EE 6886: Topics in Signal Processing -- Multimedia Security System

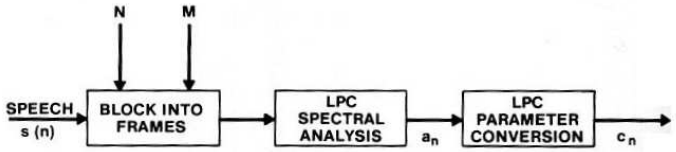
Lecture 9: Biometric Authentication (II) – Speaker and Face Recognition

Ching-Yung Lin
Department of Electrical Engineering
Columbia University, New York, NY 10027

Course Outline

- Multimedia Security :
  - Multimedia Standards – Ubiquitous MM
  - Encryption and Key Management – Confidential MM
  - Watermarking – Uninfringible MM
  - Authentication – Trustworthy MM
- Security Applications of Multimedia:
  - Audio-Visual Person Identification – Access Control, Identifying Suspects
  - Surveillance Applications – Abnormality Detection
  - Media Sensor Networks – Event Understanding, Information Aggregation

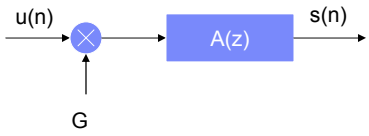
Linear Predictive Coding (LPC) Analysis



- LPC model:

$s(n) = - a_1 s(n-1) – a_2 s(n-2) - \ldots - a_p s(n-p) + G u(n).$

$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_p z^{-p}$



For each speech frame (10 msec or 20 msec), we want to find a set of a_1 .. a_p that forms the LPC feature vector of the frame.

LPC (II)

$$s(n) = -\sum_{k=1}^{P} a_k s(n-k) + Gu(n).$$

- Assume each frame has N samples (m = 0 … N-1). And let n represents a frame starts at time n. We want to find a vector **â** to minimize the mean average error :

$$\begin{aligned}\hat{\mathbf{a}} &= \arg\min_{\mathbf{a}} \sum_{m=0}^{N+P-1} e_n^2(m) \\ &= \arg\min_{\mathbf{a}} \sum_{m=0}^{N+P-1} (s_n(m) - \tilde{s}_n(m))^2 \\ &= \arg\min_{\mathbf{a}} \left\{ \sum_{m=0}^{N+P-1} \left[s_n(m) + \sum_{k=1}^{P} a_k s_n(m-k) \right]^2 \right\},\end{aligned}$$

- We can get the predictor coefficients by differentiating the error function with respect to each a_k and set the result to zero:

$$\frac{\partial E_n}{\partial a_k} = 0$$

LPC (III)

- We can then get results as:

$$\begin{bmatrix} r_n(0) & r_n(1) & \cdots & r_n(P-1) \\ r_n(1) & r_n(0) & \cdots & r_n(P-2) \\ r_n(2) & r_n(1) & \cdots & r_n(P-3) \\ \cdots & \cdots & \cdots & \cdots \\ r_n(P-1) & r_n(P-2) & \cdots & r_n(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \cdots \\ \hat{a}_P \end{bmatrix} = \begin{bmatrix} -r_n(1) \\ -r_n(2) \\ \\ \cdots \\ -r_n(P) \end{bmatrix}.$$

where

$$r_n(k) = \sum_{m=0}^{N-1-k} s_n(m) s_n(m+k).$$

We only estimate the autocorrelation function of the samples between 0 ~ N-1-k, i.e., ignoring the boundary samples.

LPC (IV)

- Example:
  - Let s(0) = 2.0, s(1) = -1.0, s(2) = 1.0, and s(3) = s(4) = … = s(N-1) = 0.0.
  - Use P=2.
  - Thus,

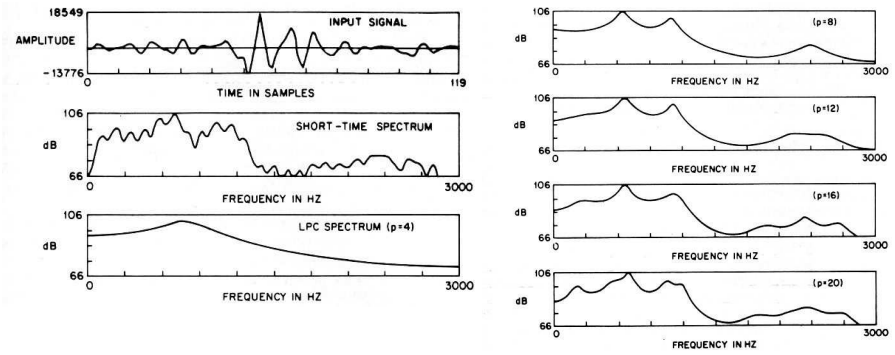
$$\begin{aligned} r(0) &= \sum_{m=0}^{N-1} s^2(m) \\ &= s^2(0) + s^2(1) + s^2(2) + \cdots + 0 = 4 + 1 + 1 = 6 \\ r(1) &= \sum_{m=0}^{N-2} s(m)s(m+1) \\ &= s(0)s(1) + s(1)s(2) + s(2)s(3) + 0 \cdots + 0 = 2(-1) + (-1)(1) = -3 \\ r(2) &= \sum_{m=0}^{N-3} s(m)s(m+2) \\ &= s(0)s(2) + s(1)s(3) + s(2)s(4) + 0 \cdots + 0 = 2(1) + (-1)(0) = 2. \end{aligned}$$

Then,

$$\begin{bmatrix} 6 & -3 \\ -3 & 6 \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \Rightarrow \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \end{bmatrix} = \begin{bmatrix} 0.44 \\ -0.11 \end{bmatrix}.$$

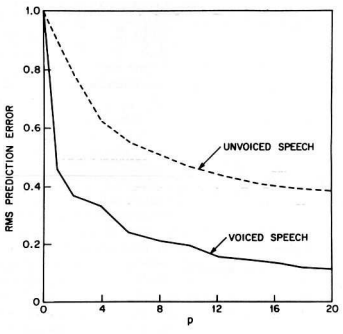
Prediction Noise and the Order of LPC

- The larger the order the fewer the prediction error.



Selection of the order of LPC

- Order v.s. prediction error:
  - The larger the order the fewer the prediction error.
  - The unvoiced speech is harder to predict than the voiced speech.
  - A sharp decrease in normalized prediction error occurs for small values of P (e.g., 1 – 4).
  - It is generally acknowledged that values of P on the order of 8 – 10 are reasonable for most speech-recognition and speaker recognition applications.



Linear Predictive Cepstral Coefficient (LPCC)

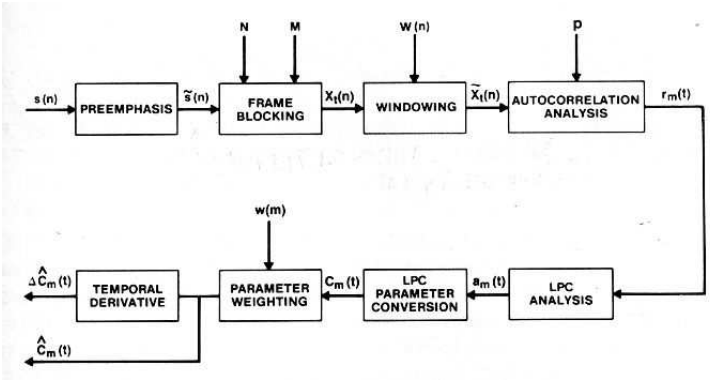
- It was found that smoothing the LPC envelopes by cepstral processing can provide a more consistent representation of a speaker's vocal tract characteristics from one utterance to another.
- The LPCC coefficients c_n can be computed directly from LPC a_k as follows:

$$
\begin{aligned}
c_0 &= \ln G \\
c_n &= -a_n - \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) c_k a_{n-k} \qquad 1 \le n \le P \\
c_n &= -\sum_{k=1}^{n-1} \left(\frac{k}{n}\right) c_k a_{n-k} \qquad n > P,
\end{aligned}
$$

where n is usually chosen between 0 ~ 1.5P-1

Enhanced System for LPC-based Feature Extraction

- Block diagram of LPC processor for speaker recognition



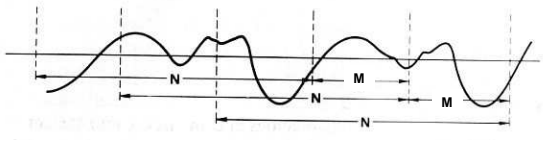
Preemphasis

- The digitized speech signal, s(n), is put through a low-order digital system to spectrally flatten the signal and to make it less susceptible to finite precision effects later in the signal processing.
- The most widely used preemphasis network is the fixed first-order system:

$$H(z) = 1 - a z^{-1}, \quad a \text{ is around } 0.95.$$

Frame Blocking

- Overlapping blocks can be used to make the resulting LPC spectral estimates more correlated from frame to frame.
- A typical case is that M = (1/3) N.



Windowing

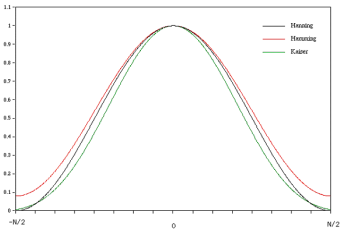
- Hamming window is the most commonly used filter used for windowing.

$$w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N-1})$$

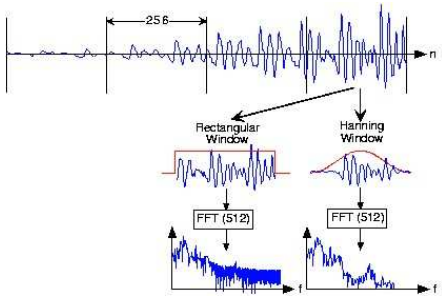
0 <= n <= N-1.

- It is used to smooth the temporal discontinuity when calculating FFT.
- Hanning Window:

$$w(n) = 0.5 - 0.5\cos(\frac{2\pi n}{N-1})$$



Reference source: http://astronomy.swin.edu.au/~pbourke/analysis/windows/



Reference source: http://cnx.rice.edu/content/m11734/latest/

Parameter Weighting

- Low-order cepstral coefficients are sensitive to overall spectral slope.
- High-order cepstral coefficients are sensitive to noise.
- A bandpass filter may be applied on the cepstral domain:

$$w_m = \left[1 + \frac{Q}{2}\sin(\frac{\pi m}{Q})\right] \qquad 1 \le m \le Q$$

Temporal Cepstral Derivatives

- The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal.
- An improved representation can be obtained by extending the analysis to include information about the temporal cepstral derivative.
- An approximation of the first- and second order LPCCs (or MFCCs) can be computed by using polynomial approximation:

$$\Delta c_n = \frac{\sum_{k=-L}^{L} k \cdot c_{n+k}}{\sum_{k=-L}^{L} |k|} \qquad \Delta\Delta c_n = \frac{\sum_{k=-L}^{L} k^2 \cdot c_{n+k}}{\sum_{k=-L}^{L} k^2}$$

  - L is typically equal to 3.
  - Usually, the original, the first order and the second order cepstral coefficeints can be combined together to form a vector of 3Q components.

Other extensions for speaker recognition

- Dynamic time warping (DTW) – for speaker recognition using the same two utterances at the training set and the test set
- Hidden Markov Modeling (HMM) – similar to the purpose of DTW. Performs a little bit better.
- The above techniques are mainly used for text-dependent speaker verifications.
- For text-independent speaker verification, Vector Quantization, Gaussian Mixture Models and Neural Networks are the most commonly used methods.

Classification – Learning Method

- Supervised Learning:
  - Teacher involves – i.e., there is a training set in which the input/output pair is available
  - Examples:
    - Support Vector Machine: binary classifier
    - Gaussian Mixture Model + Maximum Likelihood Detection

- Unsupervised Learning:
  - There is no known input/output pairs.
  - Examples:
    - K-mean clustering (hard cluster boundary) – each cluster is a class.
      - Very similar to the vector quantization (VQ) algorithm
      - Use the nearest-neighbor criteria.
    - Gaussian Mixture Model (soft cluster boundary) – usually trained independently from the feature vectors of "a" known class. All clusters combined together is a class.

K-Means and VQ algorithms

- Cluster a given data set $X = \{x_t; t=1, \ldots T\}$ into K groups, each represented by its centroid denoted by $\mu^{(j)}$, $j = 1, \ldots, K$.
- In our homework 3, we already know the ground truth of the training data.
  ➔ We don't need to do this clustering on training.
- The nearest-neighbor rule assigns a pattern x to the class associated with its nearest centroid, say $\mu^{(i)}$
- In the case of unsupervised learning, the goal is to find out K centroids that minimize the following sum of squared errors:

$$E(X) = \sum_t \left\| x_t - \mu_t \right\|^2$$

Steps for K-Mean algorithm

- K-Mean algorithm:

1. *Determine the membership of a data pattern:*

$$\mathbf{x} \in X_k \quad \text{if} \quad \|\mathbf{x} - \mu_k\| < \|\mathbf{x} - \mu_j\| \quad \forall j \neq k.$$

2. *Updating the representation of the cluster:* In a clustering process, the inclusion (or removal) of a new pattern in a cluster (or from a cluster) affects the representation (e.g., the centroid or variance) of the cluster. Therefore, the centroid should be updated based on the new membership:

$$\mu_j = \frac{1}{N_j} \sum_{\mathbf{x} \in X_j} \mathbf{x}.$$

- The Expectation-Maximization (EM) scheme can be seen as a generalized version of K-means clustering.
- The main difference hinges on the notion of a hard-versus-soft membership.

Notation of Gaussian Mixture Model

- Given a set of N-independent and identically distributed patterns $X^{(i)} = \{x_t; t=1, \ldots T\}$ associated with class w_i, The likelihood function $p(x_t \mid w_i)$ for class wi is a mixture of Gaussian distributions:

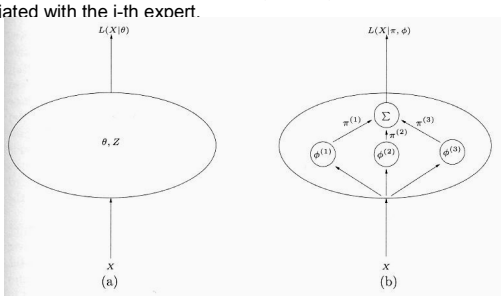
$$p(\mathbf{x}_t|\omega_i) = \sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(\mathbf{x}_t|\omega_i, \Theta_{r|i}),$$

where $\Theta_{r|i}$ represents the parameters of the r-th mixture component; R is the total number of mixture components; $p(\mathbf{x}_t|\omega_i, \Theta_{r|i}) \equiv \mathcal{N}(\mathbf{x}; \mu_{r|i}, \Sigma_{r|i})$ is the probability density function of the r-th component; and $P(\Theta_{r|i}|\omega_i)$ is the prior probability of the r-th component. Typically, $\mathcal{N}(\mathbf{x}; \mu_{r|i}, \Sigma_{r|i})$ is a Gaussian distribution with mean $\mu_{r|i}$ and covariance $\Sigma_{r|i}$.

In short, the output of a GMM is the weighted sum of R-component densities. The training of GMMs can be formulated as a maximum likelihood problem, where the mean vectors $\{\mu_{r|i}\}$, covariance matrices $\{\Sigma_{r|i}\}$, and mixture coefficients $\{P(\Theta_{r|i}|\omega_i)\}$ are often estimated by the iterative EM algorithm—the main topic of the current chapter.

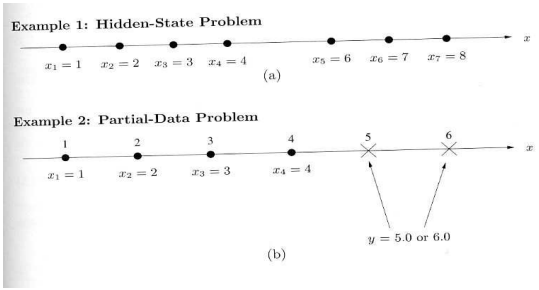
Expectation-Maximization Algorithm Example

- One important feature of the EM algorithm is that it can be applied to probablems in which observed data provide "partial" information only or when artifically introducing some information can greatly simplify the parameter estimation process.
- Examples of parameter estimation by EM.
  - (a) EM for general missing data problems, where θ is the constructural parameters to be estimated and Z is the set of missing data.
  - (b) EM for hidden-state problems in which the parameter θ can be divided into two groups: π and ϕ , where π represents the prio probability of the j-th expert and ϕ defines the density function associated with the i-th expert.



Expectation-Maximization Algorithm Example II

- Examples of parameter estimation by EM.



Traditional Derivation of EM

- Each EM iteration is composed of two steps – Estimation (E) and Maximization (M).
- The M-step maximizes a likelihood function that is further refined in each iteration by the E-step.

An Example of Simple EM (basic non-EM example)

- (A. Moore) Let events be ''grades in a class"
  - W1 = gets an A, P(A) = ½
  - W2 = gets a B, P(B) = μ,
  - W3 – gets a C, P(C) = 2μ,
  - W4 = gets a D, P(D) = ½ - 3μ
- Assume we want to estimate m from data. In a given class there were {a, b, c, d} number of students in each category, respectively.
- What's the MLE of μ given a, b, c, d?

➔log p(a, b, c, d | m) = a log ½ + b log μ + c log 2μ + d log (1/2 – 3μ).

➔We will get μ = (b+c) / 6(b+c+d).

An example of simple EM (II)

- In the above example, if we only know the total number of categories A+B = h,
- Then we can answer the question circularly:
  - Expectation: If we know μ, then the expected values of a and b can be represented by
    a = ½ h / (½+μ) and b = μ h / (½+μ).
  - Maximization: If we know the expected values a and b, we can calculate the maximum likelihood estimation of μ:
    μ = (b+c) / 6(b+c+d).
- Therefore, for an EM:
  1. We begin with a guess for m.
  2. We iterate between EXPECTATION and MAXIMIZATION to improve our estimates of μ, a and b.
  3. Continue iterating until converged. Coverging to local optimum is assured.
- Convergence proof based on fact that Prob (data | μ) must increase or remain the same between each iteration.

EM algorithm for General GMM

- General GMM: including mixture coefficient, mean vector and covariance matrix.

Assume a Gaussian mixture model:

$$\theta = \{\pi^{(j)}, \mu^{(j)}, \Sigma^{(j)}; j = 1, \ldots, J\},$$

where $\pi^{(j)}$, $\mu^{(j)}$, and $\Sigma^{(j)}$ denote, respectively, the mixture coefficient, mean vector, covariance matrix of the j-th component density. The GMM's output is given by

$$p(x_t|\theta) = \sum_{j=1}^{J} \pi^{(j)} p(x_t|\delta_t^{(j)} = 1, \phi^{(j)}),$$

where

$$p(x_t|\delta_t^{(j)} = 1, \phi^{(j)}) = (2\pi)^{-\frac{D}{2}} |\Sigma^{(j)}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_t - \mu^{(j)})^T (\Sigma^{(j)})^{-1}(x_t - \mu^{(j)})\right\} \quad (1)$$

is the j-th Gaussian density of the GMM.

EM for GMM algorithm

- Process

After the initialization of θ_0, the EM iteration is as follows:

- *E-step.* In the n-th iteration, compute $h_n^{(j)}(x_t)$ for each j and t using Eqs. (1) and (2). This is followed by the M-step described next.
- *M-step.* Maximize $Q(\theta|\theta_n)$ with respect to θ to find θ^*. Replace θ_n by θ^*. Then, increment n by 1 and repeat the E-step until convergence.

$$h_n^{(j)}(x_t) = P(\delta_t^{(j)} = 1|x_t, \theta_n) = \frac{p(x_t|\delta_t^{(j)} = 1, \phi_n^{(j)})\pi_n^{(j)}}{\sum_{k=1}^{J} p(x_t|\delta_t^{(k)} = 1, \phi_n^{(k)})\pi_n^{(k)}}. \quad (2)$$

EM for GMM Example

Numerical Example 1. This example uses the data in Figure 3.3(a) as the observed data. Assume that when EM begins, $n = 0$ and

$$\theta_0 = \left\{\pi_0^{(1)}, \{\mu_0^{(1)}, \sigma_0^{(1)}\}, \pi_0^{(2)}, \{\mu_0^{(2)}, \sigma_0^{(2)}\}\right\} = \{0.5, \{0, 1\}, 0.5, \{9, 1\}\}.$$

Therefore, one has

$$h_0^{(1)}(x_t) = \frac{\frac{\pi_0^{(1)}}{\sigma_0^{(1)}} e^{-\frac{1}{2}(x_t-\mu^{(1)})^2/(\sigma_0^{(1)})^2}}{\sum_{k=1}^{2} \frac{\pi_0^{(k)}}{\sigma_0^{(k)}} e^{-\frac{1}{2}(x_t-\mu^{(k)})^2/(\sigma_0^{(k)})^2}} = \frac{e^{-\frac{1}{2}x_t^2}}{e^{-\frac{1}{2}x_t^2} + e^{-\frac{1}{2}(x_t-9)^2}}$$

and

$$h_0^{(2)}(x_t) = \frac{e^{-\frac{1}{2}(x_t-9)^2}}{e^{-\frac{1}{2}x_t^2} + e^{-\frac{1}{2}(x_t-9)^2}}.$$

Table 3.1. Values of $h_0^{(j)}(x_t)$ in Example 1

Pattern Index (t)	Pattern (x_t)	$h_0^{(1)}(x_t)$	$h_0^{(2)}(x_t)$
1	1	1	0
2	2	1	0
3	3	1	0
4	4	1	0
5	6	0	1
6	7	0	1
7	8	0	1

Table 3.2. Values of $Q(\theta|\theta_n)$, $\mu^{(j)}$ and $(\sigma^{(j)})^2$ in the course of EM iterations. Data shown in Figure 3.3(a) were used as the observed data.

Iteration (n)	$Q(\theta\|\theta_n)$	$\mu_n^{(1)}$	$(\sigma_n^{(1)})^2$	$\mu_n^{(2)}$	$(\sigma_n^{(2)})^2$
0	$-\infty$	0	1	9	1
1	-43.71	2.50	1.25	6.99	0.70
2	-25.11	2.51	1.29	7.00	0.68
3	-25.11	2.51	1.30	7.00	0.67
4	-25.10	2.52	1.30	7.00	0.67
5	-25.10	2.52	1.30	7.00	0.67

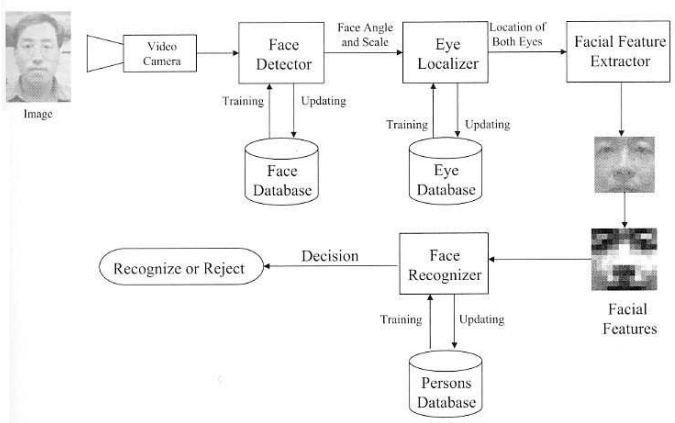
Substituting $X = \{1, 2, 3, 4, 6, 7, 8\}$ into Eqs. 3.2.28 and 3.2.29, Table 3.1 is obtained. Substituting $h_0^{(j)}(x_t)$ in Table 3.1 into Eqs. 3.2.17 through 3.2.19 results in

$$\theta_1 = \{0.57, \{2.50, 1.12\}, 0.43, \{6.99, 0.83\}\}.$$

Then, continue the algorithm by computing $Q(\theta|\theta_1)$

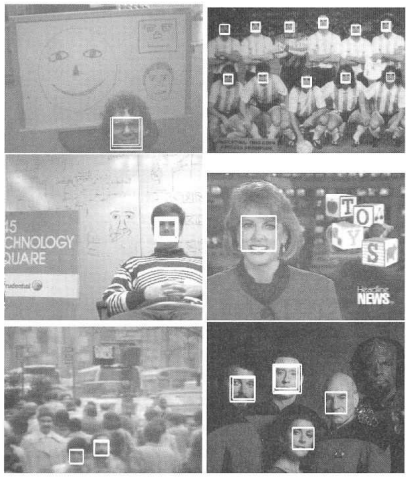
Face Recognition -- Introduction

- Face Recognition System



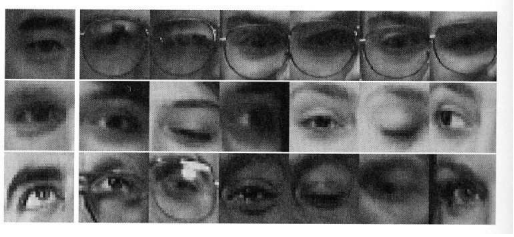
Face Detection

- Examples of Face Detectors:



Eye Detector

- Examples of Eye detectors:



Categories of Face Detection Algorithms

- Knowledge-based / Rule-based methods: use known human prior knowledge.
- Feature invariant approaches: aim to find structural features that exist even when the pose, viewpoint or lighting conditions vary, and then use these to locate faces.
- Template matching methods: Several standard face patterns are stored to describe the face as a whole or the facial features separately.
- Appearance-based methods / supervised learning methods: learn models or templates from a set of training images

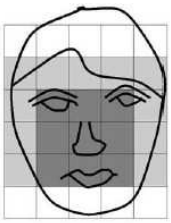
Knowledge-based Methods

- Human-specified rules:
  - A face appears in an image with two eyes that are symmetric to each other, a nose, and a mouth.
  - The relations between features can be represented by their relative distances and positions.
  - Facial features in an input image are extracted first, and face candidates are identified based on the coded rules.
  - A verification process is usually applied to reduce false detections.
- Difficulties of these methods:
  - The trade-off of details and extensibility
  - It is hard to enumerate all possible cases. In a restricted case – heuristics about faces work well in detecting frontal faces in uncluttered scenes.
- Example: (Yang and Huang 1994) Multiresolution Rule-based Methods
  - Three levels of rules:
    - All possible face candidates are found by scanning a window over the input image.
    - The rules at a higher level are general descriptions of what a face looks like
    - The rules at lower levels rely on details of facial features.

Multiresolution of face image

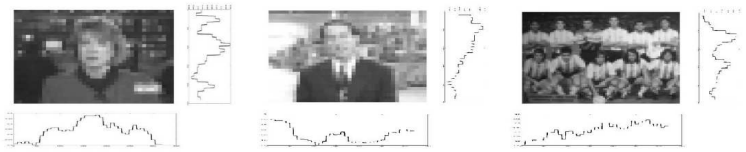
Knowledge-Based Method – Yang and Huang 1994

- Rules at the lowest resolution (Level 1):
  - The center part of the face has four cells with a basically uniform intensity
  - The upper round part of a face has a basically uniform intensity
  - The difference between the average gray values of the center part and the upper round part is significant
- The lowest resolution image is searched for face candidates and these are further processed at finer resolution.
- Rules at the Level 2:
  - Local histogram equalization is performed on the face candidates regions, followed by edge detection.
- Rules at the Level 3,
  - Detail rules of eyes and mouth
- Performance:
  - detect 50 faces out of 60 images.
  - 28 of the detected faces are false alarm
- Multiresolution approach inspired many follow-up researches.



Knowledge-based Method – Kotropoulos and Pitas 1997

- Use horizontal and vertical projections of the pixel intensity.
- The horizontal profile of an input image is obtained first, and then two local minima may correspond to the left and right side of the head.
- The vertical profile is obtained the the local minima are determined for the locations of mouth lips, nose tip, and eyes.
- This method has been tested ona video sequence database of 37 different people.
- Report a detection rate of 86.5%.
- Have difficulty to locate a face in a complex background.



References

- L. Rabiner and B. H. Juang, “Fundamentals of Speech Recognition”, Prentice-Hall 1993 (Chapter 3)
- S.Y. Kung, Mak, and S. H. Lin, “Biometric Authentication”, Prentice-Hall, 2005. (Chapters 2, 3 and 8)
- Andrew Moore, “Tutorial of Gaussian Mixture Models” http://www-2.cs.cmu.edu/~awm/tutorials/gmm14.pdf
- M.-H. Yang, D. Kriegman and N. Ahuja, “Detecting Faces in Images: A Survey”, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 24, No. 1, Jan 2002.