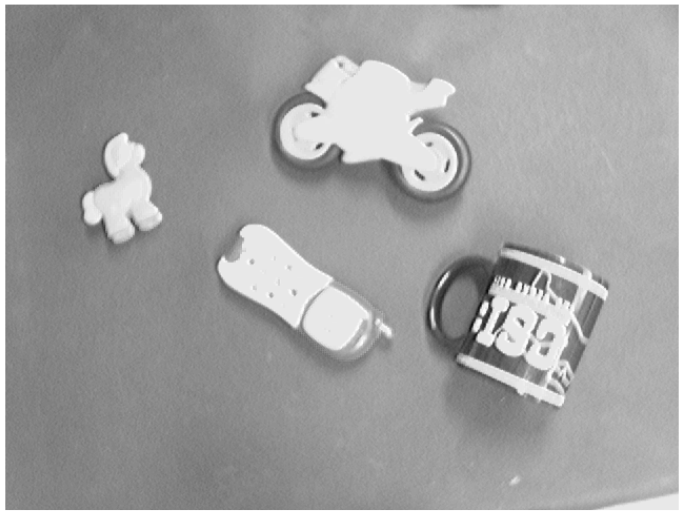
$$r(x, y) = r(x, y - 1) + I(x, y)$$

2: Compute pixel of integral image

$$S(x, y) = S(x - 1, y) + r(x, y)$$

Initialize $r(x, 0) = S(0, y) = 0$, and $x = \{1, 2...M\}, y = \{1, 2...N\}$

Example: Integral Image



The numerical meaning

IMAGE

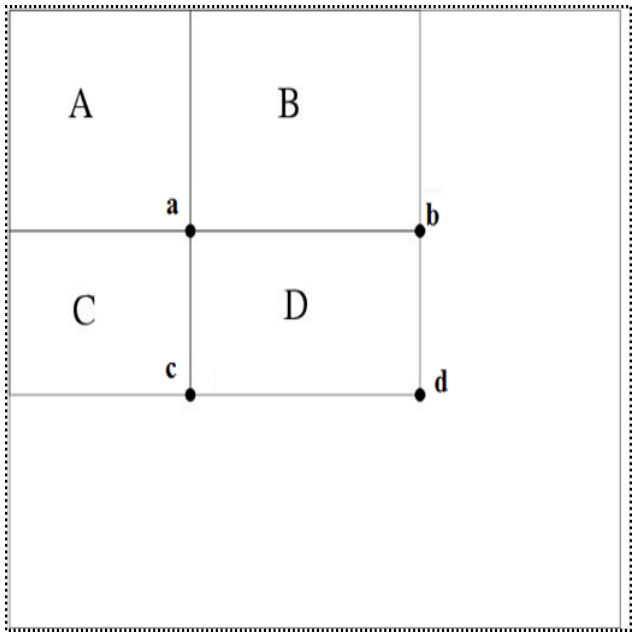
0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

INTEGRAL IMAGE

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

Areas and Integral Image

How to compute the area of region D?



The region D is defined by four points (x, y) of the integral image

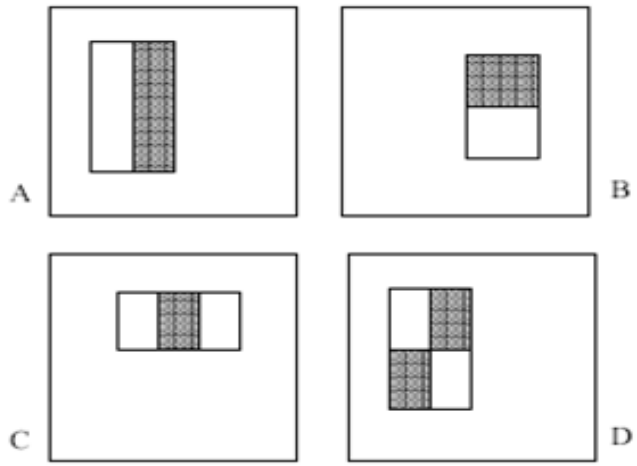
- Point **a** gives the area of region A
- Point **b** gives the area of regions A+B
- Point **c** gives the area of regions A+C
- Point **d** gives the area of regions A+B+C+D

Therefore

$$A + (A + B + C + D) - (A + B) - (A + C) \Rightarrow S(\mathbf{d}) + S(\mathbf{a}) - S(\mathbf{b}) - S(\mathbf{c})$$

Feature Extraction: Haar Like features

Haar features are extracted from **sub windows** of a sample image.



Each filter is composed of several rectangular regions where

- the white area is associated with -1
- the gray area with 1

Overlaying them on image I means that white areas are subtracted from the black ones.

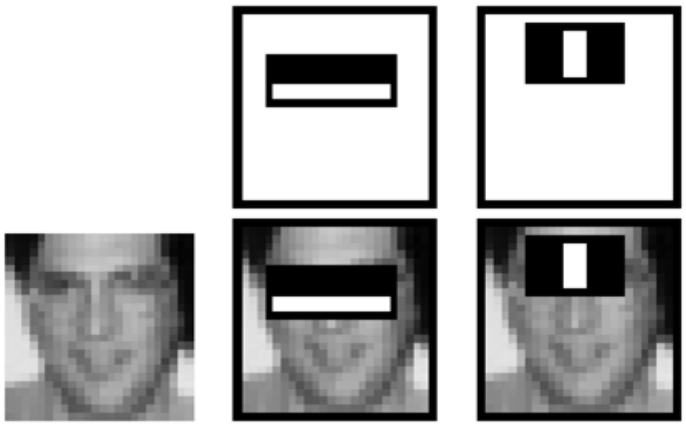
Easy to compute in integral image S

- two rectangular areas: 6 different points on S
- three rectangular areas: 8 different points on S
- four rectangular areas: 9 different points on S

Haar Like features and Face Recognition

In face recognition systems (Viola's method)

The base size for a sub window is 24 by 24 pixels



Each of the four feature types are **scaled and shifted** across all possible combinations

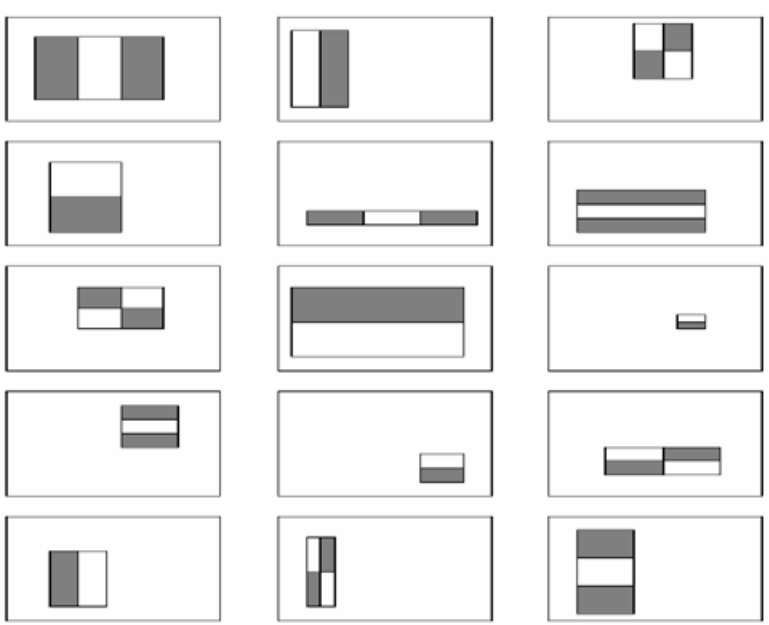
Relevant feature

Irrelevant feature

- Not all features are relevant

Haar Like features and Face Recognition

Classifier is trained on sub-window images of fixed size (Viola uses $24x24$)

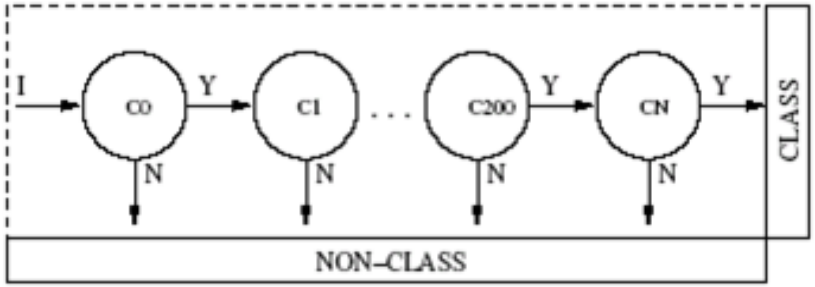


In a sub-window there are 160000 possible features to be calculated.

Cascade of Boosted Classifiers

Classifier is trained on images of fixed size (Viola uses 24×24)

- Sequence of boosted classifiers with constantly increasing complexity
- Chained into a cascade with the simpler classifiers going first.

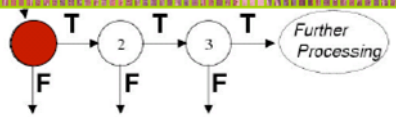


Detection is done by sliding a search window (of size 24×24)

- Goal of Cascade Classifiers: Quick rejection of sub windows when testing.

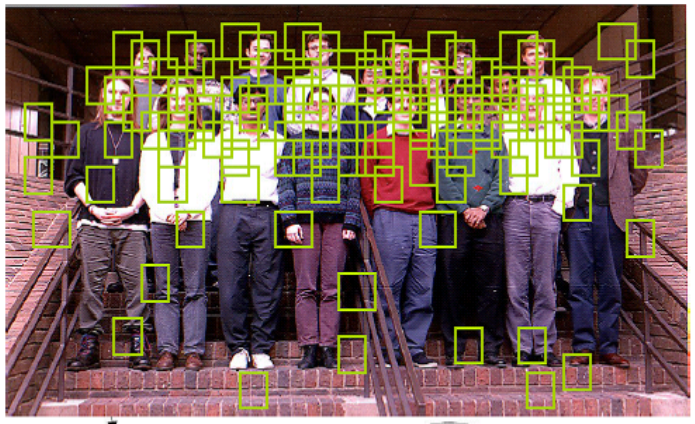
Cascade of Boosted Classifiers

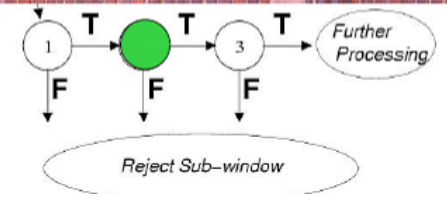
Fleuret and Geman 2001, Viola and Jones 2001



8192
search windows

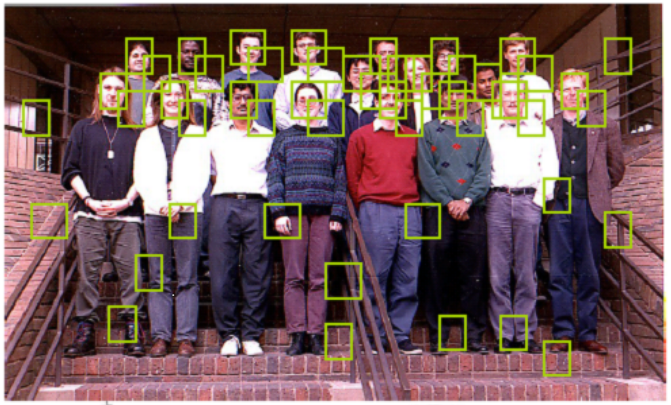
Fleuret and Geman 2001, Viola and Jones 2001

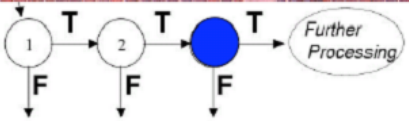




188
search windows

Fleuret and Geman 2001, Viola and Jones 2001





48
search windows

Bibliography

- Richard Szeliski- Computer Vision: Algorithms and Applications (chapter 14). (available online: `http://szeliski.org/Book/`)
- Theodoridis, Koutoumbas- Pattern recognition- Academic Press
  - chapter 3: linear classifiers
  - chapter 8: template matching

Images of toys : courtesy Aneesh Chauhan (ex-PhD student in IEETA).