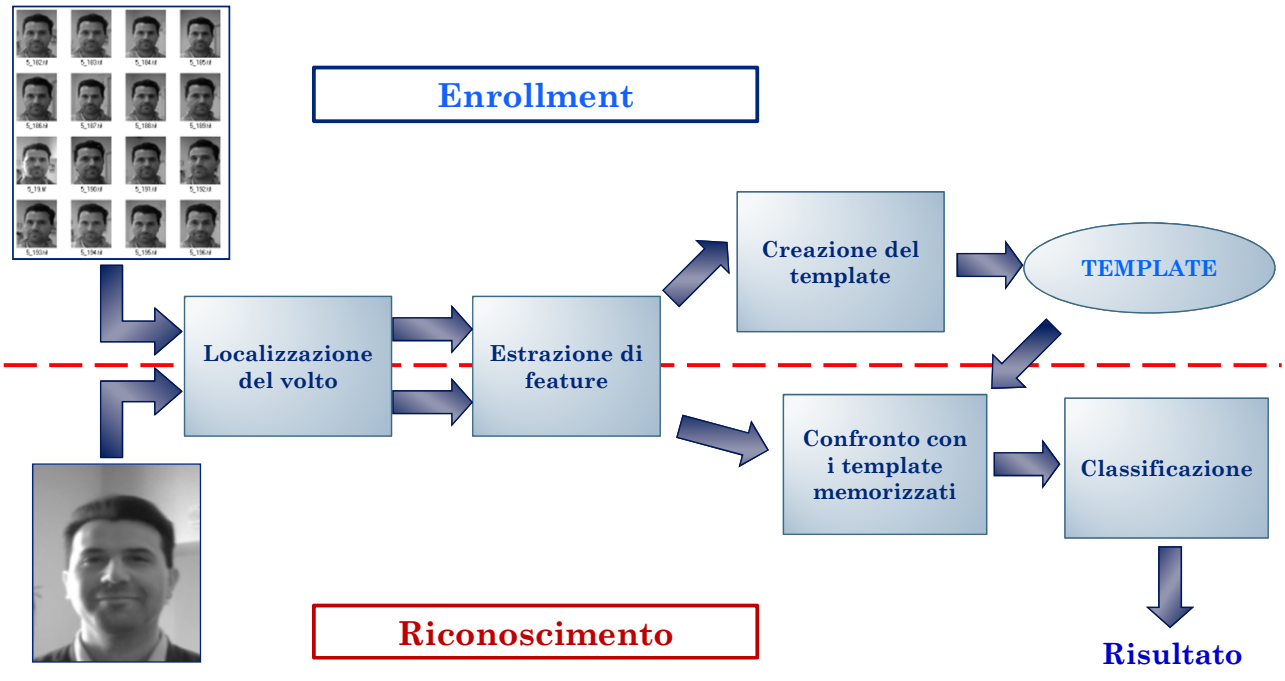
Riconoscimento del volto

Tecniche 2D

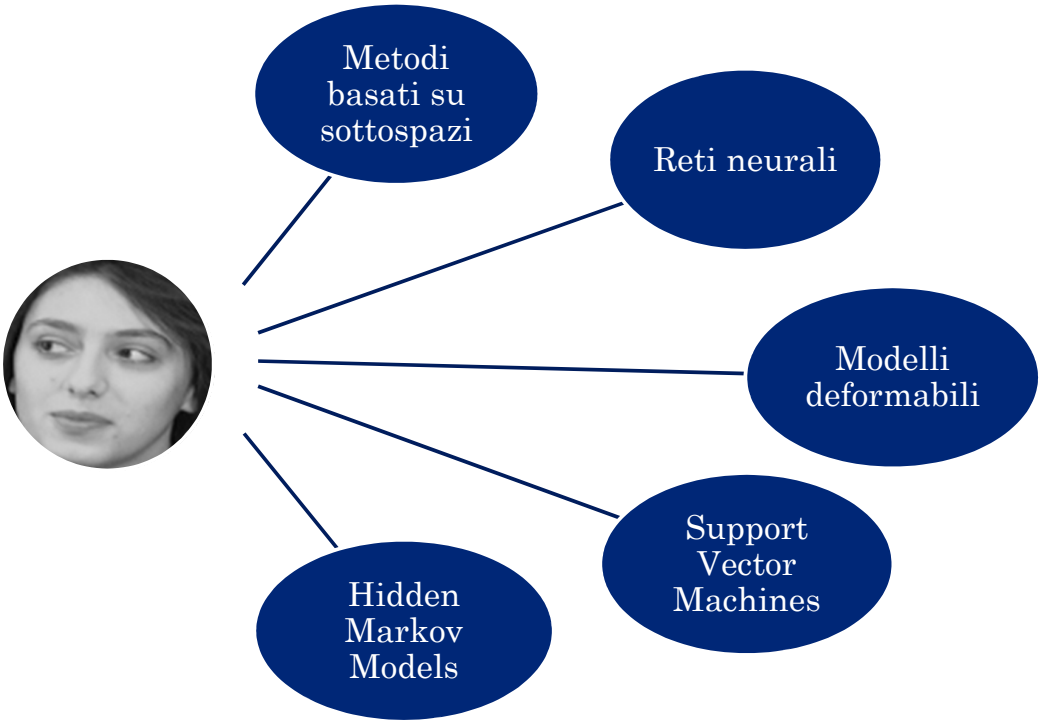
Annalisa Franco
annalisa.franco@unibo.it

Dario Maio
dario.maio@unibo.it

Il sistema di riconoscimento



Alcune tecniche di riconoscimento



Possibili strategie

- Le difficoltà del riconoscimento possono essere affrontate facendo ricorso fondamentalmente a **due strategie di base**:

1. Costruzione di uno **spazio di feature ben rappresentativo**, in cui le distribuzioni dei volti siano più semplici da trattare (ovvero non lineari e non convessi in misura minore rispetto ad altri possibili spazi). Ciò implica il ricorso a tecniche di normalizzazione geometrica e fotometrica nonché l'estrazione di feature stabili rispetto alle possibili variazioni.
2. Costruzione di **classificatori robusti** in grado di risolvere la non linearità e la non convessità e di generalizzare al meglio. Infatti una buona normalizzazione e un'efficace estrazione di feature alleviano i problemi ma non li risolvono completamente.
3. Dunque un algoritmo efficace ed efficiente usa le due strategie in combinazione.

Note sui metodi di riconoscimento (1)

- **Geometric feature-based approach**: le proprietà e le relazioni (aree, distanze, angoli, ..) delle caratteristiche estratte (occhi, naso, bocca, mento,…) sono usate come descrittori per il riconoscimento.

  Questi metodi – ormai datati - sono semplici ed efficienti se capaci di ottenere una tangibile riduzione delle informazioni da trattare e se risultano non sensibili a variazioni di illuminazione e punto di vista. Tuttavia le sole caratteristiche geometriche non tengono in conto della tessitura del volto né del suo aspetto, pertanto tali metodi non possono in generale garantire buoni risultati.

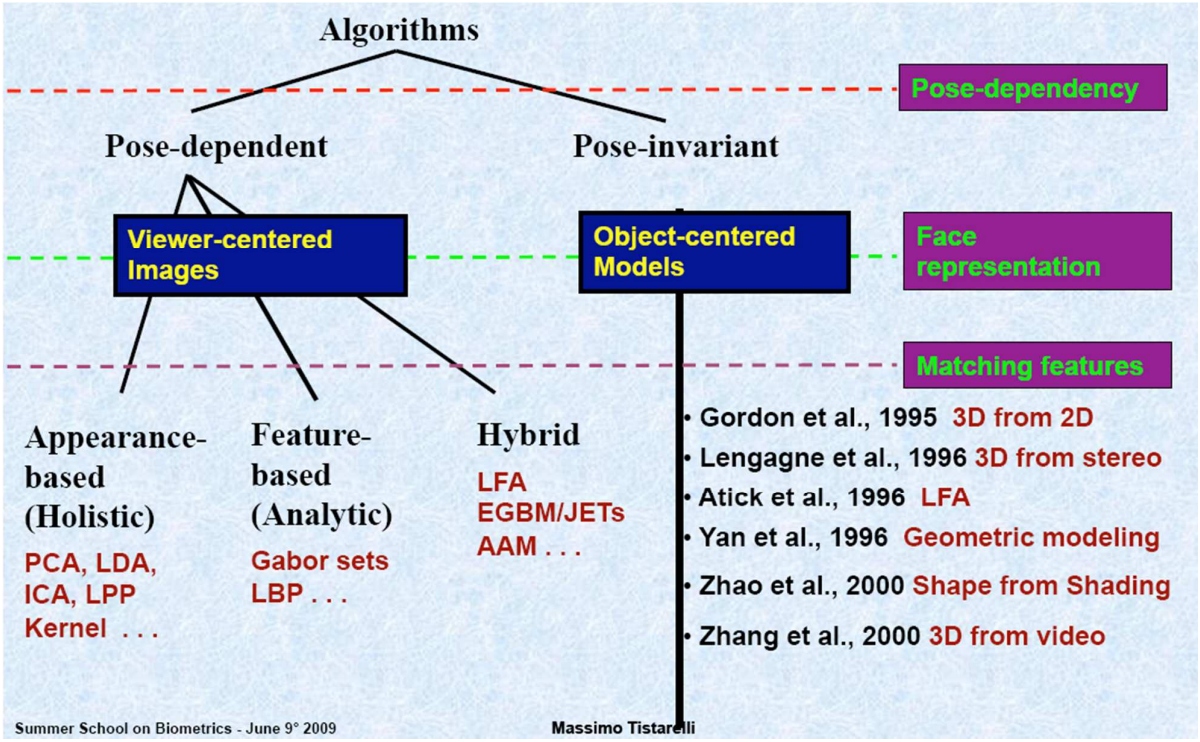
- **Statistical learning approach**: apprende dagli esempi (immagini dell'aspetto o feature estratte dall'aspetto) al fine di costruire buone feature e costruire classificatori ottimali. Durante la fase di learning si deve far buon uso della conoscenza a priori sui campioni.

  Molti algoritmi moderni fanno uso di questo approccio.

Note sui metodi di riconoscimento (2)

- **Appearance-based approach (es. PCA, LDA)**: si opera direttamente su una rappresentazione dell'immagine (es. array delle intensità dei pixel) ed si estraggono feature in un sottospazio derivato dalle immagini di training. Sebbene questi metodi olistici lineari evitino alcuni dei problemi d'instabilità dei metodi basati su feature geometriche, il fatto proprio di non essere in grado di gestire la non linearità li rende poco accurati in generale.
- **Non-linear kernel techniques**: i suddetti metodi possono essere estesi (**es. Kernel PCA, Kernel LDA**) attraverso una proiezione non lineare (riduzione della dimensionalità) in uno spazio di feature, preservando i particolari d'interesse. Buone sono le prestazioni sui volti di training, non altrettanto si osserva con volti mai appresi proprio a causa della maggiore flessibilità rispetto ai metodi lineari.
- **Local appearance-based approach**: si fa uso di appropriati filtri al fine di ottenere una maggiore robustezza rispetto alle variazioni, es. Local Feature Analysis (**LFA**), Elastic Graph Bunch Matching (**EGBM**) che fa uso di Gabor wavelet-based feature, Local Binary Pattern (**LBP**).

Tassonomia dei metodi di riconoscimento



Rappresentazione vettoriale di un volto

- Un volto può essere rappresentato come un'immagine bidimensionale **I**, ovvero una matrice di dimensioni $w \times h$ in cui ciascun elemento rappresenta il valore di intensità di un pixel, ad esempio, in una scala di grigi, un valore compreso tra 0 e 255. Alternativamente un'immagine può essere rappresentata come un vettore di $n = w \times h$ pixel, ad esempio concatenando le righe dell'immagine una dietro l'altra.
- Un'immagine, rappresentata con un vettore, è dunque un punto in uno spazio di dimensione *n-dimensionale*. Le coordinate del punto corrispondono a ***caratteristiche estratte dall'immagine*** (valore dei livelli di grigio dei pixel o, più in generale, altre caratteristiche più "complesse").

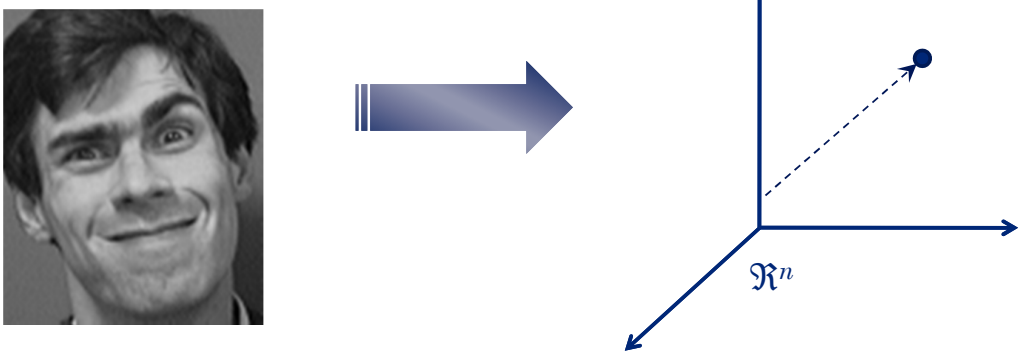
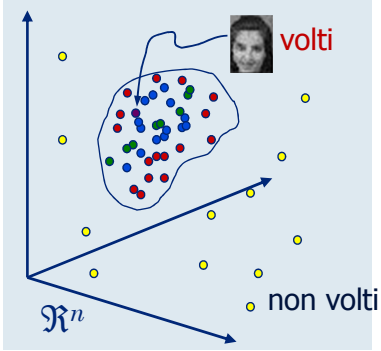


Image Space vs Feature Space

- **Image space**:
  - Un'immagine di dimensioni $w \times h$ è rappresentata da un vettore n-dimensionale, con $n = w \times h$
  - Notevoli problemi di dimensionalità
  - Tipicamente si adottano tecniche di riduzione della dimensionalità
- **Feature space**:
  - **Filtri di Gabor**
  - **Discrete Cosine Transform**
  - **Operatore Local Binary Pattern**
  - **Codifica frattale**
  - Rappresentazione più "compatta"
  - Maggiore potere discriminante

Dati a elevata dimensionalità

- La rappresentazione di un'immagine tramite il valore dei suoi livelli di grigio dà luogo a un vettore di feature di dimensionalità molto elevata ($w \times h$).
- La gestione di questo tipo di dati è molto problematica a causa di un fenomeno noto come "***curse of dimensionality***":
  - i dati sono spesso affetti da rumore;
  - alcune dimensioni (pixel) non portano informazioni significative ai fini del riconoscimento;
  - i dati sono sparsi: è necessaria una grande quantità di immagini di training per avere una conoscenza adeguata dello spazio;
  - il confronto è un'operazione molto costosa in termini di complessità computazionale.
- Le informazioni più rilevanti circa le immagini dei volti si concentrano in **sottospazi dello spazio n-dimensionale.**
- Le tecniche di **riduzione della dimensionalità** hanno lo scopo di individuare e rappresentare questi sottospazi.



Trasformata Karhunen-Loève (1)

- **La trasformata KL** (la formulazione è identica a quella della **Principal Component Analysis (PCA)** nell'ipotesi di dati a media nulla), opera una riduzione della dimensionalità dei dati attraverso una proiezione in un sottospazio di dimensione k dello spazio n-dimensionale delle feature.
- Il sottospazio ottimale è calcolato a partire da un insieme di pattern di training non classificati.
- Sia $P = \{\mathbf{x}_i \in \Re^n \mid i=1,\ldots m\}$ un insieme di m pattern n-dimensionali, e siano:

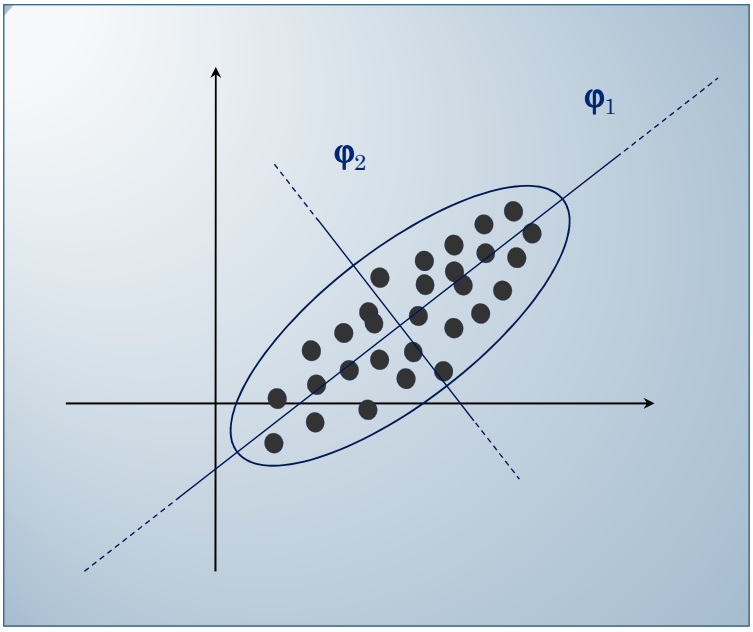
$$\bar{\mathbf{x}} = \frac{1}{m}\sum_{\mathbf{x}\in P}\mathbf{x}$$ **il vettore medio**

$$\mathbf{C} = \frac{1}{m}\sum_{\mathbf{x}\in P}(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T$$ **la matrice di covarianza**

allora per un dato k ($k<m$, $k<n$, $k>0$), lo spazio KL k-dimensionale ($S_{\bar{\mathbf{x}},\mathbf{\Phi}_k}$) è definito dal vettore medio e dalla matrice di proiezione $\mathbf{\Phi}_k \in \Re^{n\times k}$ le cui colonne sono costituite dagli autovettori di $\mathbf{C}$ corrispondenti ai k autovalori più grandi :

$\mathbf{\Phi}_k = [\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{ik}]$ con $\lambda_{i1} \geq \lambda_{i2} \geq \ldots \geq \lambda_{ik}$ φ_{i*} è l'autovettore corrispondente all'autovalore λ_{i*}

Trasformata Karhunen-Loève: esempio

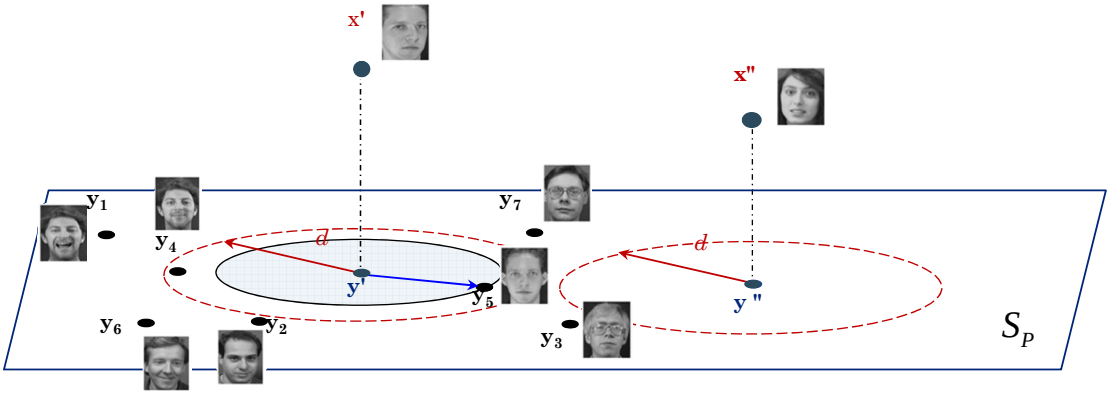


φ_1 e φ_2 sono gli autovettori della matrice di covarianza

φ_1 rappresenta la direzione di maggior varianza dei dati (autovalore più grande)

Un'applicazione: il metodo Eigenfaces

- **Idea di base**: effettuare le ricerche all'interno di un sottospazio KL (k-dimensionale) rappresentativo dei volti di training.
- Ogni immagine da riconoscere è proiettata nel sottospazio, e la classificazione avviene sulla base di un semplice criterio ***nearest neighbor***: il volto da riconoscere è associato alla faccia del training set a cui corrisponde la distanza minima nel sottospazio.
- È possibile fissare **una soglia *d* di riconoscimento**: il volto è accettato solo se la distanza minima è inferiore alla soglia prefissata.



M.A. Turk, A.P. Pentland, "Face recognition using eigenfaces", *In Proc. Computer Vision and Pattern Recognition*, pp.586-591, 1991.

Il metodo Eigenfaces: algoritmo (1)

- **Training**
  - Collezionare un training set $P=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m\}$ di punti n-dimensionali rappresentativi dei volti
  - Calcolare il vettore medio $\bar{\mathbf{x}} = \frac{1}{m}\sum_{\mathbf{x}\in P}\mathbf{x}$
  - Calcolare la matrice di covarianza $\mathbf{C} = \frac{1}{m}\sum_{\mathbf{x}\in P}(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T$
  - Calcolare autovettori e autovalori della matrice ***C***
  - Ordinare gli autovalori in ordine decrescente
  - Creare la matrice di proiezione

$$\mathbf{\Phi}_k = [\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{ik}] \quad \text{con} \quad \lambda_{i1} \geq \lambda_{i1} \geq \ldots \geq \lambda_{ik}$$

  mantenendo i k autovettori corrispondenti agli autovalori più grandi
  - Proiettare tutti i punti $\mathbf{x}_i \in P$ nel sottospazio ottenendo l'insieme di punti k-dimensionali $P'=\{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_m\}$, usati per il riconoscimento.

$$\mathbf{y}_i = \mathbf{\Phi}_k^T (\mathbf{x}_i - \bar{\mathbf{x}})$$

Una qualunque faccia $\mathbf{x}_i$ può essere rappresentata dalla proiezione di $(\mathbf{x}_i - \bar{\mathbf{x}})$ nell'autospazio determinato dagli eigenface

Il metodo Eigenfaces: algoritmo (2)

- **Riconoscimento**
  - *Proiettare l'immagine* da riconoscere **x** *nel sottospazio* KL ottenendo una sua rappresentazione ridotta **y**.
  - Calcolare *la distanza di* **y** *da tutti i punti* $\mathbf{y}_i \in P'$.
  - Se la distanza minima è inferiore alla soglia prefissata d, il volto è riconosciuto come appartenente all'individuo associato al punto $\mathbf{y}_i$ più vicino.

Note sul calcolo degli autovettori di C (1)

- La matrice di covarianza **C** ha dimensioni $n \times n$ e determinare i suoi autovettori è un problema molto pesante dal punto di vista computazionale, per la maggior parte dei casi.
- Posto $\mathbf{\Psi}_i = \mathbf{x}_i - \bar{\mathbf{x}}$, la matrice di covarianza si può esprimere come:

$$\mathbf{C} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{\Psi}_i \mathbf{\Psi}_i^T = \frac{1}{m}\mathbf{A}\mathbf{A}^T \quad \text{essendo } \mathbf{A} = [\mathbf{\Psi}_1, \mathbf{\Psi}_2, ..., \mathbf{\Psi}_m] \text{ di dimensioni } n \times m$$

- **Vi sono al più *m*-1 autovettori**.
- Sia r il rango della matrice **C**; r è minore o uguale al minimo tra n e m-1. Fortunatamente è possibile calcolare gli r autovettori di **C**, precalcolando gli autovettori e gli autovalori della matrice $\mathbf{A}^T\mathbf{A}$, che ha dimensioni $m \times m$, e quindi trattabile facilmente, essendo in generale $m << n$.
- Segue la dimostrazione di questo asserto.

Una qualunque faccia
può essere rappresentata dalla proiezione di
$\Phi = \Gamma - \Psi$

Note sul calcolo degli autovettori di C (2)

Siano $\mathbf{L}=\mathbf{A}^T \cdot \mathbf{A}$; $\mathbf{v}_k$, $k=1,..,r$ gli autovettori di $\mathbf{L}$; λ_k, $k=1,..,r$ gli autovalori non nulli di $\mathbf{L}$. Vale allora:

$$\mathbf{A}^T \cdot \mathbf{A} \cdot \mathbf{v}_k = \lambda_k \cdot \mathbf{v}_k .$$

Premoltiplicando ambo i membri della precedente equazione per $\mathbf{A}$, si ottiene:

$\mathbf{A} \cdot \mathbf{A}^T \cdot \mathbf{A} \cdot \mathbf{v}_k = \mathbf{A} \cdot \lambda_k \cdot \mathbf{v}_k$, che può essere riscritta come:

$\left(\mathbf{A} \cdot \mathbf{A}^T\right) \cdot \mathbf{A} \cdot \mathbf{v}_k = \lambda_k \cdot \mathbf{A} \cdot \mathbf{v}_k$, da cui si deduce per definizione di autovettore che:

$\mathbf{u}_k = \mathbf{A} \cdot \mathbf{v}_k$ è autovettore di $\mathbf{A} \cdot \mathbf{A}^T$ e quindi anche della matrice $\mathbf{C}$.

Gli autovalori di $\mathbf{A} \cdot \mathbf{A}^T$ sono gli stessi di $\mathbf{A}^T \cdot \mathbf{A}$ e sono anche quelli di $\mathbf{C}$.

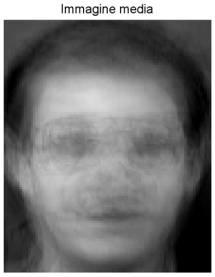
Il metodo Eigenfaces: un esempio (1)

database ORL (40 individui, 10 pose per ogni individuo)

Il metodo Eigenfaces: un esempio (2)

Training set

Immagine media



Nel training set per ognuna delle 10 persone sono presenti 2 pose; ogni immagine è 92x112 pixel di 8 bit.

Eigenfaces

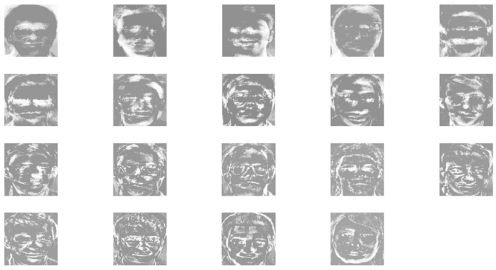
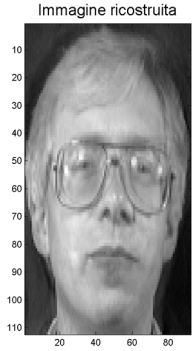


Immagine da riconoscere

Immagine ricostruita



In questo esempio non si escludono autovettori, $k=m-1=2*10-1$

Il metodo Eigenfaces: un altro esempio (1)

Faccia media

Autovettori principali

Autovettori minori

Il metodo Eigenfaces: un altro esempio (2)

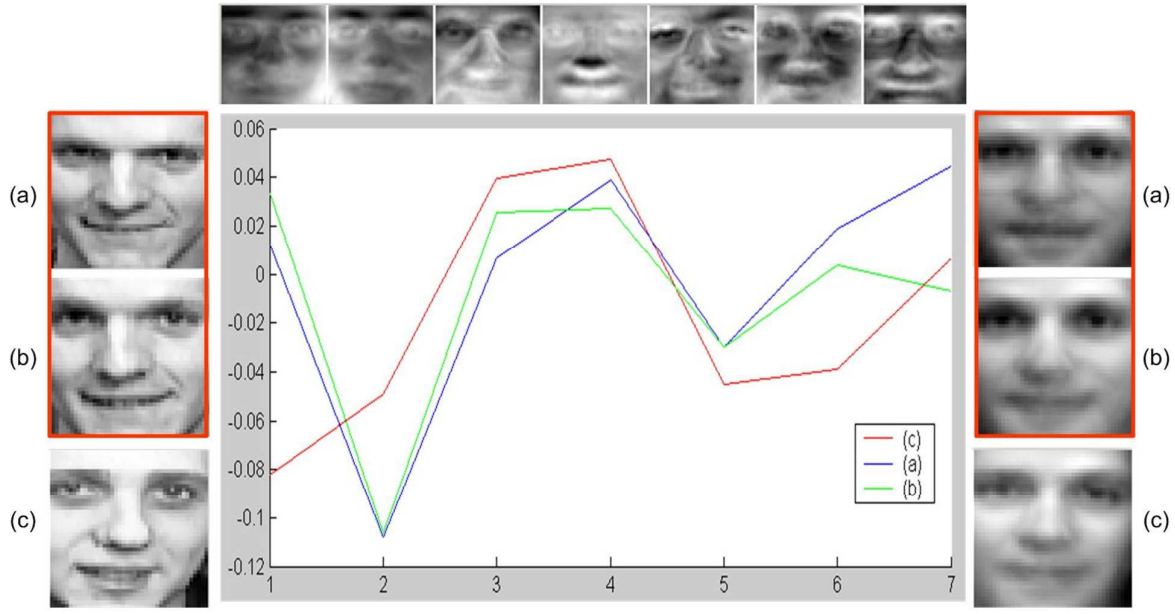
- Data la matrice di proiezione

$$\mathbf{\Phi}_k = [\varphi_{i1}, \varphi_{i2}, ..., \varphi_{ik}] \quad \text{con} \quad \lambda_{i1} \geq \lambda_{i2} \geq ... \geq \lambda_{ik}$$

ogni immagine è ricostruibile nello spazio originario come ***combinazione lineare*** dei *k* autovettori: $\mathbf{x} \approx \bar{\mathbf{x}} + a_1\varphi_{i1} + a_2\varphi_{i2} + ... + a_k\varphi_{ik} = \breve{\mathbf{x}}$

N.B.
Nella figura i pesi $a_1, a_2..., a_k$ sono di fatto le componenti del vettore proiezione **y**

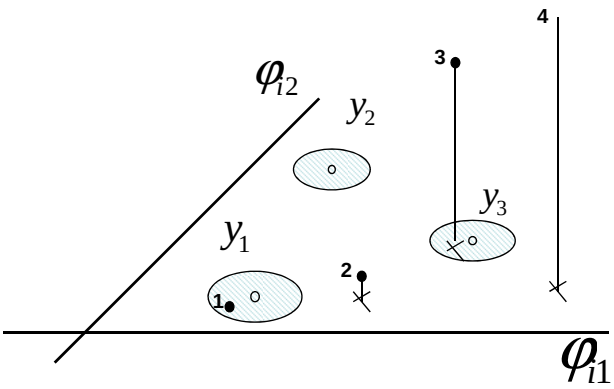
Il metodo Eigenfaces: un altro esempio (3)



Projection coefficients of images onto the eigenvectors. (a) and (b) are images from the same subject. Subject (c) is different from (a) and (b). Left column: original test images. Right column: reconstructed images using the first 7 eigenvectors.

Bontà del riconoscimento

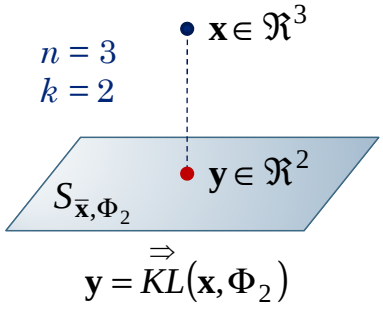
Per migliorare la bontà del riconoscimento è opportuno misurare anche quanto l'immagine da riconoscere sia vicina allo spazio delle facce del training set. Infatti l'immagine di un generico volto dovrebbe proiettarsi in estrema prossimità dello spazio delle facce che in generale, per come è stato costruito (volti del training set), dovrebbe descrivere tutte le immagini con l'aspetto di un volto. Si può allora calcolare anche la distanza $\|\mathbf{x}-\breve{\mathbf{x}}\|_2$ e imporre che sia inferiore a una certa soglia.



1. Il vettore è vicino allo spazio delle facce e la sua proiezione a un volto noto.
2. Il vettore è vicino allo spazio delle facce, ma la sua proiezione non è vicina ad alcun volto noto.
3. Il vettore è lontano dallo spazio delle facce, ma la sua proiezione è vicina a un volto noto.
4. Il vettore è lontano dallo spazio delle facce e la sua proiezione non è vicina ad alcun volto noto.

Trasformata Karhunen-Loève: operatori (1)

- **Proiezione**:
  - proiezione geometrica di un punto sull'iper-piano che definisce il sottospazio KL.
  - In realtà la vera proiezione geometrica è un vettore che ha la stessa dimensionalità n del vettore proiettato mentre qui indichiamo con proiezione il vettore (ridotto) nello spazio KL (*ciò corrisponde ad eseguire un cambio di coordinate dopo la proiezione, a seguito del quale n-k coordinate divengono nulle).*
  - L'operazione è eseguita moltiplicando la matrice di proiezione trasposta per il vettore punto al quale è preventivamente sottratta la media.



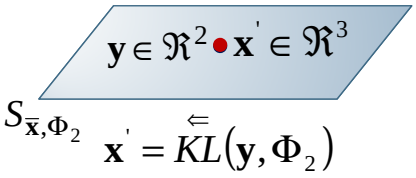
$$\vec{KL}: \Re^n \to \Re^k \quad \vec{KL}\left(\mathbf{x}, S_{\bar{\mathbf{x}},\Phi_k}\right) = \Phi_k^T(\mathbf{x}-\bar{\mathbf{x}})$$

Trasformata Karhunen-Loève: operatori (2)

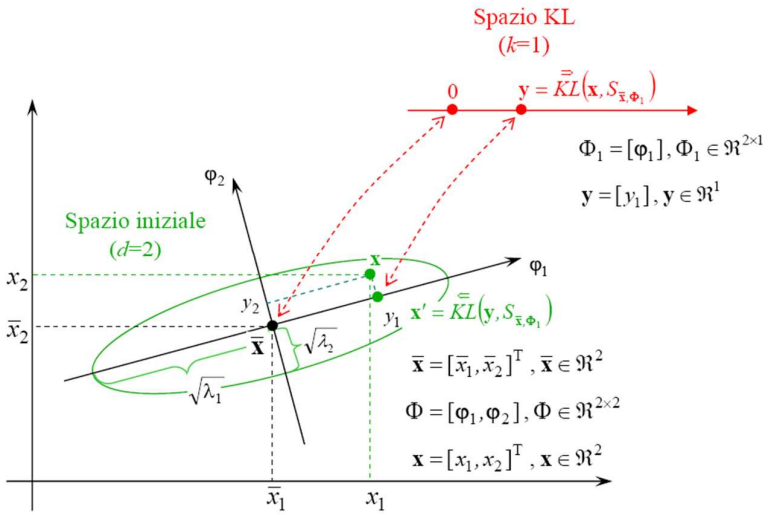
- **Retroproiezione**:
  - La retroproiezione nello spazio originale si ottiene moltiplicando il vettore per la matrice di proiezione e sommando il vettore medio.
  - Questa trasformazione non sposta spazialmente il vettore, che giace ancora sullo spazio KL, ma opera un *cambiamento di coordinate* che ne permette la codifica in termini delle n componenti dello spazio originale.

$$\overleftarrow{KL}: \Re^k \rightarrow \Re^n \qquad \overleftarrow{KL}\left(\mathbf{y}, S_{\bar{\mathbf{x}},\Phi_k}\right) = \Phi_k \mathbf{y} + \bar{\mathbf{x}}$$

$n = 3$
$k = 2$



Trasformata Karhunen-Loève: esempio



- L'ellisse rappresenta la distribuzione dei pattern nel training set P.
- ϕ_1 e ϕ_2 sono gli autovettori della matrice di covarianza.
- Gli autovalori λ_1 e λ_2 rappresentano le varianze della distribuzione lungo gli assi
- **y_1 e y_2 sono le proiezioni di x sugli assi ϕ_1 e ϕ_2.**

- Se λ_2 è piccolo, **x può essere approssimato con x' (retroproiezione di y)** senza perdite significative di informazione.
- **Si può dimostrare che tra tutte le riduzioni di dimensionalità lineari KL è quella che preserva al massimo l'informazione dei vettori originali.**

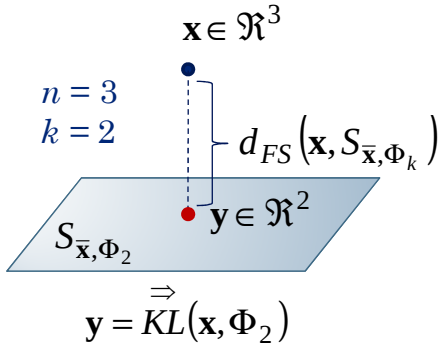
Trasformata Karhunen-Loève: distanze

- Distanza dallo spazio:
  - distanza euclidea tra il punto **x** e la retroproiezione della sua proiezione nello spazio $S_{\bar{x},\Phi_k}$:

$$d_{FS}\left(\mathbf{x}, S_{\bar{\mathbf{x}},\Phi_k}\right) = \left\| \mathbf{x} - \overleftarrow{KL}\left(\overrightarrow{KL}\left(\mathbf{x}, S_{\bar{\mathbf{x}},\Phi_k}\right), S_{\bar{\mathbf{x}},\Phi_k} \right) \right\|_2$$

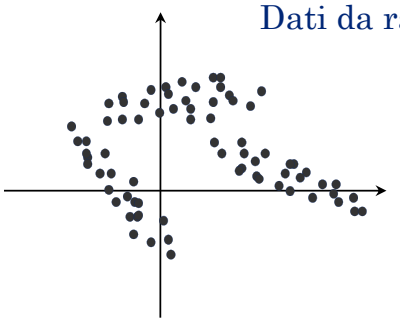
che si può semplificare come:

$$d_{FS}\left(\mathbf{x}, S_{\bar{\mathbf{x}},\Phi_k}\right) = \sqrt{\left\|\mathbf{x}-\bar{\mathbf{x}}\right\|_2^2 - \left\|\mathbf{y}\right\|_2^2}$$

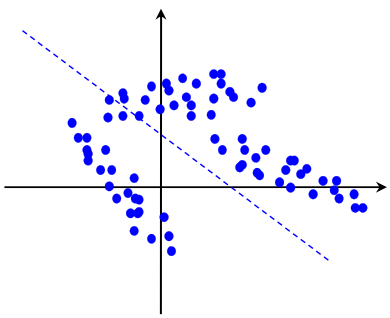


Trasformata MKL (Multi-space KL)

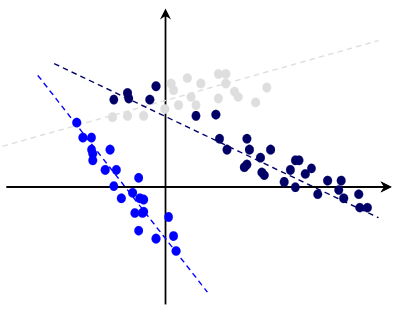
Dati da rappresentare



Trasformata KL

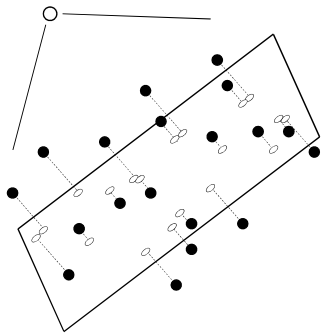


Trasformata MKL

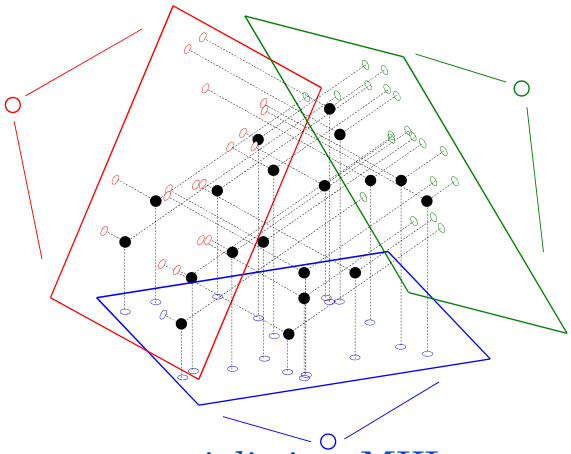


KL e MKL a confronto

Ogni sottospazio può essere una specifica vista sui dati relativa a un certo punto di osservazione. MKL è di fatto una generalizzazione di KL.



punto di vista KL



punti di vista MKL

Trasformata MKL: definizione

Sia $P = \{\mathbf{x}_i \in \Re^n \mid i=1,\ldots,m\}$ un insieme di m vettori n-dimensionali; per ogni partizione $\wp=\{P_1, P_2,\ldots P_s\}$ di P e per ogni insieme $K=\{k_1, k_2,\ldots k_s\}$ di scalari, tali che:

$$\bigcup_{i=1,..,s} P_i = P, \quad P_i \cap P_j = \varnothing \quad \forall i,j=1,...,s \quad i \neq j$$ ← Vincolo di partizione

$$m_i = card(P_i) \geq \left\lfloor \frac{m}{s+1} \right\rfloor \quad \forall i=1,..,s$$ ← Vincolo di bilanciamento

$$k_i < m_i, \quad k_i > 0, \quad k_i < n \quad \forall i=1,..,s$$ ← Requisiti numerici

la trasformata MKL è definita da $S = \left\{ S_i \middle| S_i = S_{\bar{\mathbf{x}}_i, \Phi_{i,k_i}}, i=1,..,s \right\}$

$\bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in P_i} \mathbf{x}$ è il vettore medio di P_i

$\mathbf{\Phi}_{i,k_i}$ è la matrice di proiezione le cui colonne sono i k_i *autovettori* corrispondenti ai k_i *più grandi autovalori della matrice di* covarianza $\mathbf{C}_i$

$$\mathbf{C}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in P_i} (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)^T$$

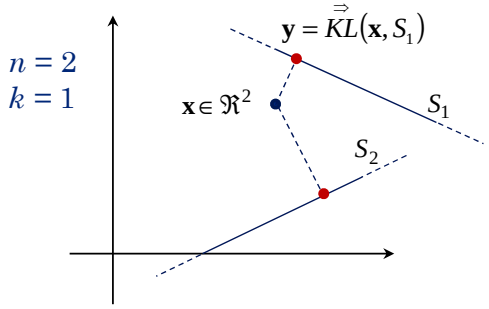
Trasformata MKL: operatori

- **Proiezione**:
  - operazione che mappa il pattern nello ***spazio a esso più vicino*** (in base alla metrica *distanza dallo spazio*).

$$\overrightarrow{MKL}:\Re^n \rightarrow \Re^{k_t} \qquad \overrightarrow{MKL} = \langle t, \mathbf{y} \rangle$$

$$\mathbf{y} = \overrightarrow{KL}(\mathbf{x}, S_t) \quad t = \arg\min_{i=1,..,s} \{d_{FS}(\mathbf{x}, S_i)\}$$

$$d_{FS}\left(\mathbf{x}, S_{\bar{\mathbf{x}},\Phi_k}\right) = \sqrt{\|\mathbf{x}-\bar{\mathbf{x}}\|_2^2 - \|\mathbf{y}\|_2^2}$$



- **Retroproiezione**:
  - operazione che mappa il pattern all'indietro nello spazio originale a partire da uno dei sotto-spazi MKL.

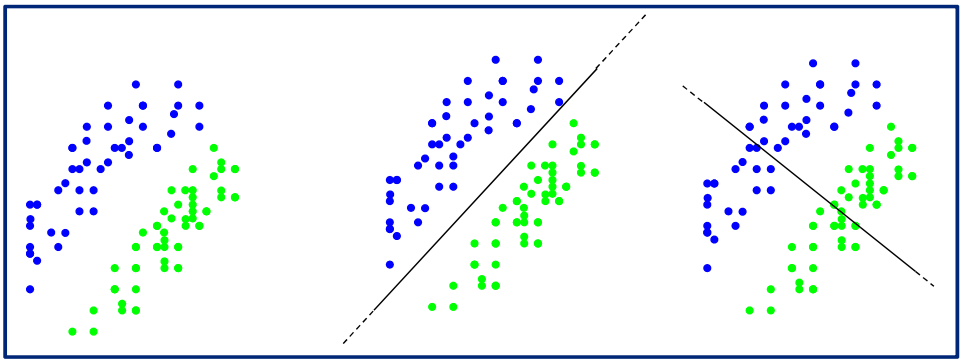
$$\overleftarrow{MKL}:\Re^{k_t} \rightarrow \Re^n \qquad \overleftarrow{MKL}(\langle t, \mathbf{y} \rangle, S) = \overleftarrow{KL}(\mathbf{y}, S_t)$$

Linear Discriminant Analysis (1)

- Il principale svantaggio del metodo Eigenfaces risiede nel fatto che la separazione (**scattering**) che è massimizzata non dipende solo dalla separazione inter-classe (utile per la classificazione) ma anche da quella intra-classe che ai fini della classificazione è un'informazione non desiderata.
- Quando sono presenti sensibili differenze nell'illuminazione e nell'espressione delle varie pose di un volto, molta della variazione nei dati è dovuta proprio questi cambiamenti. Le tecniche PCA di fatto conservano la maggior parte di queste variazioni e pertanto la similarità nello spazio delle facce non è necessariamente determinata dall'identità degli individui; in altre parole le proiezioni non sempre risultano ben raggruppate e, cosa peggiore, le classi possono risultare non giustamente separate.
- Belhumer et al. (1997) hanno proposto, per risolvere i suddetti problemi, un'applicazione di LDA (Linear Discriminant Analysis), denominata **Fisherfaces**.

Linear Discriminant Analysis (2)

- Tecnica di riduzione di dimensionalità lineare e supervisionata il cui obiettivo è massimizzare la separazione tra le classi (*che nel training set sono etichettate).*
- La trasformazione si determina sulla base di un criterio di ottimizzazione che ha lo scopo di massimizzare la separazione tra le classi a partire dalle matrici di scattering:
  - *within-class* S_w**:**
    - indica come i ***vettori sono sparpagliati*** rispetto al centro delle classi (ciascuno rispetto alla propria classe).
  - *between-class* S_b**:**
    - indica come i ***centri delle classi sono sparpagliati*** rispetto al centro generale della distribuzione (ovvero quanto le classi sono sparpagliate).



Linear Discriminant Analysis (3)

- Sia $P = \{\mathbf{x}_i \in \Re^n \mid i=1,\ldots,m\}$ un insieme di m pattern n-dimensionali e $\wp=\{P_1, P_2,\ldots P_s\}$ la partizione in classi di P indotta dall'etichettatura dei pattern in P. Sia inoltre m_i la cardinalità di ciascuna classe P_i. Allora le matrici di scattering sono definite come:
  - **Within-class:**

$$\mathbf{S}_{\mathbf{w}} = \sum_{i=1}^{s} m_i \mathbf{C}_i, \quad \mathbf{C}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in P_i} (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)^T, \quad \bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in P_i} \mathbf{x}$$

  - **Between-class:**

$$\mathbf{S}_b = \sum_{i=1}^{s} m_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_0)(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_0)^T, \quad \bar{\mathbf{x}}_0 = \frac{1}{m} \sum_{i=1}^{s} m_i \bar{\mathbf{x}}_i, \quad \bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in P_i} \mathbf{x}$$

Linear Discriminant Analysis (4)

- Tra i diversi criteri di ottimizzazione possibili quello più frequentemente utilizzato è la massimizzazione della quantità:

$$J_1 = tr\left(S_w^{-1} S_b\right) = \sum_{i=1}^{n} \lambda_i$$

- dove *tr è la* ***traccia*** *(somma degli autovalori) della matrice. Il criterio è* intuitivo in quanto cerca di *massimizzare lo scattering tra le classi* ($\mathbf{S}_b$) *minimizzando* al contempo (matrice inversa $\mathbf{S}_w^{-1}$) *quello all'interno di ogni classe*.
- Si dimostra che per la massimizzazione di J_1 *lo spazio ridotto ideale può* essere ottenuto (come nel caso di KL) tramite il calcolo degli autovettori relativi ai primi *k autovalori di una matrice; tale matrice in questo caso è* proprio $\mathbf{S}_w^{-1}\mathbf{S}_b$ **(e non C).**

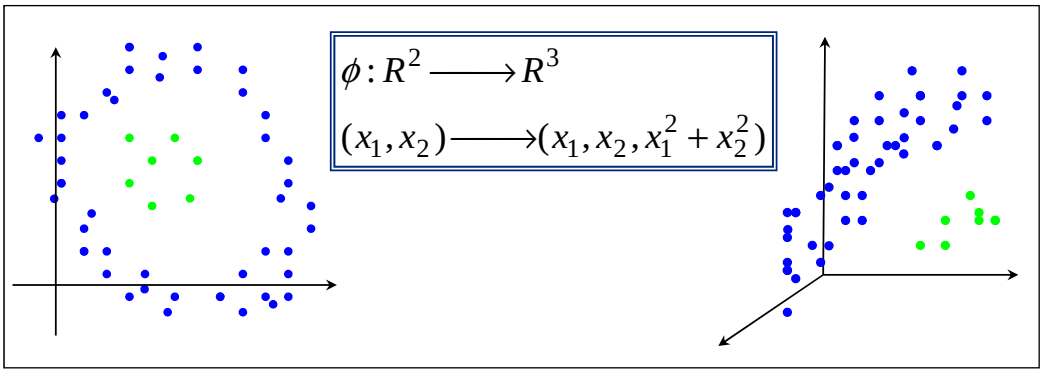
Un'applicazione: il metodo Fisherfaces

- Il metodo Fisherfaces è analogo al metodo Eigenfaces ma utilizza Linear Discriminant Analysis come trasformata di base.

Fisher faces

Kernel PCA

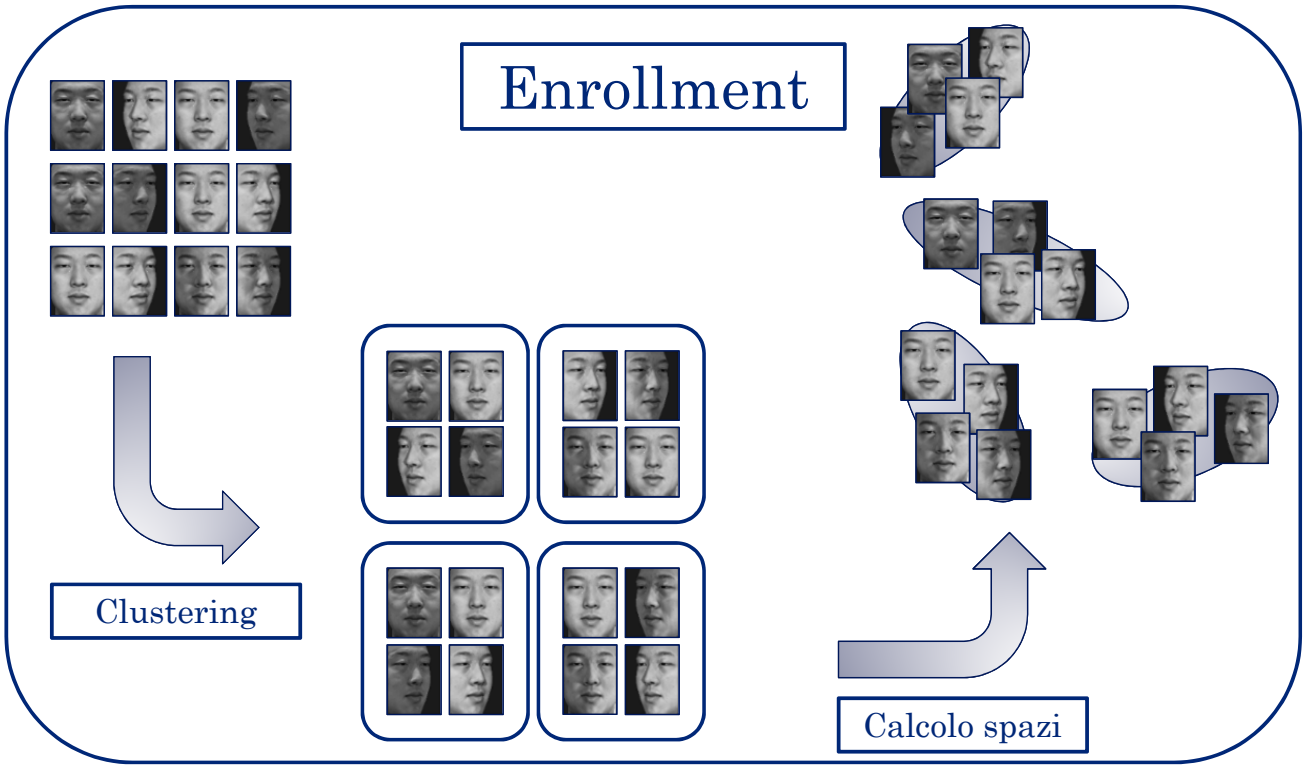
- La Kernel PCA è una tecnica che permette una analisi delle componenti principali (PCA) non lineare.
- La tecnica opera una trasformazione dei dati sulla base di una funzione kernel che realizza un *mapping in uno spazio a dimensionalità più elevata* rispetto allo spazio originale, dove la discriminazione tra pattern di classi diverse può risultare più semplice.



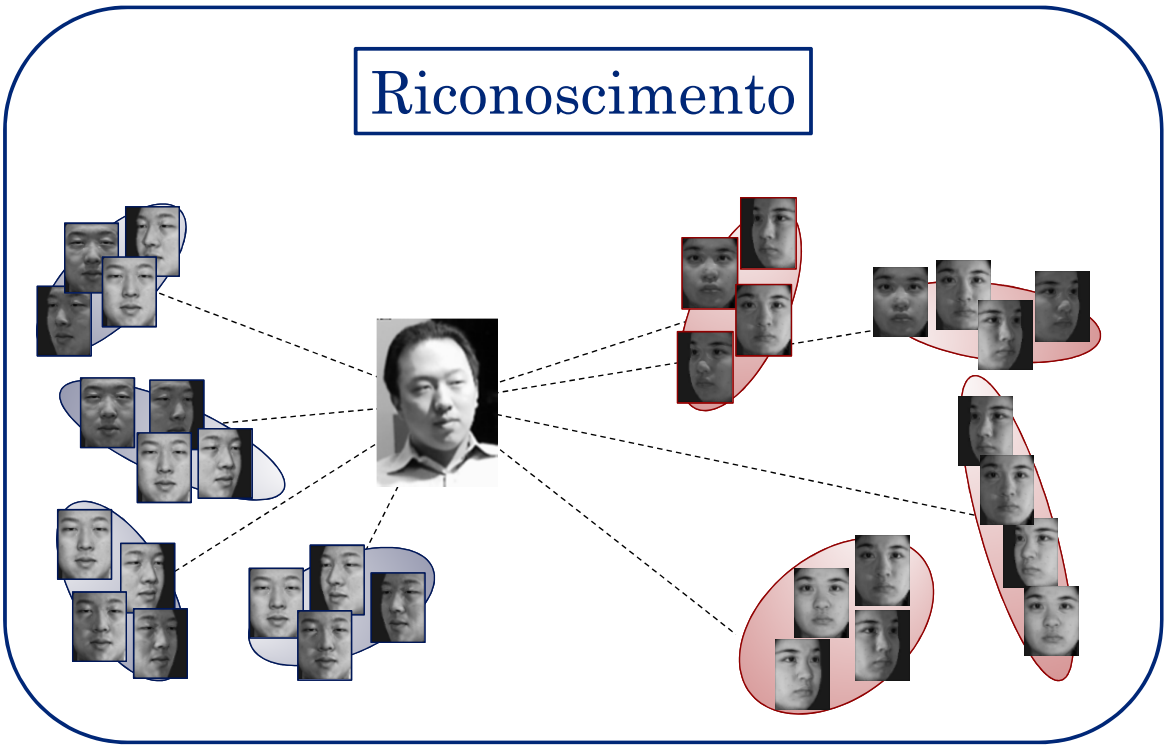
Riconoscimento basato su sottospazi

- Molti metodi di riconoscimento fanno uso di sottospazi per rappresentare i volti degli individui noti.
- I sottospazi possono essere usati in modi diversi.
  - ***Creando un solo sottospazio per tutti gli individui***, all'interno del quale effettuare le ricerche.
    - Eigenface e Fisherface
    - Problemi di aggiornamento
  - ***Creando un sottospazio per ciascun individuo***
    - Il riconoscimento avviene associando un'immagine all'individuo rappresentato dallo spazio più vicino.
  - ***Creando più sottospazi per ogni individuo***. L'uso di più spazi permette spesso di avere una rappresentazione migliore delle possibili variazioni del volto.
    - Anche in questo caso il riconoscimento avviene sulla base della distanza dallo spazio.

Riconoscimento basato su spazi MKL



Riconoscimento basato su spazi MKL



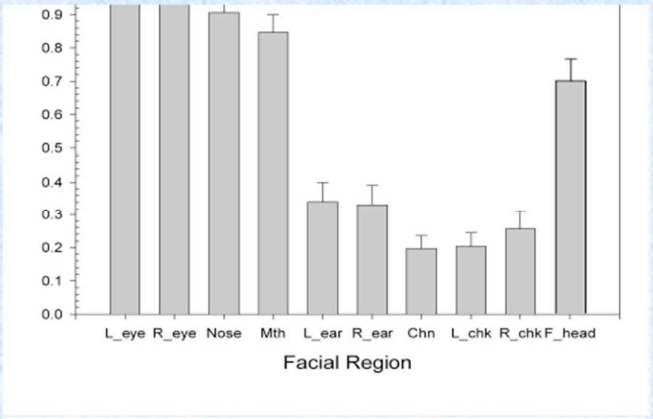
Estrazione di feature

Feature space

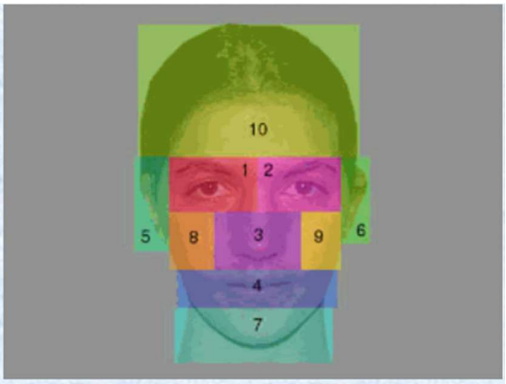
Introduzione

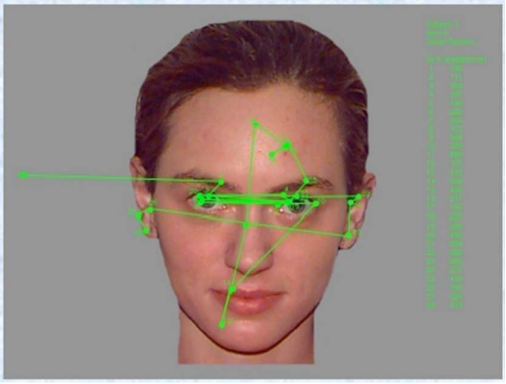
- Le caratteristiche salienti dell'immagine del volto possono essere individuate applicando all'immagine stessa filtri, trasformate, operatori, ciascuno progettato per mettere in luce particolari proprietà.
- I vettori di feature estratti possono eventualmente essere sottoposti a un processo di riduzione della dimensionalità usando una delle tecniche precedentemente descritte.

Come definire le feature del volto?



Mean percentage of times (averaged across viewers) that each facial region was fixated at least once.

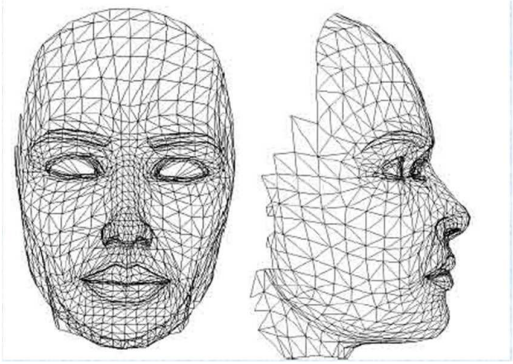




J.H. Henderson et al. "Gaze Control for Face Learning and Recognition by Humans and Machine"; in T. Shipley and P. Kellman (Eds.), *From Fragments to Objects: Segmentation and Grouping in Vision*

Punti di riferimento 2D/3D

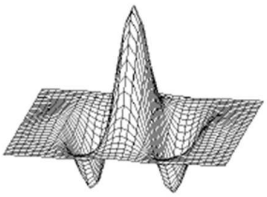
Landmark 2D possono essere definiti e tracciati sull'immagine del volto



Una rappresentazione 3D più complessa

Filtri di Gabor (1)

- Il vettore di feature si ottiene tramite la *convoluzione* dell'immagine con un *banco di filtri*.
- Ogni filtro è costituito da una funzione sinusoidale che è attenuata progressivamente da una gaussiana.
- Il filtro, costituito da una parte reale e una immaginaria, è regolato da 3 parametri:
  - la *frequenza f* della sinusoide
  - l'*orientazione* θ della sinusoide rispetto al piano x,y
  - l'*ampiezza* σ della gaussiana



$$g(x, y, \theta, f) = e^{-\frac{1}{2}\left(\frac{u^2}{\sigma_x^2}+\frac{v^2}{\sigma_y^2}\right)} \cdot \cos(2 \cdot \pi \cdot f \cdot u)$$

dove

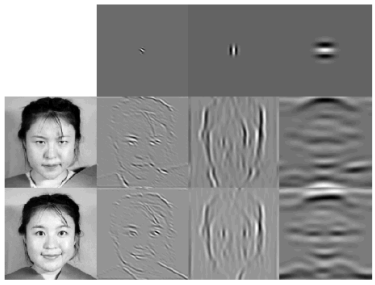
$$u = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

$$v = y \cdot \cos(\theta) - x \cdot \sin(\theta)$$

Banco di filtri

Filtri di Gabor (2)

Come effettuare la convoluzione?



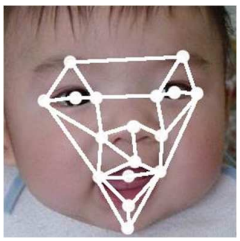
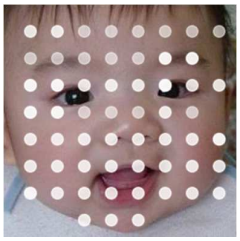
Su tutti i pixel dell'immagine, ma il vettore di feature risultante ha dimensione molto elevata

In corrispondenza dei nodi di una griglia uniforme sovrapposta all'immagine

In corrispondenza dei punti salienti del volto

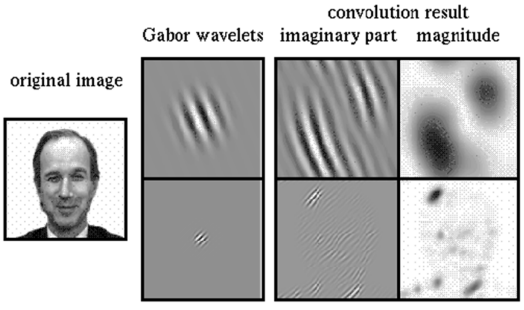
Riconoscimento basato su filtri di Gabor

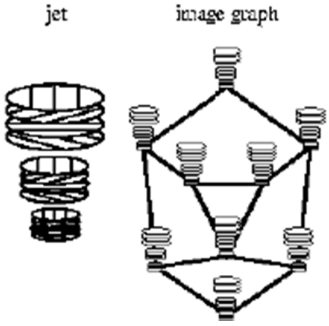
- I principali metodi di riconoscimento del volto che usano filtri di Gabor sono:
  - Elastic Bunch Graph Matching (**EBGM**)
  - Gabor Fisher Classifier (**GFC**)
  - AdaBoosted Gabor Fisher Classifier (**AGFC**)
    - che usa AdaBoost per scegliere feature di Gabor
    - solo le feature più discriminanti sono usate per la successiva classificazione


Elastic Bunch Graph Matching (1)

- Le immagini dei volti sono rappresentate mediante una struttura ad hoc denominata ***labeled graph***. La rappresentazione fa uso di informazioni topografiche, ma anche di template locali. In particolare un grafo etichettato è costituito da un insieme di nodi connessi tra loro da archi. Il grafo per un volto è basato su Gabor Wavelet Transform, ovvero una convoluzione con un insieme di filtri di Gabor.

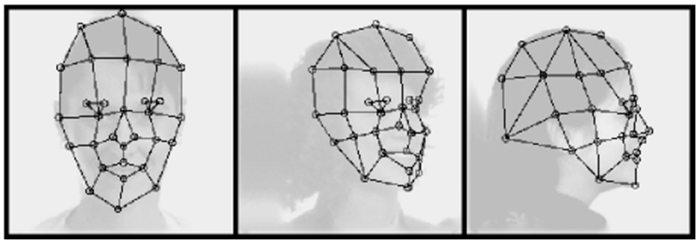




I nodi sono etichettati con **jet**, rappresentazione compatta e robusta di una distribuzione locale di livelli di grigio (relativa alla parte di immagine individuata dal nodo) costituita da tutti i coefficienti ottenuti dall'applicazione di filtri di Gabor con diversa scala e orientazione. I jet risultano robusti rispetto a: traslazioni, distorsioni, variazioni di illuminazione e di scala. Gli archi sono caratterizzati dalle distanze tra i nodi.

Elastic Bunch Graph Matching (2)

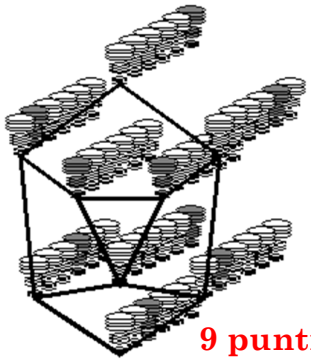
- I grafi possono essere facilmente traslati, ruotati, ridimensionati, e deformati elasticamente. Per individuare volti da diverse viste è utile fare riferimento a particolari punti distintivi (**fiducial points**) come occhi, estremità del naso, angoli della bocca, ecc.
- Questa tecnica può essere utilizzata anche per l'individuazione di volti parzialmente nascosti: in tal caso il matching deve essere effettuato utilizzando un grafo parziale, ma che nella sua struttura mantiene caratteristiche inalterate ed indipendenti dalla parte occultata.



Esempio di grafo adattato per un volto in pose diverse.

Elastic Bunch Graph Matching (3)

- **La geometria di un volto è codificata dagli archi, mentre la distribuzione dei valori di grigio viene codificata dai nodi.** Per descrivere una vista bidimensionale è sufficiente un solo grafo, mentre per rappresentare più viste o per ottenere una vista 3D occorre integrare più grafi.



9 punti cardine

Benché costruito con solo sei volti campione, questo FBG è potenzialmente in grado di rappresentare (6^9= 10'077'696) volti diversi.

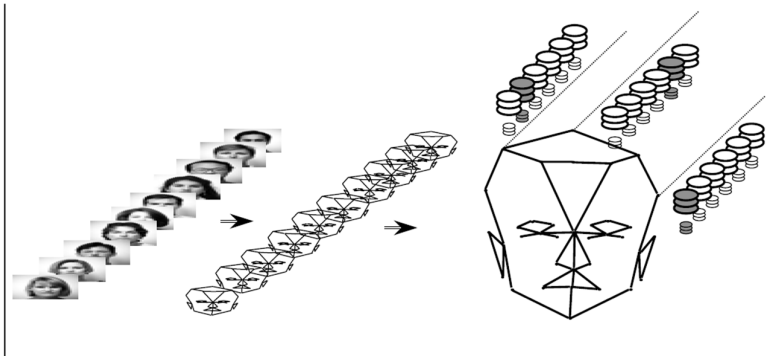
•Un **Face Bunch Graph** (FBG) può rappresentare volti in generale. È progettato per coprire molteplici variazioni nell'aspetto dei volti.

•Combina informazioni di un certo numero di grafi di volto. I nodi sono etichettati con insiemi di jet detti **bunches**, e gli archi sono etichettati con medie di vettori di distanza.

•Durante il confronto di un'immagine, si seleziona il jet più appropriato in ogni bunch (colorato di griglio in figura).

Elastic Bunch Graph Matching (4)

- Un FBG è costruito a partire da un insieme rappresentativo di grafi modello (es. 70) aventi la stessa posa (es. frontale, di profilo, …), dunque la stessa struttura. Ogni nodo è etichettato con tutti i jet presi dai modelli nei medesimi punti. Nuove facce possono essere rappresentate prendendo jet da differenti modelli, ad esempio occhio sinistro da un modello e naso da un altro modello.



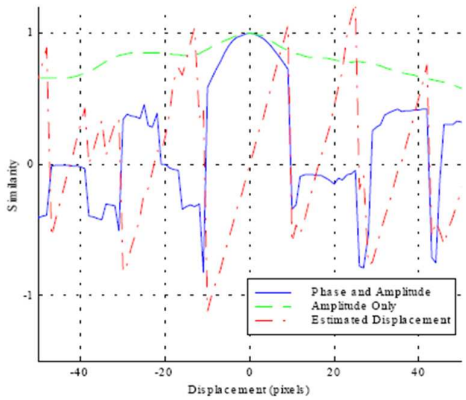
Per nuove immagini, sono generati automaticamente nuovi grafi usando una funzione di similarità tra un FBG e un image graph, funzione che tiene conto della distorsione spaziale e della similarità fra jet.

Elastic Bunch Graph Matching (5)

Jet

$$J = \begin{pmatrix} a_1 e^{i\phi_1} \\ \cdots \\ a_j e^{i\phi_j} \\ \cdots \\ a_{40} e^{i\phi_{40}} \end{pmatrix}$$

5 frequenze, 8 orientazioni



Similarità fra Jet – solo ampiezza

$$S_a(J,J') = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 \sum_j a_j'^2}}$$

Similarità fra Jet –ampiezza e fase

$$S_\phi(J,J') = \frac{\sum_j a_j a'_j \cos(\phi_j - \phi'_j - \vec{d}.\vec{k}_j)}{\sqrt{\sum_j a_j^2 \sum_j a_j'^2}}$$

Scostamento relativo fra i jet

$$\vec{d} = f(J,J')$$

Elastic Bunch Graph Matching (6)

Similarità fra grafi :
Il Bunch Graph B contiene M esempi; ogni grafo ha N nodi and E archi. La similarità tra un grafo semplice G, relativo a un volto, e il Bunch Graph B è calcolata come:

$$S(G,B) = \frac{1}{N}\sum_{n}\max_{m}(S_{\phi}(J_n^G, J_n^{B_m})) - \frac{\lambda}{E}\sum_{e}\frac{\left(\Delta\vec{x}_e^{\,G} - \Delta\vec{x}_e^{\,B}\right)^2}{\left(\Delta\vec{x}_e^{\,B}\right)^2}$$

Feature (Jets) comparison term.

Metric (distances) comparison term. (Distortions)

Elastic Bunch Graph Matching (7)

L'intera procedura

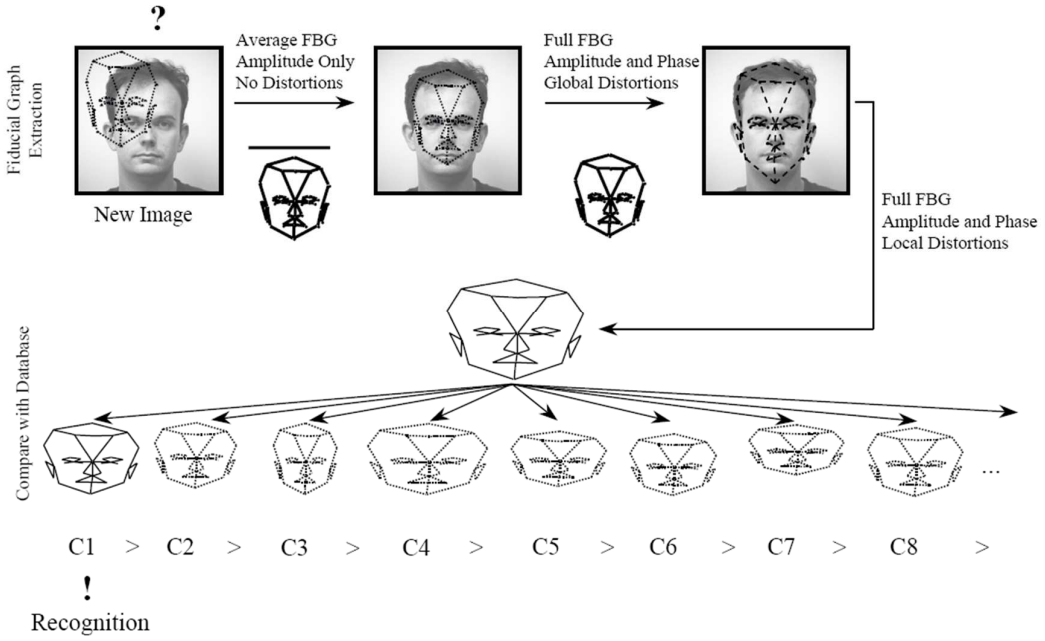
Pre-elaborazione (dipendente dal DB)
- Cropping, padding, resizing

Construzione dei vari FBG (uno per posa)
- Es. 70 esempi caratteristici- individuazione manuale dei fiducial point e costruzione dei fiducial graph

Costruzione dei vari DB (uno per posa)
- Si usano gli FBG per localizzare i fiducial point automaticamente
- Si memorizzano i fiducial graph (es. 45 nodi ciascuno)

Identificazione di una nuova faccia
- Si usano gli FBG per determinare la posa e per localizzare i fiducial point automaticamente
- Si confronta con tutte le occorrenze del database e si classificano i risultati in ordine decrescente di similarità
- Si prende la decisione

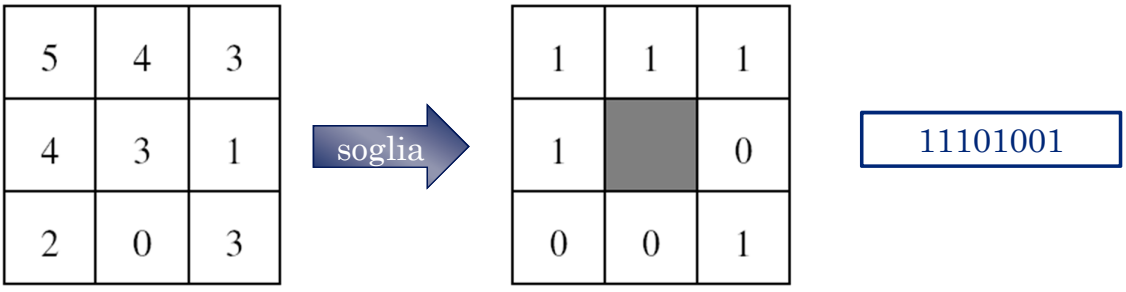
Elastic Bunch Graph Matching (8)



N.B. alcune immagini relative al metodo EBGM sono state prese dal poster *www.snl.salk.edu/~fellous/**posters**/**Bu97poster**/BU**Poster**.pdf*

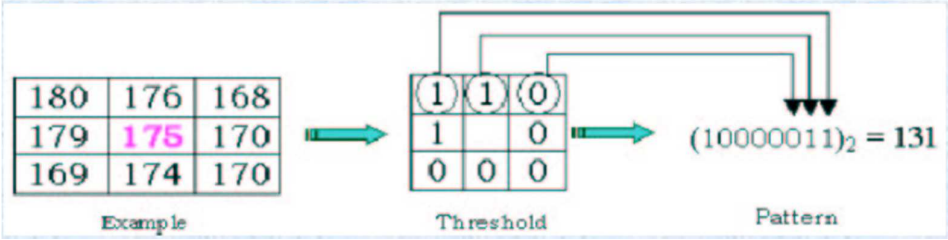
Local Binary Pattern (1)

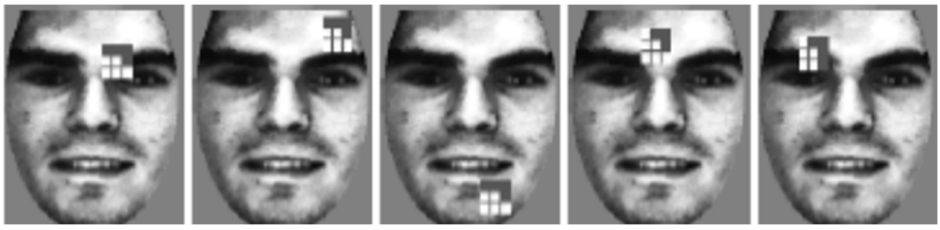
- L'operatore **Local Binary Pattern** (LBP) è stato introdotto da Ojala per l'analisi della tessitura delle immagini.
- L'operatore assegna ai pixel di un'immagine, in un intorno di dimensione 3×3, un valore binario (0 o 1).
- Sia p un pixel dell'intorno e p_c il pixel centrale : il valore binario è assegnato confrontando il valore del pixel p con quello del pixel p_c :
  - Se p ha un valore superiore o uguale a quello di p_c allora a p è assegnato il valore 1, altrimenti il valore 0.



T. Ojala, M. Pietikainen, D. Harwood, "A comparative study of texture measures with classification based on feature distributions", *Pattern Recognition*, vol. 29, no. 1, pp. 51-59, 1996.

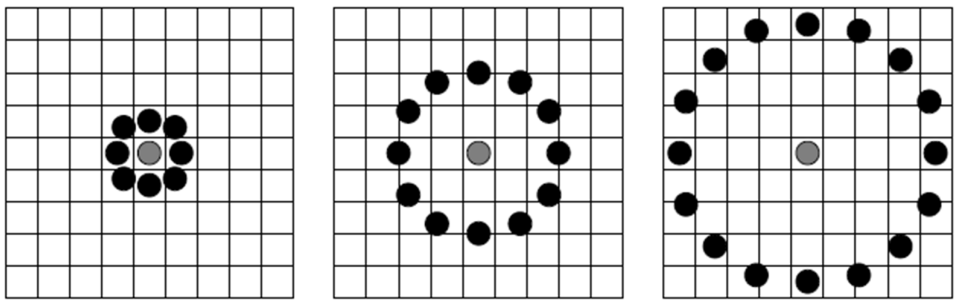
Local Binary Pattern (1): esempio





Local Binary Pattern (2)

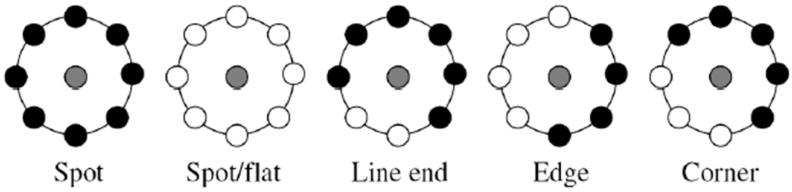
- L'operatore LBP di base è stato esteso per gestire intorni di dimensione variabile di un pixel.
- Si usa il concetto di intorno circolare di un pixel e i punti campione sono individuati tramite interpolazione.
- L'operatore in questo caso è definito da due parametri:
  - il numero di punti campione *P*;
  - il raggio dell'intorno circolare *R*.



P=8, *R*=1.0 *P*=12, *R*=2.5 *P*=16, *R*=4.0

Local Binary Pattern: pattern uniformi

- I pattern binari più interessanti sono quelli *uniformi*, in quanto rappresentano le strutture locali più rilevanti (es. edge, spot, ecc).



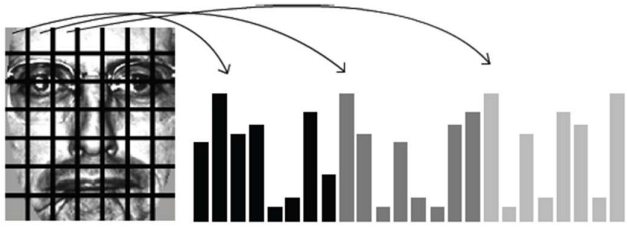
- Un pattern è detto uniforme quando, considerato in modo circolare, contiene al massimo due transizioni 0-1 o 1-0. Ad esempio i pattern 10000011, 11110000, 00000000 sono uniformi.
- Considerare solo i pattern uniformi permette di risparmiare memoria: i pattern totali sono 2^P, mentre i pattern uniformi sono solo $P \cdot (P\text{-}1)+2$.

Pixel con pattern uniforme

Pixel con pattern non uniforme

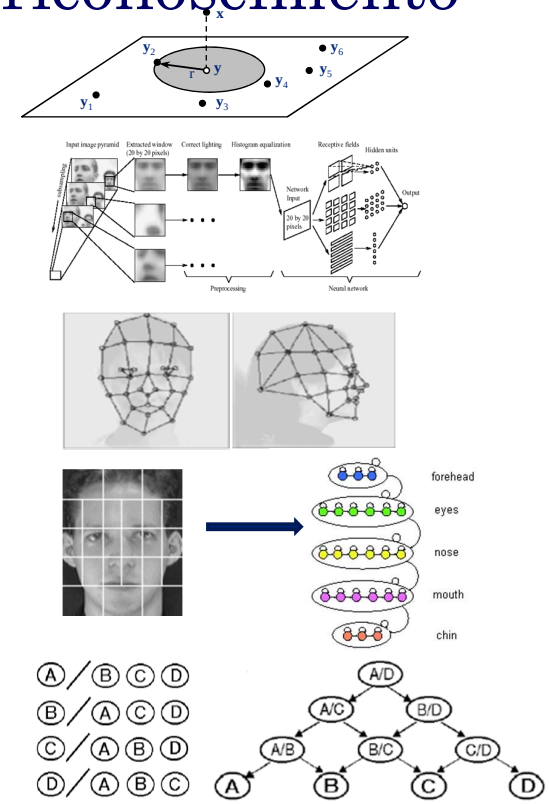
Local Binary Pattern: feature vector

- Il vettore di feature associato a un'immagine è un istogramma (eventualmente normalizzato) calcolato come segue:
  - L'immagine è partizionata in k^2 sottofinestre.
  - Per ogni sottofinestra è costruito un istogramma in cui ciascun bin è associato a uno specifico pattern; i bin sono in totale $\boldsymbol{P \cdot (P\text{-}1)+3}$: $P \cdot (P\text{-}1)$ bin per i pattern con 2 transizioni, 2 bin per i pattern con 0 transizioni, 1 bin per i pattern non uniformi.
  - Il vettore di feature si ottiene concatenando gli istogrammi calcolati per tutte le sottofinestre.



Classificatori per il riconoscimento

- I vettori di feature estratti possono essere usati per la creazione dei template rappresentativi degli utenti del sistema e per il riconoscimento.
- In letteratura esistono moltissimi approcci al riconoscimento che usano classificatori e/o tecniche di riconoscimento diversi:
  - **Metodi basati su sottospazi**
  - **Reti neurali**
  - **Modelli deformabili**
    - Es. Elastic Bunch Graph
  - **Hidden Markov Models**
  - **Support Vector Machines**
    - Approccio *One vs. all* o *pairwise*

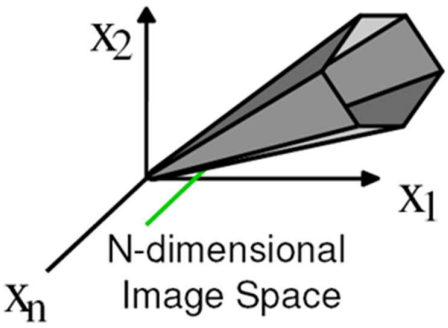


Principali problematiche

- Le principali problematiche nel riconoscimento del volto sono legate a cambiamenti di posa e illuminazione che possono compromettere l'accuratezza del sistema in modo significativo.
- I risultati di alcune recenti campagne di valutazione hanno confermato queste problematiche, mostrando che le prestazioni dei sistemi di riconoscimento sono buone in condizioni di acquisizione controllate, ma crollano in presenza di immagini più realistiche.
- Molte applicazioni reali (ad esempio videosorveglianza), e soprattutto quelle non supervisionate, comportano l'acquisizione di immagini in condizioni non controllate

Illumination cone

- L'illumination cone è un modello che rappresenta un oggetto sotto tutte le possibili condizioni di illuminazione.
- Assunzione:
  - La superficie dell'oggetto ha una funzione di riflettanza lambertiana.
  - La superficie dell'oggetto ha una forma convessa.
- Ipotesi:
  - Le immagini dell'oggetto sotto tutte le possibili condizioni di illuminazione formano un **cono convesso** nello spazio delle immagini.
  - Il cono d'illuminazione può essere stimato a partire da 3 immagini dell'oggetto, acquisite in diverse condizioni di illuminazione.



Una rappresentazione grafica del cono d'illuminazione disegnato nello spazio dell'immagine

A. S. Georghiades , D. J. Kriegman , P. N. Belhumeur, "Illumination Cones for Recognition under Variable Lighting: Faces", *In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p.52-58, 1998.