Lecture 2
AdaBoost and Cascade Structure
(with a case on face detection)

Lin ZHANG, PhD
School of Computer Science and Technology
Tongji University
Fall 2025

Any faces contained in the image?

Who are they?

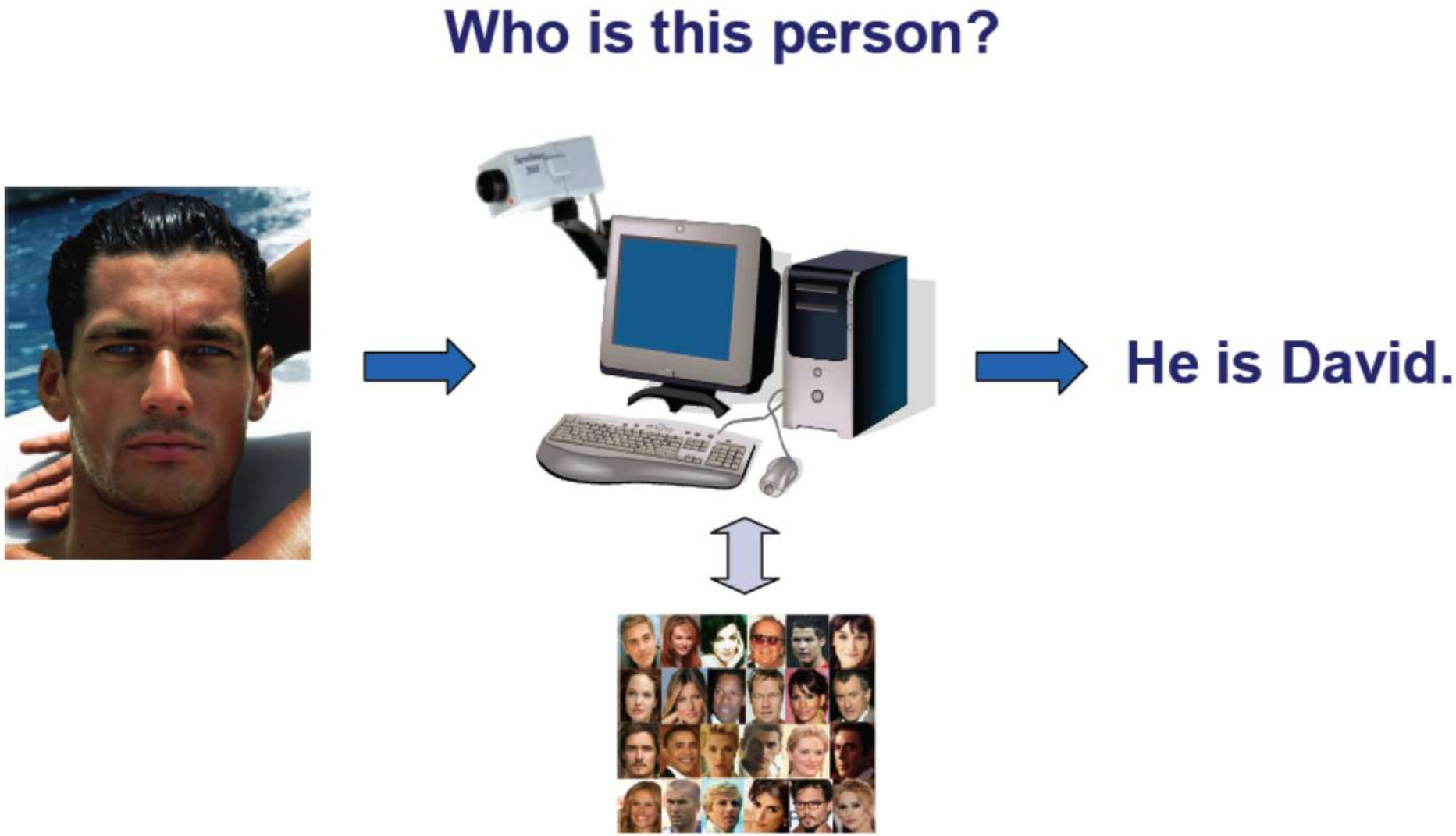
Overview

- Face recognition problem
  - Given a still image or video of a scene, identify or verify one or more persons in this scene using a stored database of facial images

Who is she?

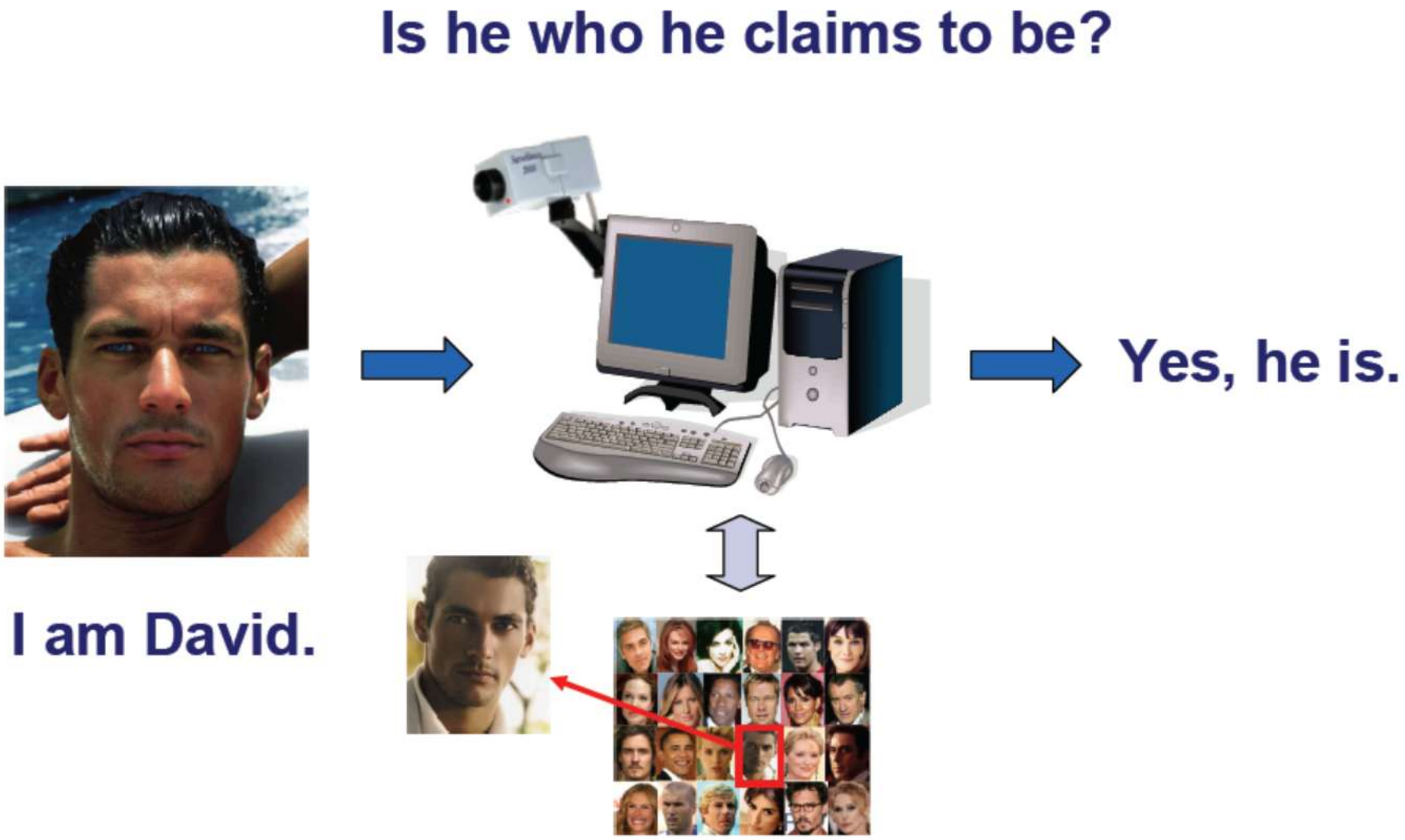
Overview

- Face identification



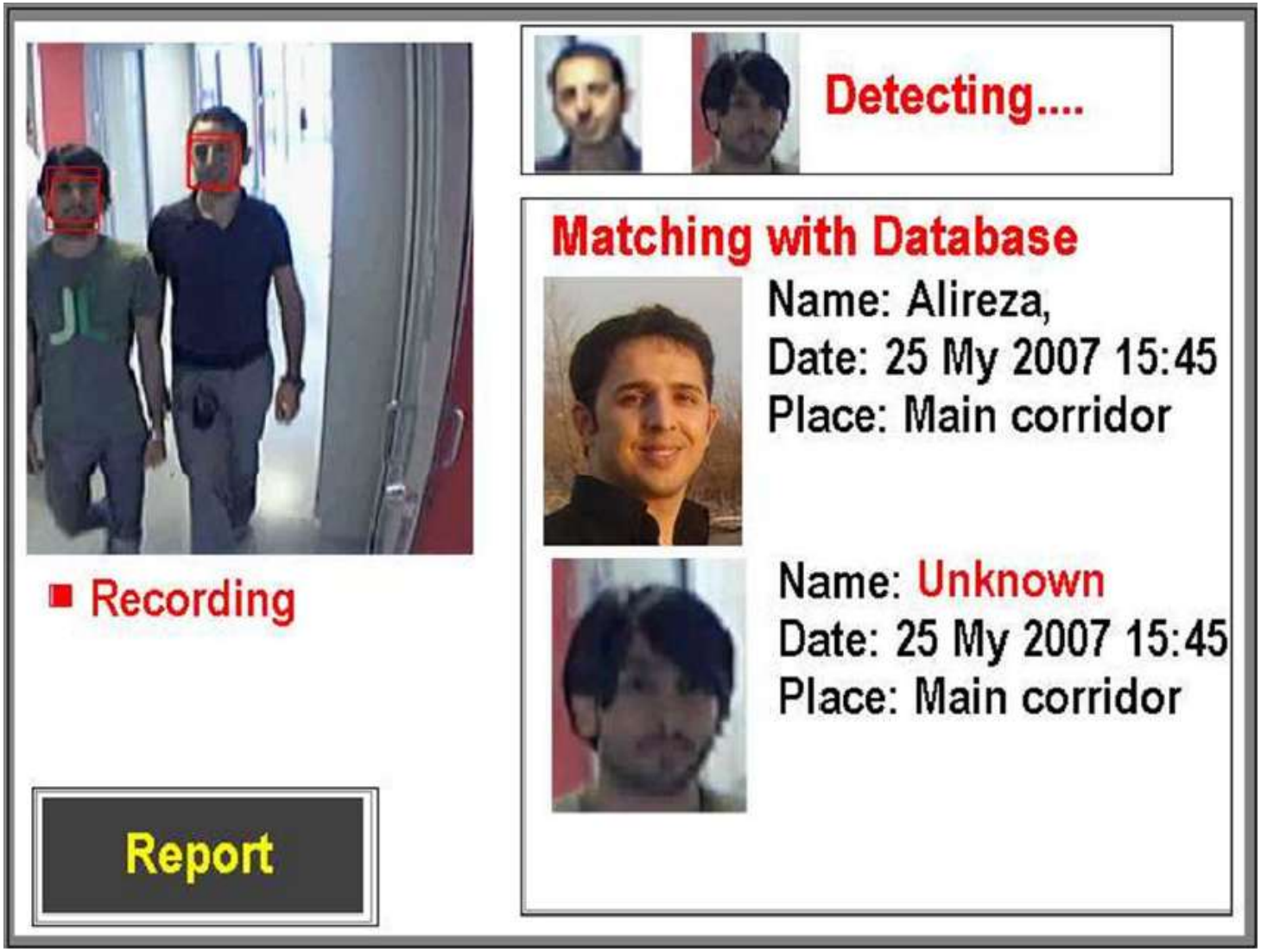
Overview

- Face verification



Overview

- Applications of face detection&recognition



Intelligent surveillance

Overview

- Applications of face detection&recognition

Hong Kong—Luohu, border control

E-channel

Overview

- Applications of face detection&recognition

National Stadium, Beijing Olympic Games, 2008

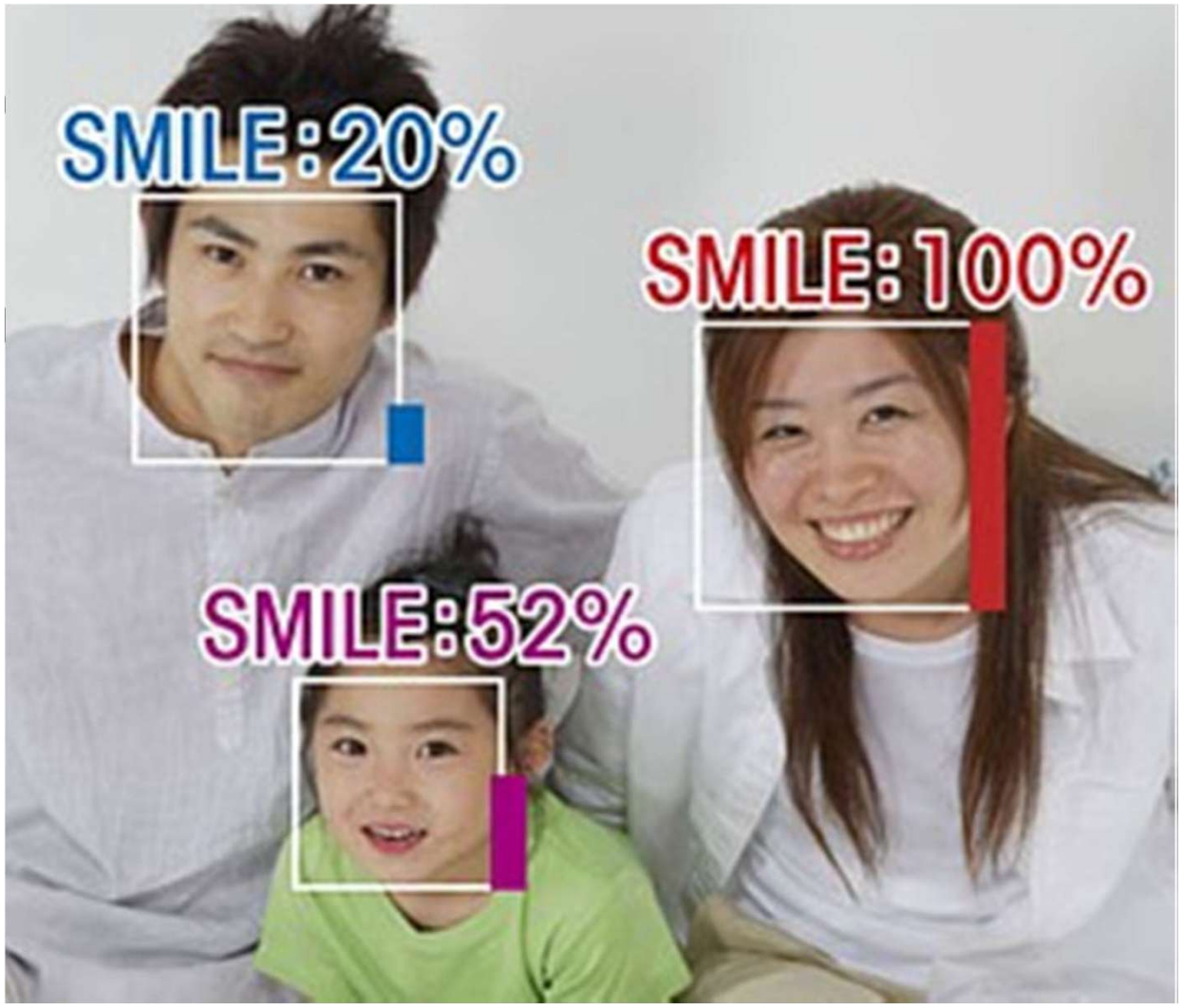
Overview

- Applications of face detection&recognition

Check on work attendance

Overview

- Applications of face detection&recognition



Smile detection: embedded in most modern cameras

Overview

- Why is face recognition so difficult?
  - Intra-class variance and inter-class similarity

Images of the same person

Overview

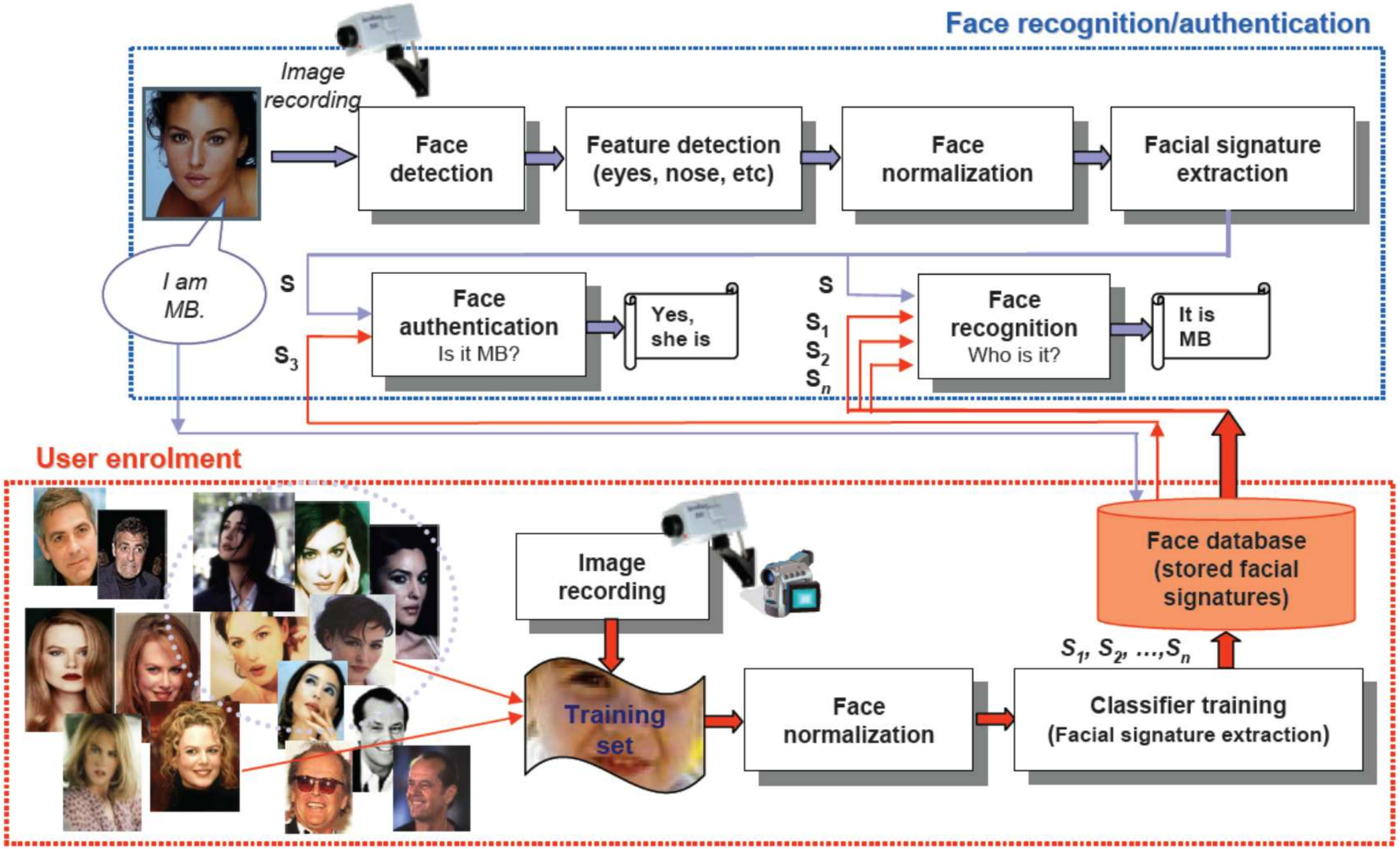
- Why is face recognition so difficult?
  - Intra-class variance and inter-class similarity

Images of twins

Overview

Who are they?

Overview-General Architecture



Introduction

- Identify and locate human faces in an image regardless of their
  - Position
  - Scale
  - Orientation
  - pose (out-of-plane rotation)
  - illumination

Introduction

Where are the faces, if any?

Introduction

- Appearance based methods
  - Train a classifier using positive (and usually negative) examples of faces
  - Representation: different appearance based methods may use different representation schemes
  - Most of the state-of-the-art methods belong to this category

The most successful one: Viola-Jones method!

VJ is based on AdaBoost classifier

AdaBoost (Adaptive Boosting)

- It is a machine learning algorithm[1]
- AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers
- The classifiers it uses can be weak, but as long as their performance is slightly better than random they will improve the final model

[1] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", Journal of Computer and System Sciences,1995

AdaBoost (Adaptive Boosting)

- AdaBoost is an algorithm for constructing a ”strong” classifier as a linear combination of simple weak classifiers,

$$H(\mathbf{x}) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) \in \{-1, +1\}$$

where $h_t(\mathbf{x}) = \operatorname{sgn}\left(f_t(\mathbf{x})\right) \in \{-1, +1\}$

- Terminology
  - $h_t(\mathbf{x})$ is a weak or basis classifier and $f_t(\mathbf{x})$ is the associated classification function
  - $H(\mathbf{x})$ is the final strong classifier and its associated classification function is $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

AdaBoost (Adaptive Boosting)

- AdaBoost is an iterative training algorithm, the stopping criterion depends on concrete applications
- For each iteration t
  - A new weak classifier $h_t(\mathbf{x})$ is added based on the current training set
  - Modify the weight for each training sample; the weight for the sample being correctly classified by $h_t(\mathbf{x})$ will be reduced, while the sample being misclassified by $h_t(\mathbf{x})$ will be increased

AdaBoost (algorithm for binary classification)

Given:

- Training set $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)$, where $y_i \in \{-1, +1\}$

Initialize weights for samples $D_1(i) = 1/m$

For $t = 1{:}T$

Train weak classifiers based on training set and the D_t

find the best weak classifier h_t with error $\varepsilon_t = \sum_{i=1}^{m} D_t(i)\left[h_t(\mathbf{x}_i) \neq y_i\right]$

if $\varepsilon_t \geq 0.5$, stop;

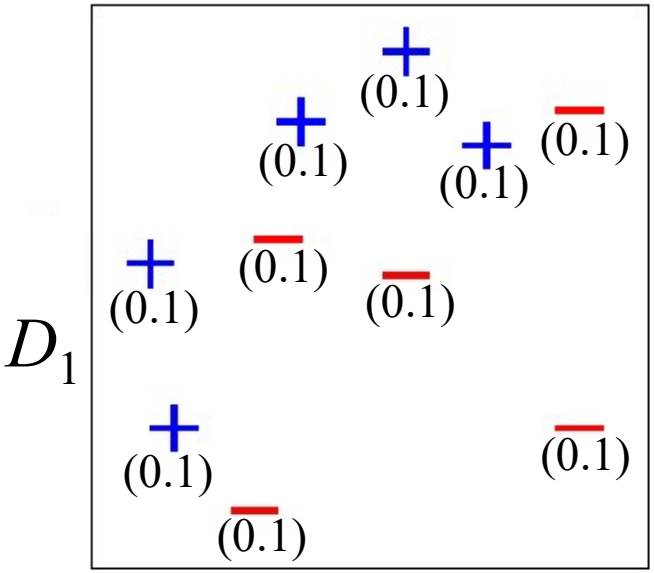
set $\alpha_t = 0.5 \ln\left(\left(1 - \varepsilon_t\right) / \varepsilon_t\right)$

update weights for samples $D_{t+1}(i) = \dfrac{D_t(i)\exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right)}{Denom}$

Outputs the final classifier,

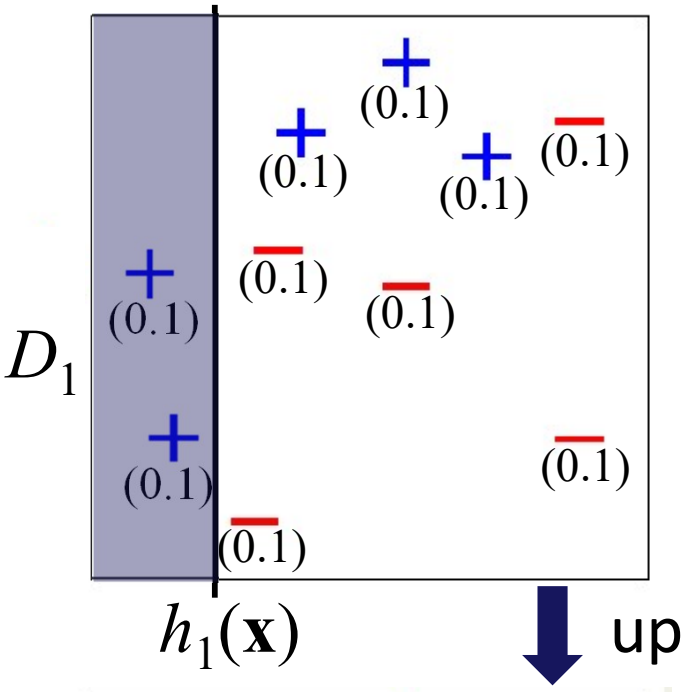
$$H(\mathbf{x}) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

AdaBoost—An Example



- 10 training samples
- Weak classifiers: vertical or horizontal lines
- Initial weights for samples $D_1(i) = 0.1, i = 1 \sim 10$
- Three iterations

AdaBoost—An Example



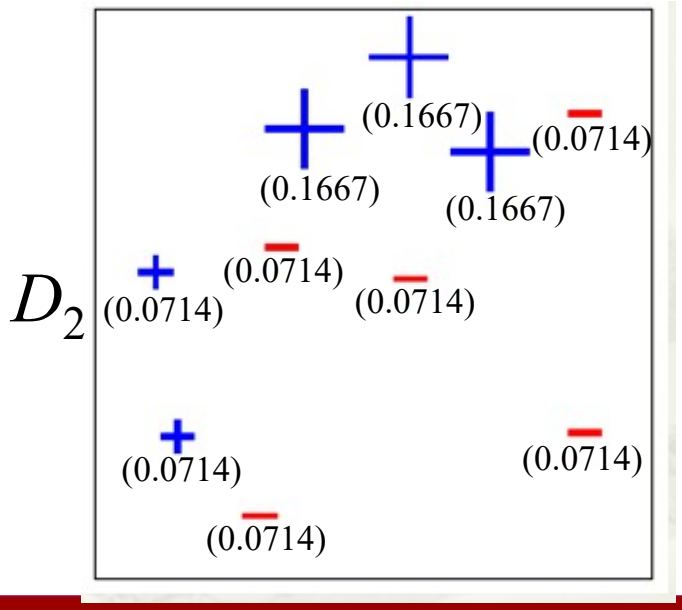
After iteration one

Get the weak classifier $h_1(\mathbf{x})$

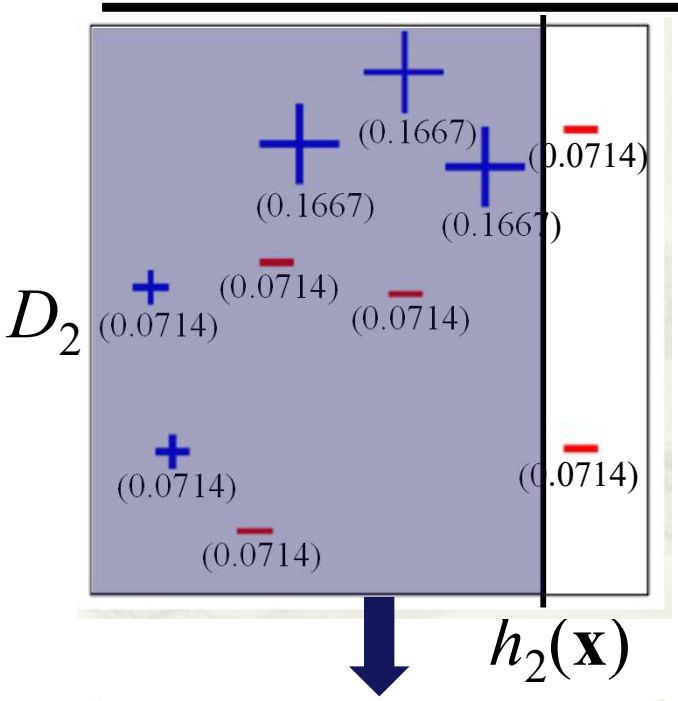
$$\varepsilon_1 = 0.3$$

$$\alpha_1 = \frac{1}{2}\ln\frac{1-\varepsilon_1}{\varepsilon_1} = 0.4236$$

update weights



AdaBoost—An Example



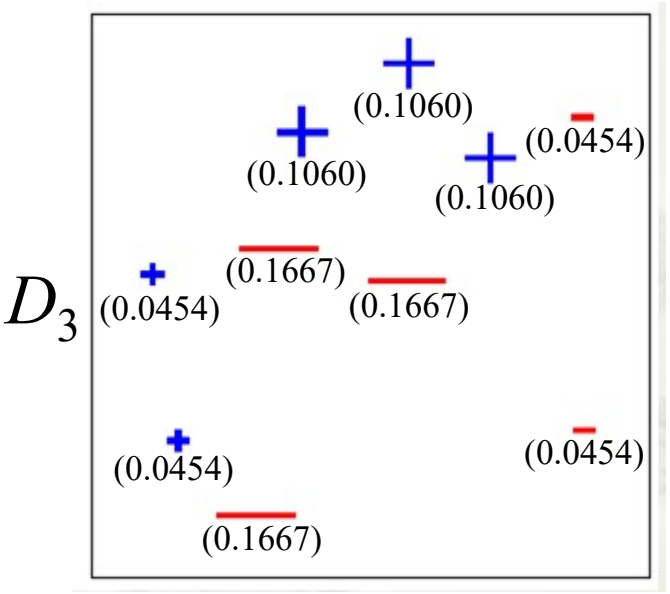
After iteration 2

Get the weak classifier $h_2(\mathbf{x})$

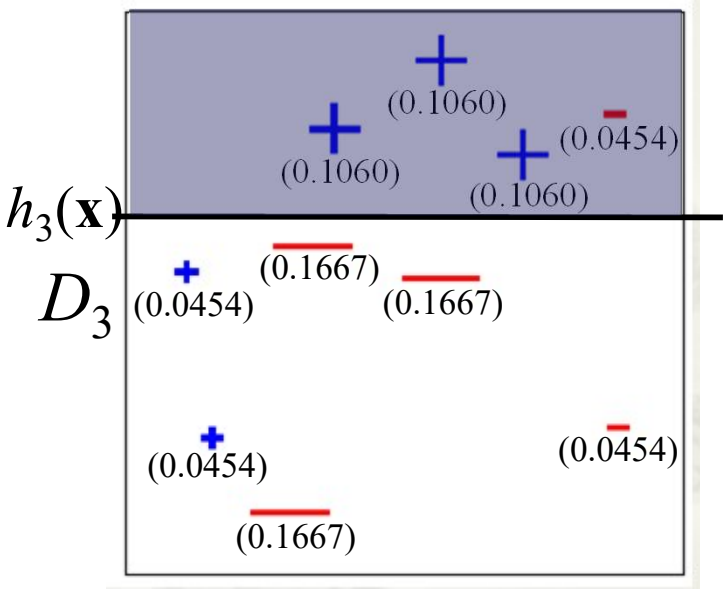
$$\varepsilon_2 = 0.2142$$

$$\alpha_2 = \frac{1}{2}\ln\frac{1-\varepsilon_2}{\varepsilon_2} = 0.6499$$

update weights



AdaBoost—An Example



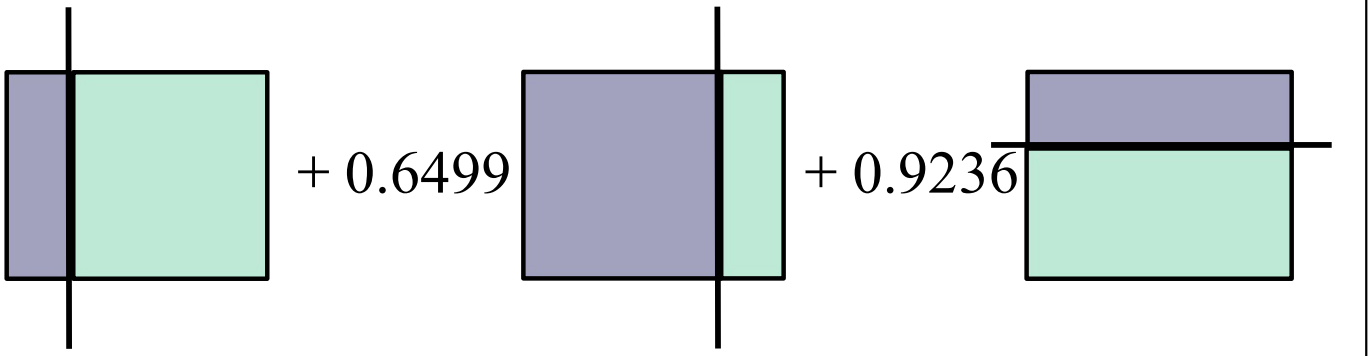
After iteration 3

Get the weak classifier $h_3(\mathbf{x})$

$\varepsilon_3 = 0.1362$

$$\alpha_3 = \frac{1}{2}\ln\frac{1-\varepsilon_3}{\varepsilon_3} = 0.9236$$

$$H(\mathbf{x}) = \mathrm{sgn}\left[0.4236 + 0.6499 + 0.9236\right]$$



Now try to classify the 10 samples using $H(\mathbf{x})$

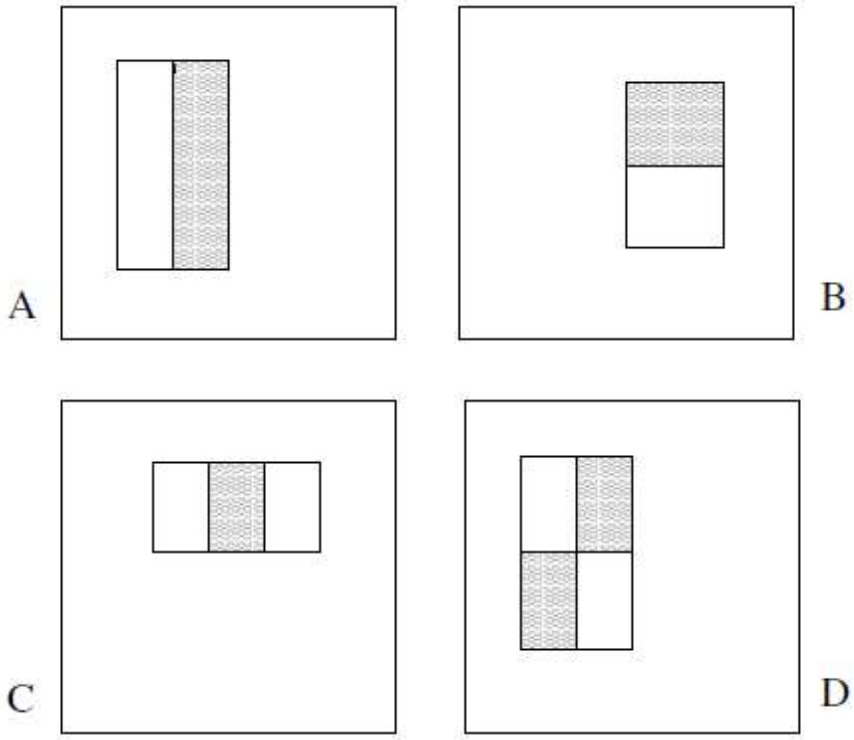
Viola-Jones face detection

- VJ face detector[1]
  - Haar-like features are proposed and computed based on *integral image*; they act as “weak” classifiers
  - Strong classifiers are composed of “weak” classifiers by using AdaBoost
  - Many strong classifiers are combined in a cascade structure which dramatically increases the detection speed

[1] P. Viola and M.J. Jones, “Robust real-time face detection", IJCV, 2004

Haar features

- Compute the difference between the sums of pixels within two (or more) rectangular regions



Example Haar features shown relative to the enclosing face detection window

Haar features

- Integral image
  - The integral image at location (x, y) contains the sum of all the pixels above and to the left of x, y, inclusive:

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

where $i(x, y)$ is the original image

  - By the following recurrence, the integral image can be computed in one pass over the original image

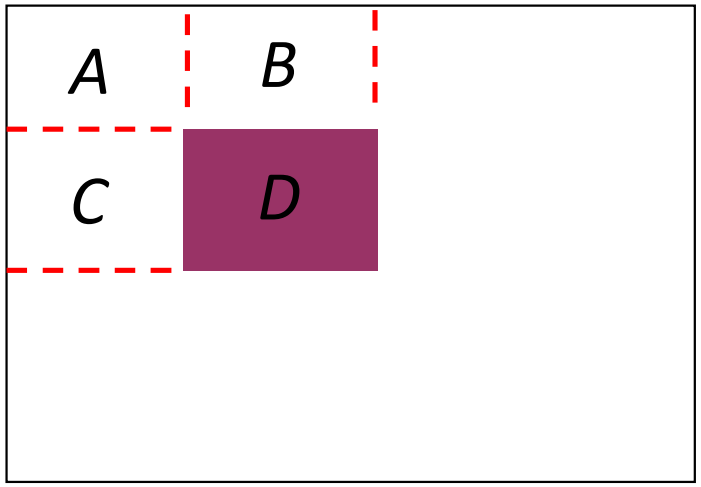
$$s(x,y) = s(x,y-1) + i(x,y)$$

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

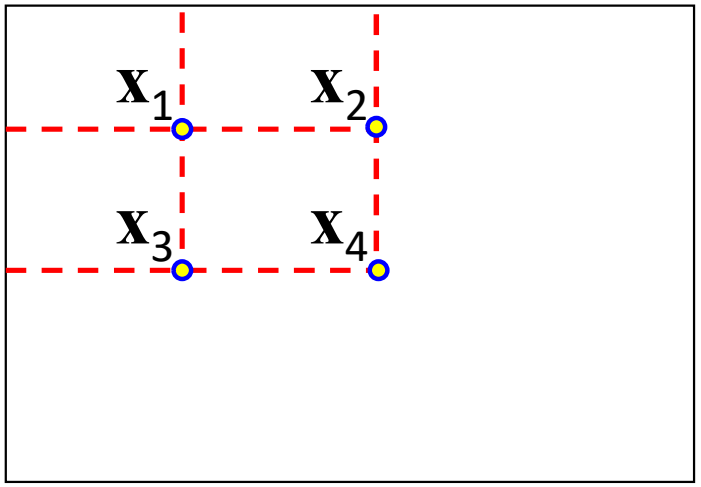
where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$, and $ii(-1, y) = 0$

Haar features

- Haar feature can be efficiently computed by using integral image



original image $i(x, y)$



integral image $ii(x, y)$

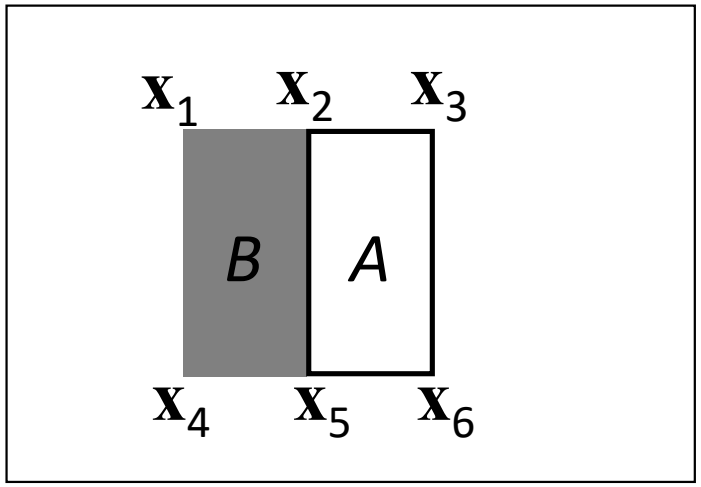
Actually,

$$ii(\mathbf{x}_1) = A$$
$$ii(\mathbf{x}_2) = A + B$$
$$ii(\mathbf{x}_3) = A + C$$
$$ii(\mathbf{x}_4) = A + B + C + D$$

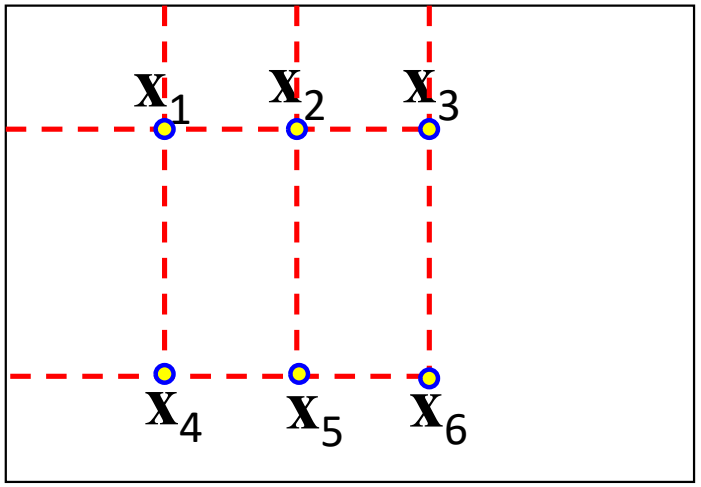
$$\Rightarrow \quad D = ii(\mathbf{x}_4) + ii(\mathbf{x}_1) - ii(\mathbf{x}_2) - ii(\mathbf{x}_3)$$

Haar features

- Haar feature can be efficiently computed by using integral image

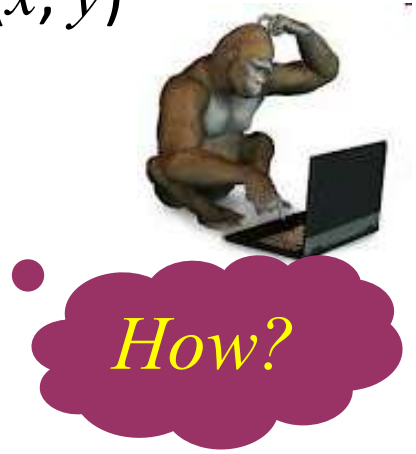


original image $i(x, y)$



integral image $ii(x, y)$

How to calculate A-B in integral image?



Haar features

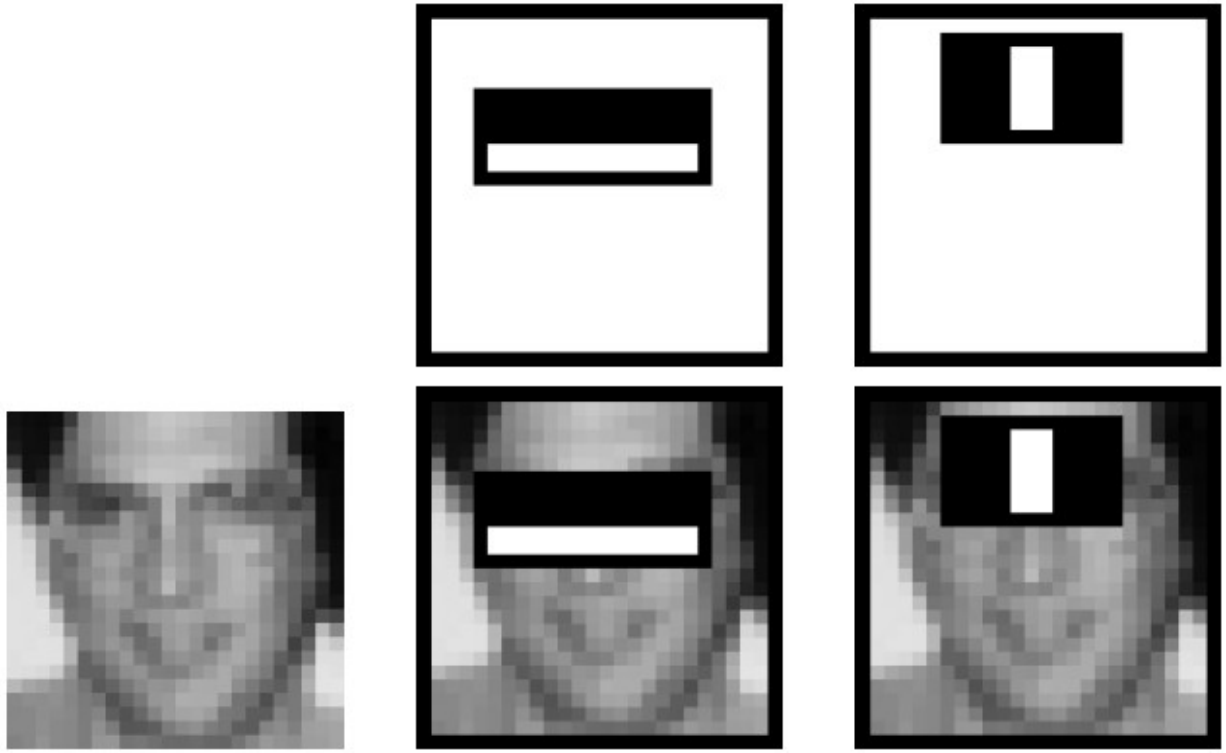
- Given a detection window, tens of thousands of Haar features can be computed
- One Haar feature is a weak classifier to decide whether the underlying detection window contains face

$$h(x, f, p, t) = \begin{cases} 1, & pf(x) < p\theta \\ -1, & otherwise \end{cases}$$

where x is the detection window, f defines how to compute the Haar feature on window x, p is 1 or -1 to make the inequalities have a unified direction, θ is a threshold

- f can be determined in advance; by contrast, p and θ are determined by training, such that the minimum number of examples are misclassified

Haar features



The first and second best Haar features. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

From weak learner to stronger learner

- Any single Haar feature (thresholded single feature) is quite weak on deciding whether the underlying detection window contains face or not
- Many Haar features (weak learners) can be combined into a strong learner by using Adaboost
- However, the most straightforward technique for improving detection performance, adding more features to the classifier, directly increases computation cost

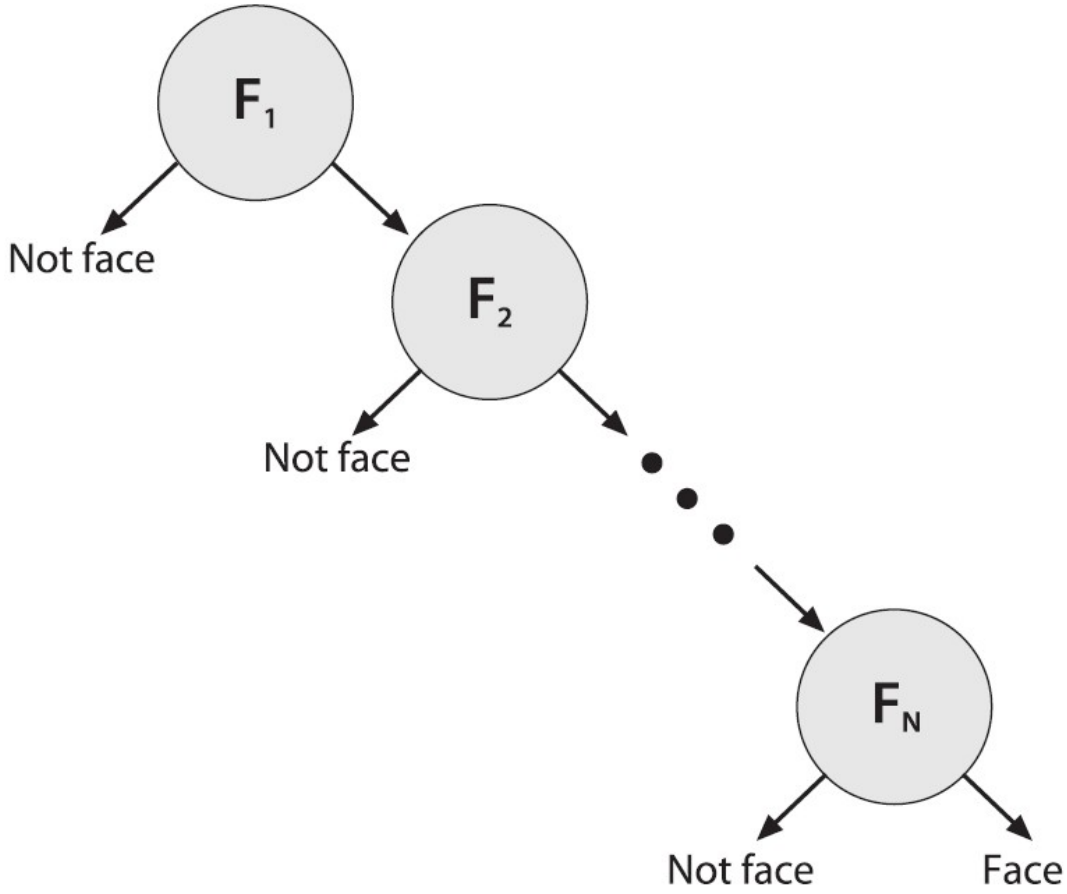
Construct a cascade classifier

Cascade classifier

- Motivations
  - Within an image, most sub-images are non-face instances
  - Use smaller and efficient classifiers to reject many negative examples at early stage while detecting almost all the positive instances
  - Simpler classifiers are used to reject the majority of sub-windows; more complex classifiers are used at later stage to examine difficult cases

Cascade classifier

- Our aim: rejection cascade



The initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. After several stages, the number of remained detection windows has been reduced radically

Cascade classifier

- Terminologies
  - Detection rate:

$$\frac{\textit{true positives}\ \text{(real faces detected)}}{\textit{true positives + false negatives}\ \text{(number of all faces)}}$$

  - False positive rate (FPR),

$$\frac{\textit{false positives}\ \text{(false faces detected)}}{\textit{false positives + true negatives}\ \text{(number of non-face samples)}}$$

Cascade classifier

Given a trained cascade of classifiers, the FPR of the cascade is,

$$F = \prod_{i=1}^{K} f_i$$

where K is the number of stages, and f_i is the FPR of the ith stage on the samples that get through to it

The detection rate of the cascade is,

$$D = \prod_{i=1}^{K} d_i$$

where d_i is the detection rate of the ith stage on the samples that get through to it

Data used for training

- A large number of normalized face samples
  - Having the same size

- A large number of non-face samples

Training Strategy

- VJ cascaded face detector training strategy
  - User sets the maximum acceptable false positive rate and the minimum acceptable detection rate for each layer
  - Each layer of cascade is trained by AdaBoost with the number of features used being increased until the target detection and false positive rates are met for this level
  - The detection rate and FPR are determined by testing the current cascade detector on a validation set
  - If the overall target FPR is not met then another layer is added to the cascade
  - The negative set for training subsequent layers is obtained by collecting all false detections found by running the current cascade on a set of images containing no face instances

- User selects values for f, the maximum acceptable false positive rate per layer and d, the minimum acceptable detection rate per layer.
- User selects target overall false positive rate, F_{target}.
- P = set of positive examples
- N = set of negative examples
- $F_0 = 1.0$; $D_0 = 1.0$
- $i = 0$
- while $F_i > F_{target}$
  - $i \leftarrow i + 1$
  - $n_i = 0$; $F_i = F_{i-1}$
  - while $F_i > f \times F_{i-1}$
    - $n_i \leftarrow n_i + 1$
    - Use P and N to train a classifier with n_i features using AdaBoost
    - Evaluate current cascaded classifier on validation set to determine F_i and D_i.
    - Decrease threshold for the ith classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects F_i)
  - $N \leftarrow \emptyset$
  - If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set N

Viola-Jones face detection

- Implementation
  - VJ face detector has been implemented in OpenCV and Matlab
  - OpenCV has also provided the training result from a frontal face dataset and the result is contained in “haarcascade_frontalface_alt2.xml”
  - A demo program has been provided on our course website: FaceDetectionEx

Viola-Jones face detection

- Demo time: some examples

original image

VJ face detection result

Viola-Jones face detection

- Demo time: some examples

Viola-Jones face detection

- Summary
  - Three main components
    - Integral image: efficient convolution
    - Use Adaboost for feature selection
    - Use Adaboost to learn the cascade classifier
  - Properties
    - Fast and fairly robust; runs in real time
    - Very time consuming in training stage (may take days in training)
    - Requires lots of engineering work


Q+A