Analisi del volto

Localizzazione

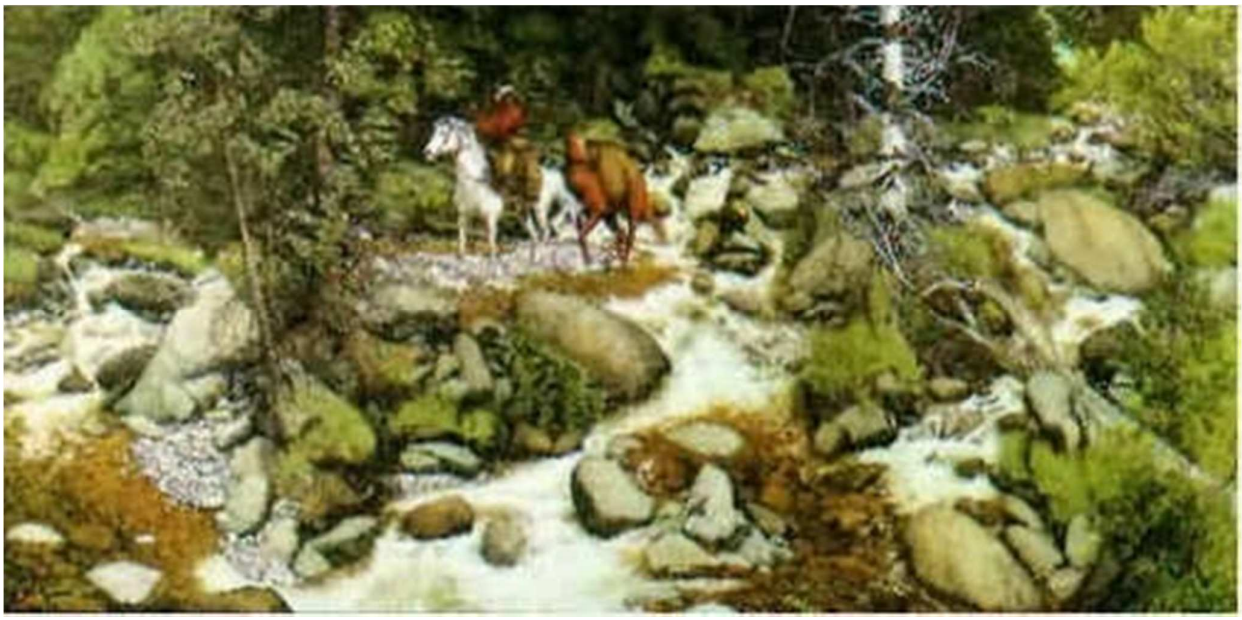
Annalisa Franco
annalisa.franco@unibo.it

Dario Maio
dario.maio@unibo.it

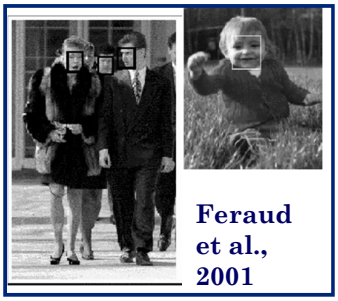
Introduzione

- Problema: data una singola immagine o una sequenza video rilevare la presenza di uno o più volti e localizzarne la posizione all'interno dell'immagine.
- È necessaria l'indipendenza rispetto a:
  - posizione, orientazione, scala, espressione del volto;
  - fattori esterni quali l'illuminazione o la presenza di uno sfondo complesso.

Quante facce vedete?

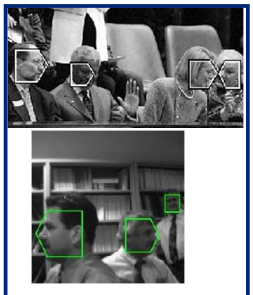


Risultati di algoritmi allo stato dell'arte



Feraud et al., 2001

Maio et al., 2000

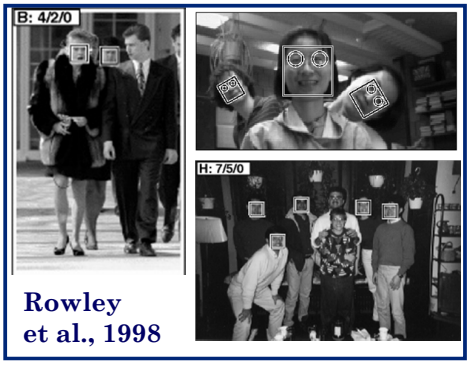


Schneiderman et al., 2000

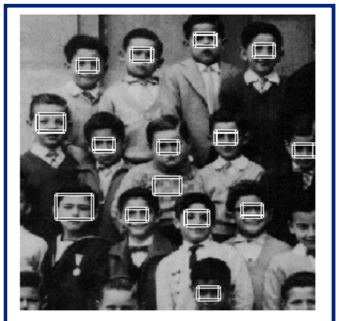
Garcia et al., 1999

Sung et al., 1998

Yow et al., 1997



Rowley et al., 1998

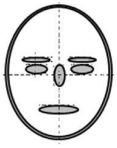


Lew et al., 1996

Approcci alla localizzazione (1)

Tecniche feature-based

- Usano esplicitamente la conoscenza dell'aspetto del volto, caratterizzato da un insieme di feature di basso livello:
  - ***Low-level analysis:*** proprietà dei pixel
    - Edges
    - Colore della pelle
  - ***Feature analysis:*** informazioni sulla geometria del volto
    - Constellation
    - Feature searching
  - ***Template matching:*** modello standard di un volto definito manualmente o descritto da una funzione
    - Correlazione
    - Snakes
    - Active Shape Models

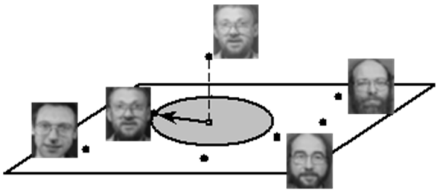


Approcci alla localizzazione (2)

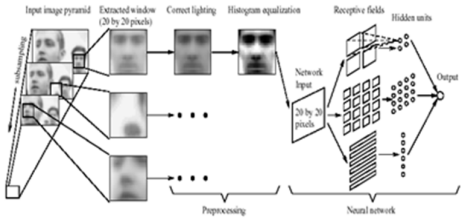
Tecniche image-based

- Affrontano il problema della localizzazione del volto come un generico problema di pattern recognition;
- L'obiettivo è imparare a riconoscere un'immagine di un volto sulla base di alcuni esempi di training.

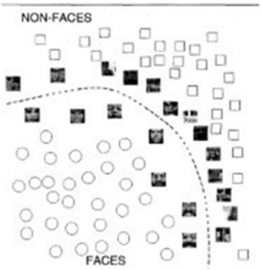
Subspace methods



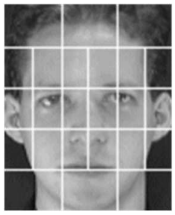
Neural networks

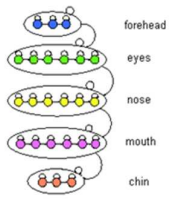


Support Vector Machine



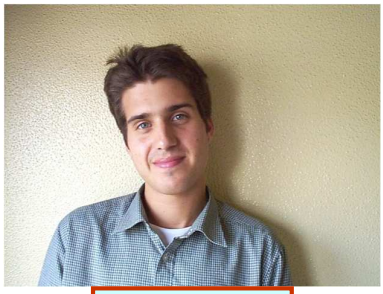
Hidden Markov Model



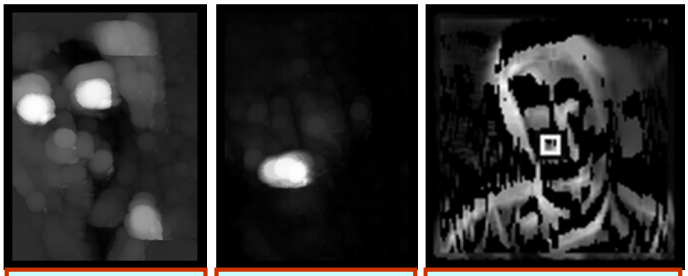


Localizzazione in immagini a colori

- Algoritmo:
  - Compensazione della luce
  - Localizzazione basata sul colore della pelle
  - Localizzazione delle principali caratteristiche del volto (occhi, bocca, contorno del viso).



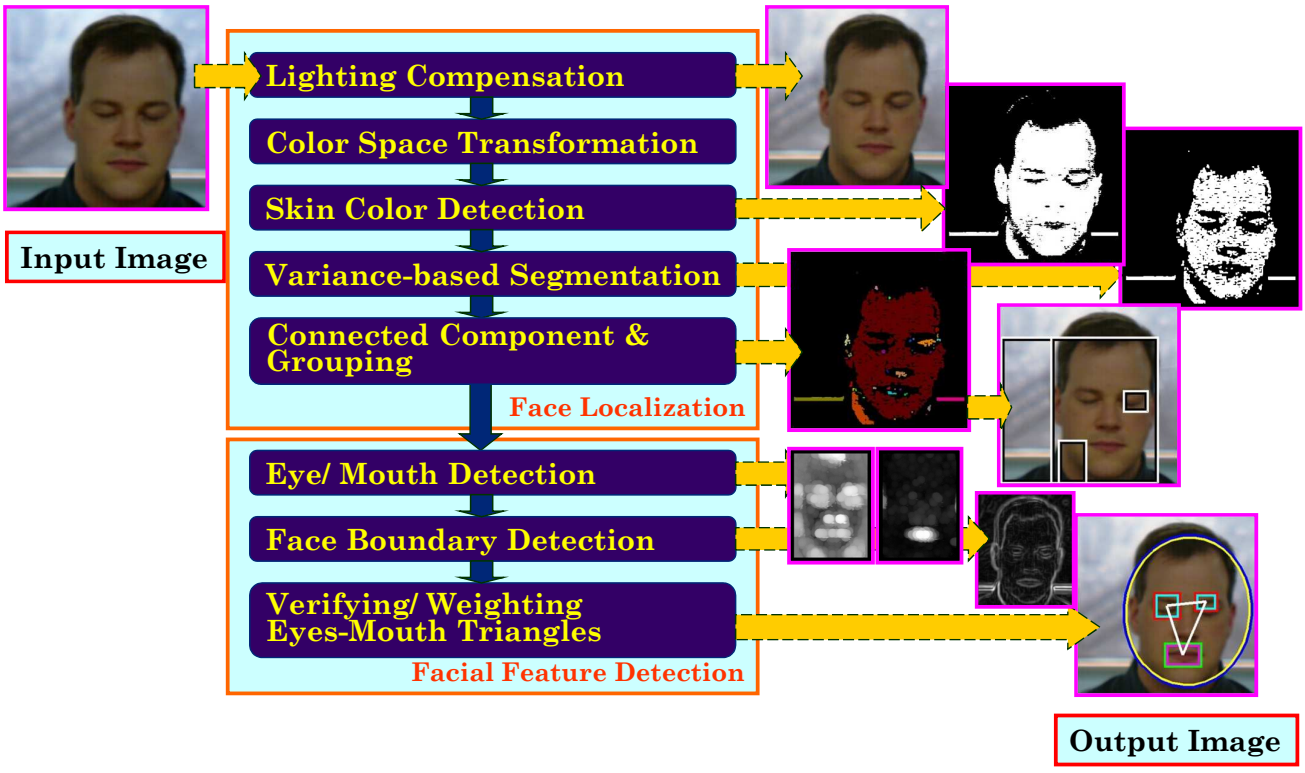
Face Image



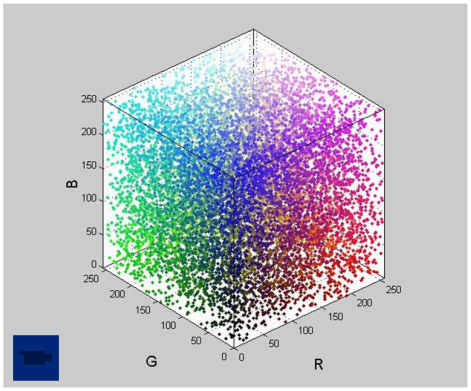
Eye Map | Mouth Map | Boundary Map

R.L. Hsu, M.A. Mottaleb, A.K. Jain, "Face Detection in Color Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696-706, 2002.

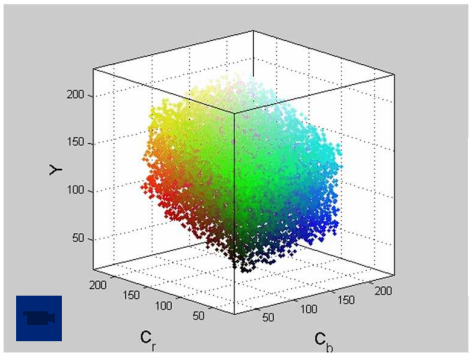
L'algoritmo di localizzazione



Spazio colorimetrico



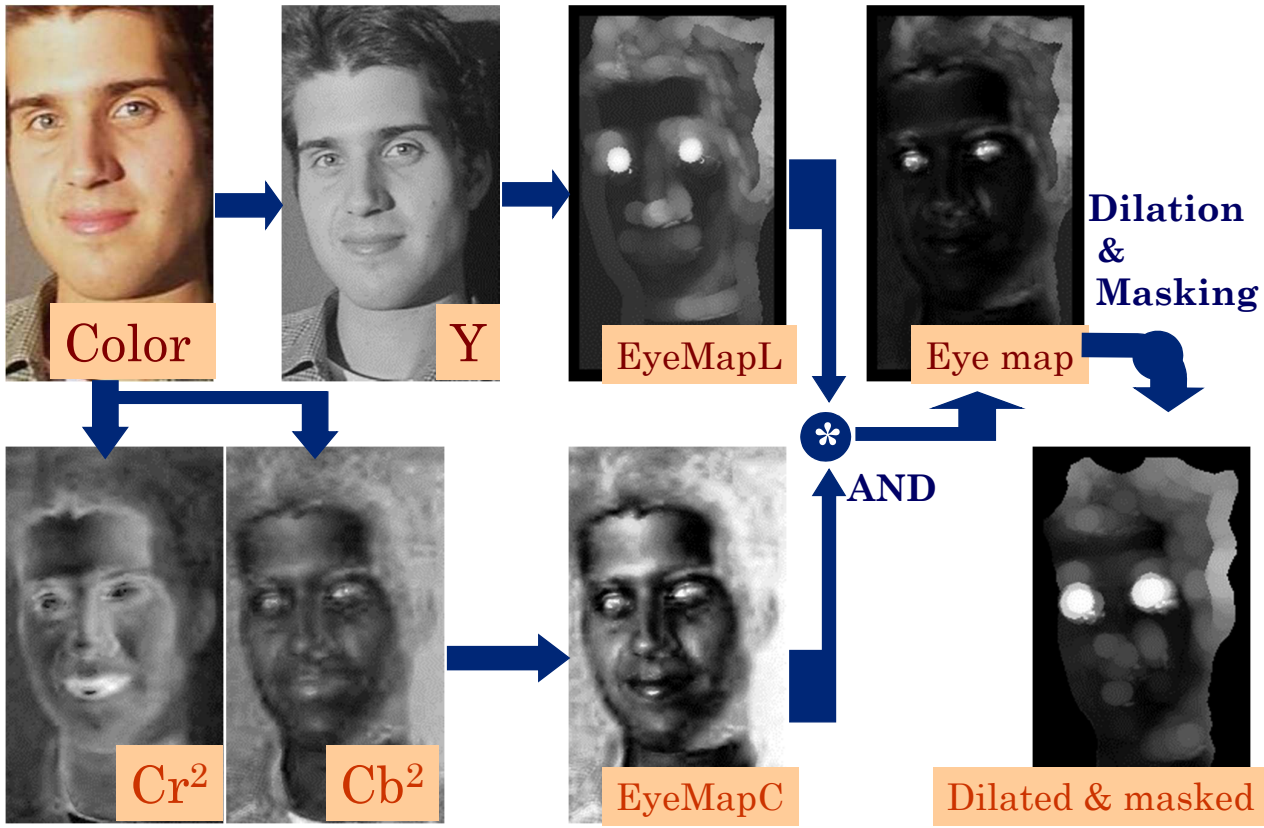
Spazio RGB (Red-Green-Blue)



Spazio Luma-Chroma(YCbCr)

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256} \begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix} \bullet \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Mappa degli occhi (1)



Mappa degli occhi (2)

- L'algoritmo costruisce due diverse mappe degli occhi (**crominanza** e **luminanza**).
  - *Crominanza*: la regione circostante gli occhi è caratterizzata da valori elevati di C_b e bassi di C_r:

$$EyeMapC = \frac{1}{3}\left\{\left(C_b^2\right)+\left(\tilde{C}_r^2\right)+\left(\frac{C_b}{C_r}\right)\right\} \qquad \tilde{C}_r = 255 - C_r$$

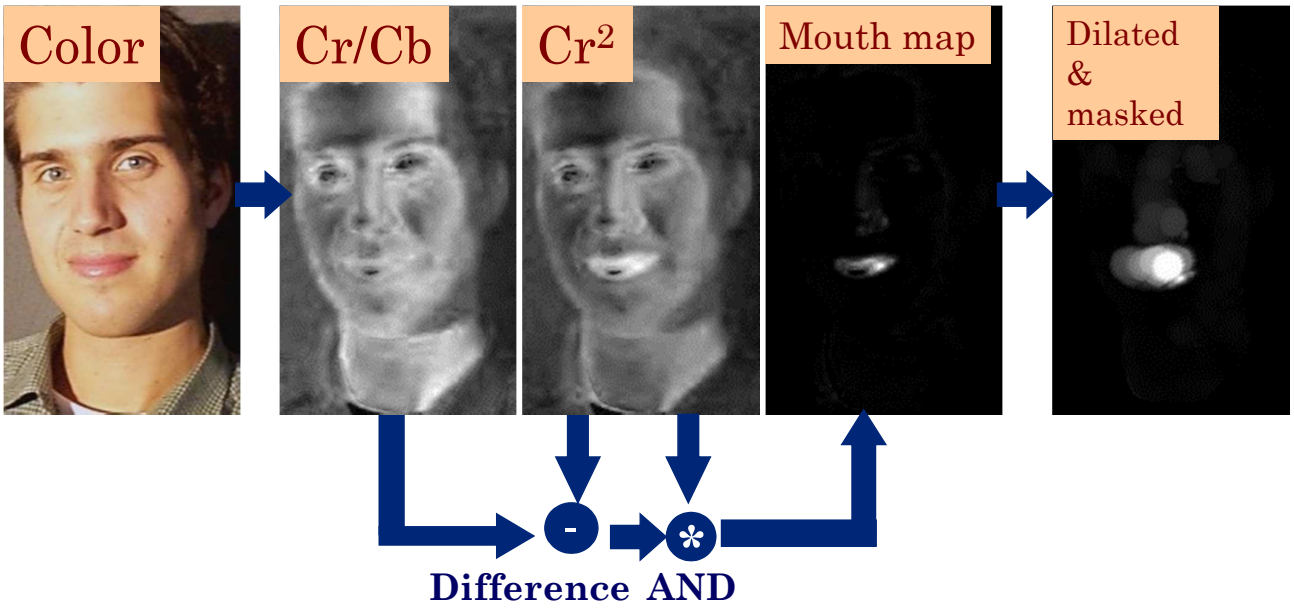
  - *Luminanza*: gli occhi contengono solitamente sia regioni chiare sia zone più scure; opportuni operatori morfologici possono evidenziare tali caratteristiche.

$$EyeMapL = \frac{Y(x,y)\oplus g_\sigma(x,y)}{Y(x,y)\ominus g_\sigma(x,y)+1}$$

- La mappa relativa alla crominanza è sottoposta a equalizzazione dell'istogramma e poi combinata con la mappa risultante dalla luminanza (AND).
- La mappa risultante dalla combinazione è poi sottoposta a dilatazione, mascherata e normalizzata per eliminare le altre regioni del volto ed evidenziare gli occhi.

Mappa della bocca (1)

La bocca è caratterizzata da una preponderanza della componente rossa rispetto a quella blu.



Mappa della bocca (2)

- Nella regione della bocca la componente C_r è maggiore di C_b; inoltre la risposta a C_r/C_b è bassa, mentre la risposta a C_r^2 è elevata.

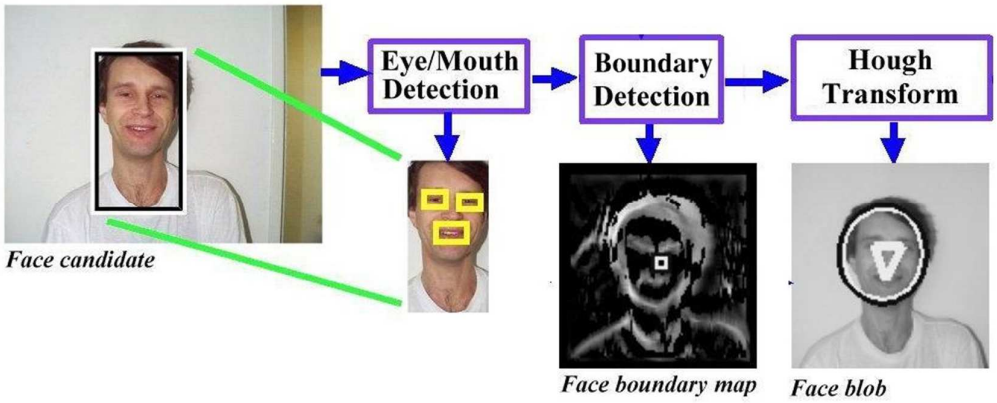
$$MouthMap = C_r^2 \cdot \left(C_r^2 - \eta \cdot C_r / C_b\right)^2$$

$$\eta = 0.95 \cdot \frac{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)^2}{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)/C_b(x,y)}$$

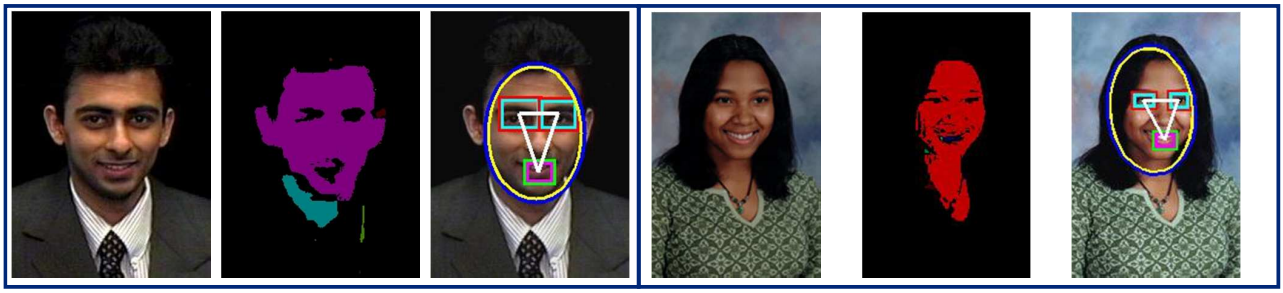
n è il numero di pixel contenuti nella maschera che delimita il volto (*FG*)

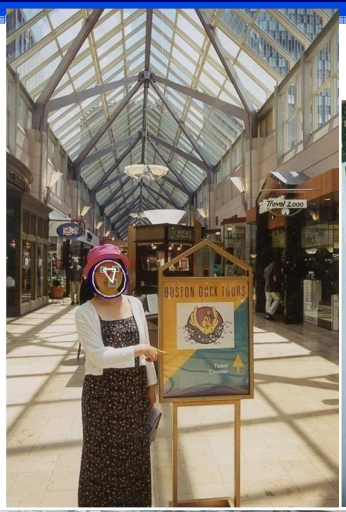
Mappa del contorno del volto

- Si analizzano tutti i triangoli occhi bocca dati dalla combinazione di due occhi e una bocca candidati.
- Ogni triangolo viene controllato verificando:
  - Variazioni nel componente luma e media del gradiente delle orientazioni;
  - Geometria e orientazione del triangolo;
  - Presenza di un contorno del volto attorno al triangolo.
- A tutti i triangoli che verificano le condizioni precedenti è assegnato uno score e viene infine selezionato il triangolo che ottiene lo score maggiore (se superiore a una soglia).

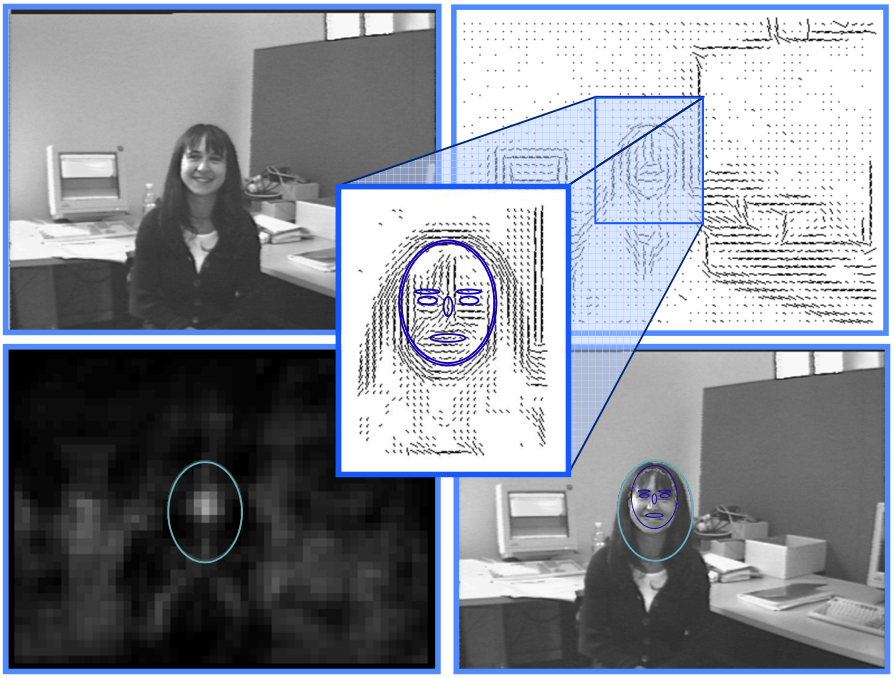


Risultati di localizzazione





Localizzazione su immagini a livelli di grigio



Trasformata Hough sulle orientazioni (spazio Hough)

Template Matching (su orientazioni)

Immagine delle orientazioni locali (multi-risoluzione)

Directional Image computation

Approximate location

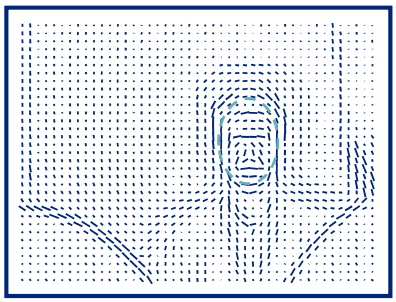
Directional Image refining

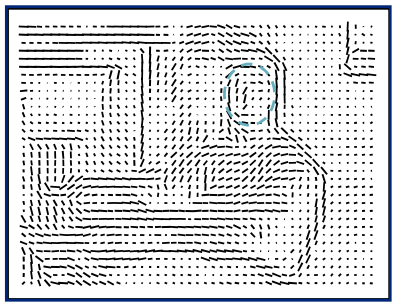
Fine location & face verification

D. Maio, D. Maltoni, "Real-time face location on gray-scale static images", *Pattern Recognition*, vol. 33, no. 9, pp. 1525-1539, 2000.

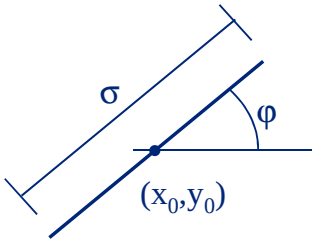
Localizzazione approssimata

Ricerca di blob ellittici sull'immagine delle orientazioni.





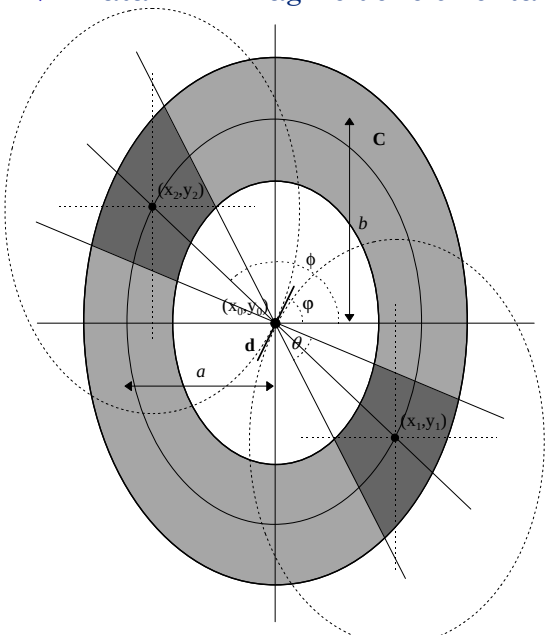
Un generico elemento $\mathbf{d} \in \mathbf{D}$



1. Calcolo dell'immagine **D** delle orientazioni
2. Generalized Hough Transform (GHT) usando settori di corone ellittiche come template.
3. Ricerca di massimi in un array di accumulatori

Localizzazione approssimata con GHT

- Data un'immagine delle orientazioni **D** e un array di accumulatori **A**:



Reset **A** ;

∀ element $\mathbf{d} \in \mathbf{D}$

{ **T** = *compute template*((x_0,y_0), φ);

∀ pixel $(x,y) \in$ **T**

{ $\mathbf{A}[x,y] = \mathbf{A}[x,y] + \sigma \cdot weight_{\mathbf{T}}(x,y)$; }

}

Per ogni elemento $\mathbf{d} \in \mathbf{D}$ si calcola un differente template **T** al fine di aggiornare l'array di accumulatori **A** tenendo conto di:

- un intervallo di variazione :

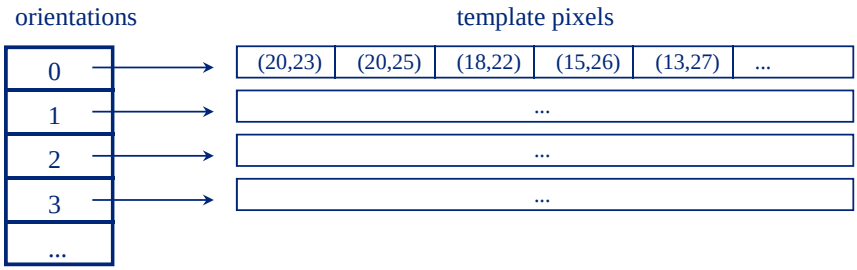
$$a_{min} \le a \le a_{max}\,,\; b_{min} \le b \le b_{max}$$

- un errore angolare θ

I punti (x_1, y_1) e (x_2, y_2) corrispondono ai centri delle 2 sole ellissi di semiassi (a, b) il cui bordo è tangente a **d** nel punto (x_0, y_0). Essendo interessati a tutte le ellissi nel range $(a_{min}...a_{max}\,,\; b_{min}...b_{max})$ è necessario considerare come centri tutti i punti giacenti sui 2 segmenti determinati dall'intersezione della retta passante per (x_1, y_1) e (x_2, y_2) con la corona **C**. Assumendo un errore massimo angolare si determina il luogo geometrico dei possibili centri di ellissi in grado di generare in (x_0, y_0) un vettore **d** di direzione φ.

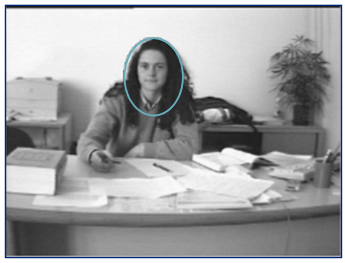
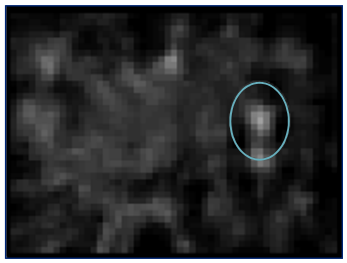
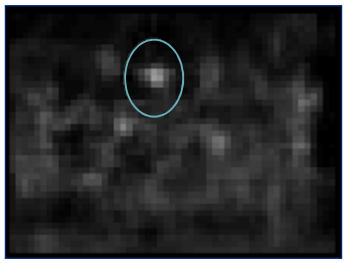
Implementazione efficiente

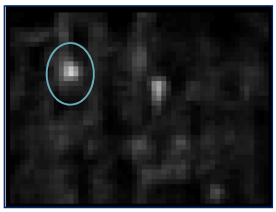
- ☑ **Discretizzazione** delle orientazioni degli elementi in **D** (es. *256 valori*)
- ☑ **Pre-Calcolo** dei template **T**; usando coordinate relative rispetto al centro dell'ellisse, il number dci differenti template corrisponde al numero di differenti



- ☑ **Discretizzazione** dell'accumulatore **A**

Risultato della localizzazione

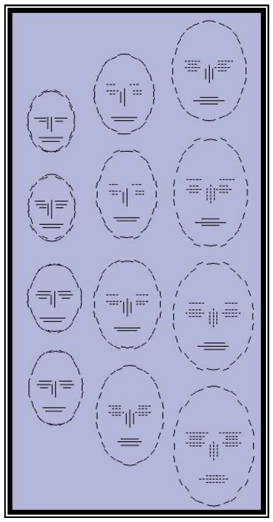







... un errore

Raffinamento della localizzazione

- Un passo di **localizzazione fine** è necessario per:
  - scartare i massimi di Hough che pur essendo ellissi non sono volti
  - localizzare più esattamente il centro

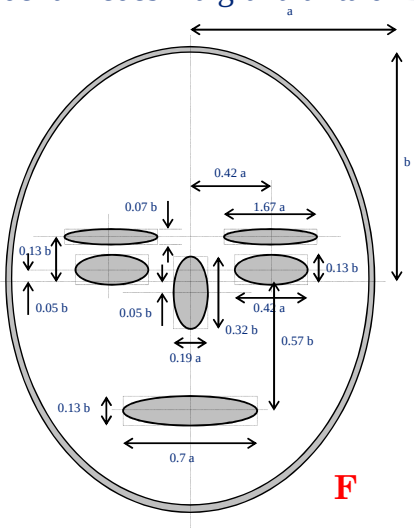


Il primo stadio trasmette al secondo una posizione alla volta (la più probabile); il secondo stadio continua a richiedere nuove posizioni al primo stadio fintanto che non è individuato un volto "valido" oppure non sono esaurite le posizioni candidate.

- Un esempio di template "direzionali" usati per il template matching in un intorno delle posizioni identificate dal primo stadio (Hough).
- L'immagine delle orientazioni è localmente ricampionata a maggiore risoluzione per ottenere una migliore accuratezza.

Maschere parametriche

F è una maschera, definita in modo parametrico rispetto ad **a** e **b**, che assume valori non nulli (**orientazione + modulo**) solo in corrispondenza dei nodi di una griglia della medesima granularità di **D**.



F

Inoltre, solo i nodi di F che appartengono alle 5 regioni grigie hanno moduli non nulli:

- **bocca, occhi e sopracciglia → orizzontale**
- **naso → verticale**
- **bordo→ tangente all'ellisse**

Ogni regione ha un peso globale unifomemente distribuito tra i suoi elementi

Discretizzando a e b nel range (a_{min}...a_{max} , b_{min}...b_{max}) è possibile pre-calcolare un insieme di maschere F ={ $\mathbf{F}_1$, $\mathbf{F}_2$, ..$\mathbf{F}_m$ }, in ciascuna della quali la posizione dei nodi è espressa in coordinate relative rispetto al centro, che possono essere efficientemente utilizzate per calcolare il grado di correlazione in qualsiasi posizione dell'immagine delle orientazioni.

Esempi di localizzazione fine

Il localizzatore di Viola e Jones (1)

- La proposta di Paul Viola e Michael Jones, introdotta in generale per la localizzazione di oggetti, è stata applicata con successo all'individuazione di volti. Prevede di creare un **classificatore** che inizialmente è addestrato mediante multiple istanze della classe da individuare (*esempi positivi*), e varie istanze di *esempi negativi*, ovvero immagini che non contengono alcun oggetto della classe in esame.

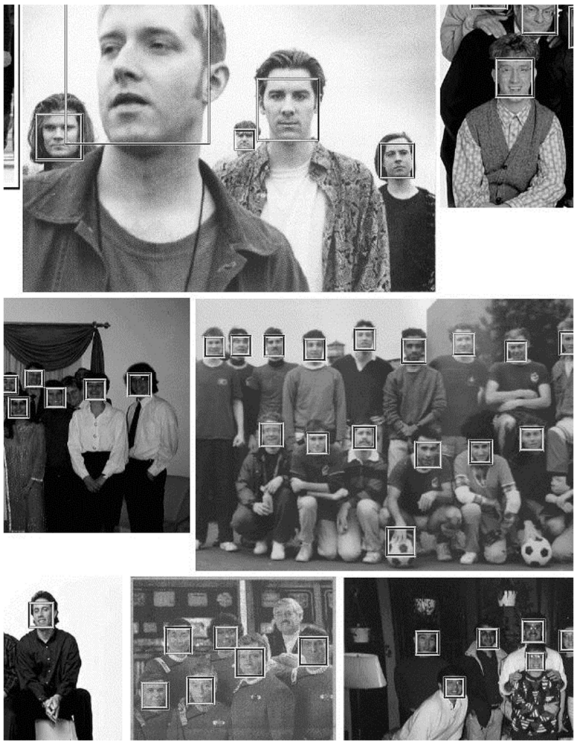
- Durante il training sono estratte diverse caratteristiche dagli esempi e selezionate quelle che risultano particolarmente discriminanti. Questo tipo di informazione è racchiusa nei parametri del modello statistico.

- Se il classificatore addestrato non trova un oggetto che invece è presente (*miss*) oppure ne indica erroneamente la presenza *(false alarm),* si può ricalibrare il suo addestramento aggiungendo gli esempi corrispondenti (positivi o negativi) al training set.

P. Viola, M. J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", in proc. IEEE International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 511-518, 2001.

Il localizzatore di Viola e Jones (2)

- Metodo di tipo image-based.
- Usa un classificatore in grado di associare un pattern in input a una delle due classi volto/non volto:
  - Le immagini dei volti hanno proprietà comuni;
  - Le immagini che non rappresentano volti sono estremamente irregolari.

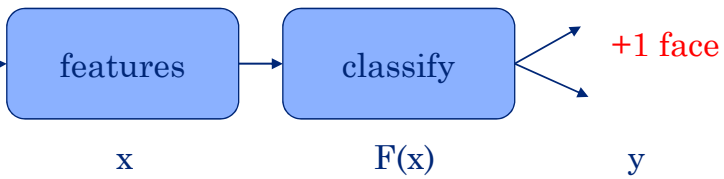


Paul Viola, Michael Jones, Robust Real-Time Face Detection, International Journal of Computer Vision 57(2), 137–154, 2004

Il localizzatore di Viola e Jones (3)

I tre più importanti contributi dell'approccio sono:

- Estrazione e valutazione di **Haar-like feature**
- Classificazione mediante **boosting** (costruzione di un classificatore robusto come combinazione di molteplici classificatori semplici)
- **Multiscale detection**



la localizzazione è effettuata facendo scorrere una finestra di ricerca (le cui dimensioni possono variare) sull'immagine, estraendo le feature presenti nella finestra e classificando la finestra come volto o non volto.

AdaBoost learning procedure

- Obiettivo: costruire un classificatore non lineare complesso $H_M(x)$ come combinazione lineare di M classificatori più semplici detti classificatori deboli (*weak classifiers*):

$$H_M(x) = \frac{\sum_{m=1}^{M} \alpha_m h_m(x)}{\sum_{m=1}^{M} \alpha_m}$$

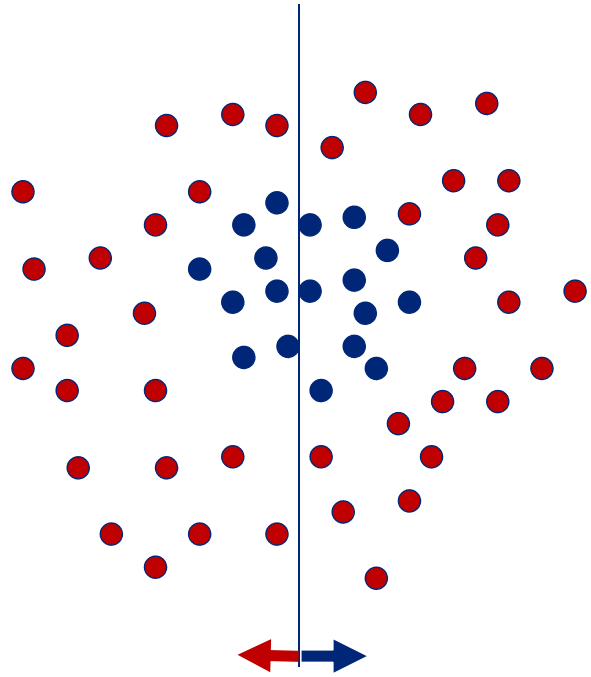
- dove x è un pattern da classificare,
- $h_m(x) \in \{-1,+1\}$ sono i classificatori deboli,
- $\alpha_m \geq 0$ sono i corrispondenti fattori di peso,
- $\sum_{m=1}^{M} \alpha_m$ è il fattore di normalizzazione

N.B. : nella versione discreta $h_m(x)$ assume un valore discreto in $\{-1,+1\}$

AdaBoost: idea di base

- AdaBoost è una tecnica di addestramento che ha lo scopo di apprendere la sequenza ottimale di classificatori deboli e i corrispondenti pesi.
- Richiede un insieme di pattern di training $\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$, dove $y_i \in \{-1,+1\}$ è l'etichetta della classe associata al pattern $x_i \in \Re^n$.
- Durante l'apprendimento è calcolata e aggiornata una distribuzione di pesi $[w_1,w_2,\ldots,w_N]$ associati ai pattern di training ; w_i è associato al pattern (x_i,y_i).
- Dopo l'iterazione m, è assegnato ai pattern più difficili da classificare un peso $w_1^{(m)}$ superiore, cosicché alla successiva iterazione $m+1$ tali pattern riceveranno un'attenzione maggiore.
- AdaBoost assume di avere a disposizione una procedura per l'apprendimento di un classificatore debole a partire da un insieme di pattern di training, data la distribuzione $[w_i^{(m)}]$.

Un esempio



Classificatore lineare debole

Ogni punto ha un'etichetta che ne indica la classe:

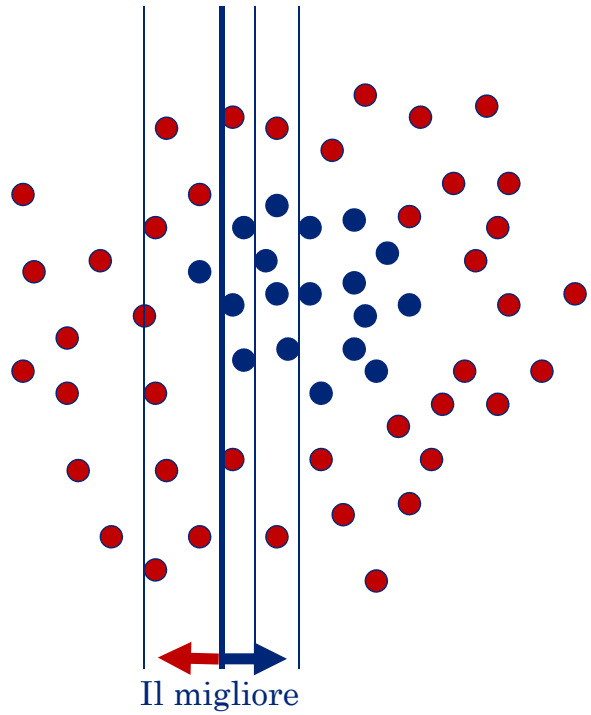
$$y_i = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

e un peso:

$$w_i = 1$$

ripreso da Antonio Torralba @MIT

Un esempio



Il migliore

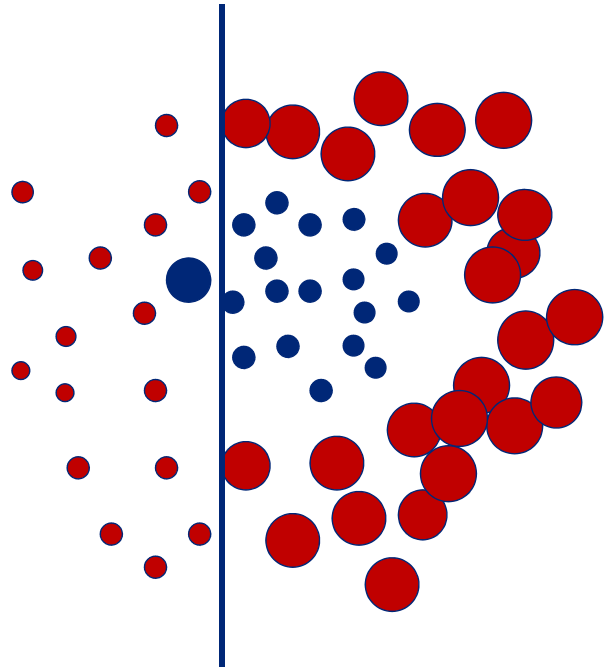
Ogni punto ha un'etichetta che ne indica la classe:

$$y_i = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

e un peso:

$$w_i = 1$$

Un esempio



Ogni punto ha un'etichetta che ne indica la classe:

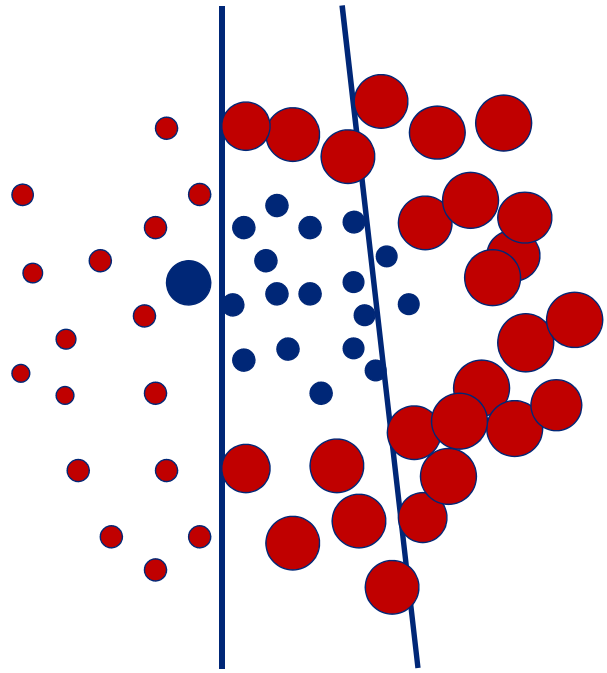
$$y_i = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

Aggiornamento pesi:

$$w_i \leftarrow \exp\{-y_i H_M(x_i)\}$$

Formuliamo ora un nuovo problema…

Un esempio



Ogni punto ha un'etichetta che ne indica la classe:

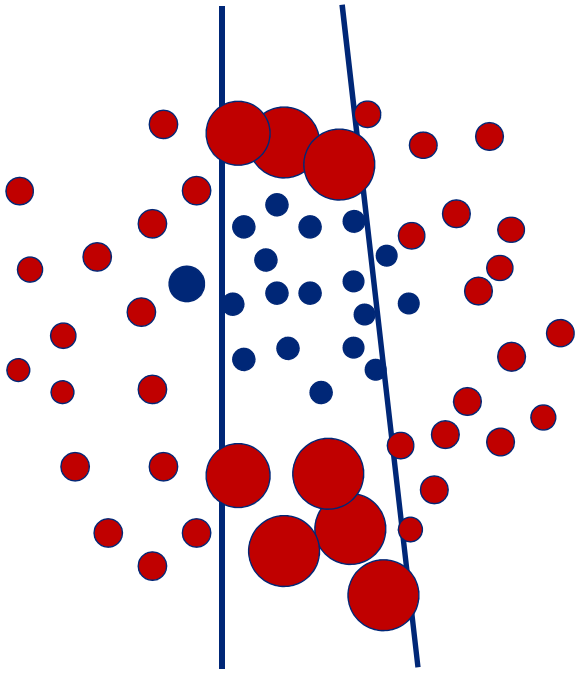
$$y_i = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

Aggiornamento pesi:

$$w_i \leftarrow \exp\{-y_i H_M(x_i)\}$$

Formuliamo ora un nuovo problema…

Un esempio



Ogni punto ha un'etichetta che ne indica la classe:

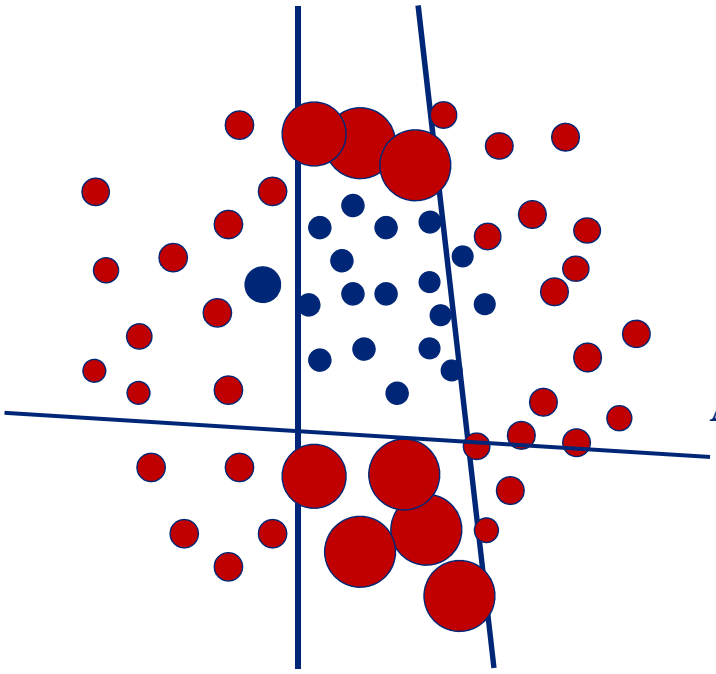
$$y_i = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

Aggiornamento pesi:

$$w_i \leftarrow \exp\{-y_i H_M(x_i)\}$$

Formuliamo ora un nuovo problema…

Un esempio



Ogni punto ha un'etichetta che ne indica la classe:

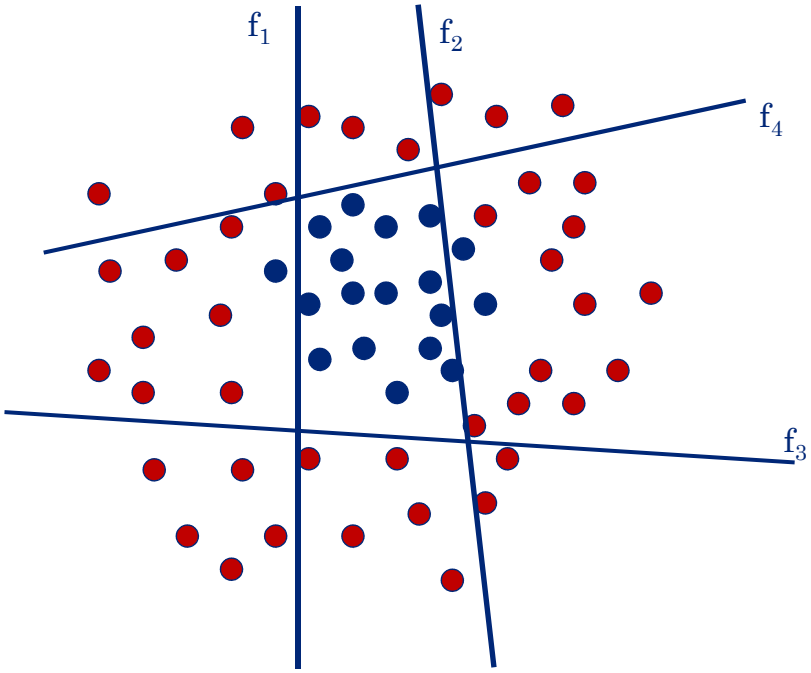
$$y_i = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

Aggiornamento pesi:

$$w_i \leftarrow \exp\{-y_i H_M(x_i)\}$$

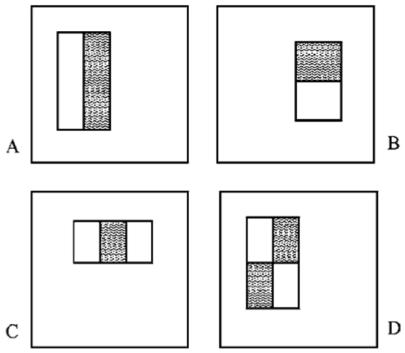
Formuliamo ora un nuovo problema…

Un esempio



Il classificatore robusto (**non lineare**) è costituito da una combinazione di classificatori deboli (**lineari**).

Haar-like features



- Il localizzatore di Viola e Jones, nelle sue molteplici varianti, utilizza alcuni tipi di feature di base. Ogni feature è posizionata in una sottoregione di una sottofinestra dell'immagine.
- Le feature sono applicate modificandone la dimensione, la forma, e la posizione nella sottofinestra.

Esempio: A, B feature con due rettangoli, C con tre, D con quattro

Usando una sottofinestra di dimensione 24×24, al variare della posizione, forma e dimensione si possono ottenere decine di migliaia di feature.

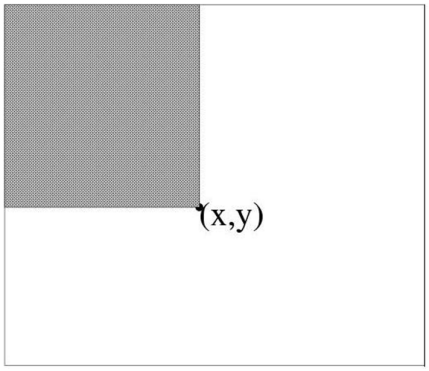
Queste feature sono utilizzate poiché:

- sono efficaci ai fini della localizzazione del volto;
- possono essere calcolate in modo efficiente usando l'***immagine integrale***.

L'immagine integrale (2)

L'***immagine integrale*** in posizione (x,y) è la somma del valore dei pixel sopra e a sinistra di (x,y):

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

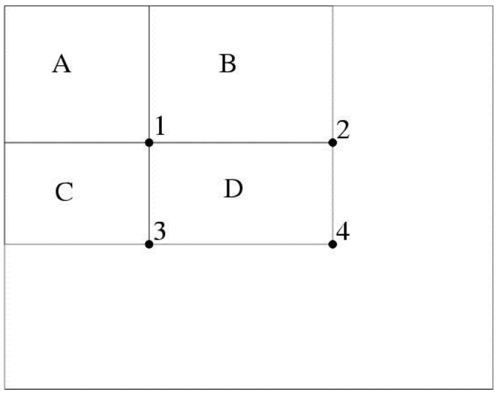


L'immagine integrale può essere calcolata con una sola scansione dell'immagine originale usando le seguenti formule ricorsive:

$$S(x,y) = S(x,y\text{-}1) + I(x,y)$$
$$II(x,y) = II(x\text{-}1,y) + S(x,y)$$

dove $I(x,y)$ è il valore del pixel nell'immagine originale, $S(x,y)$ è la somma cumulativa per righe, $S(x,-1)=0$ e $II(-1,y)=0$.

Calcolo efficiente di feature rettangolari



Usando l'immagine integrale, è possibile calcolare la somma del valore dei pixel in qualsiasi rettangolo.

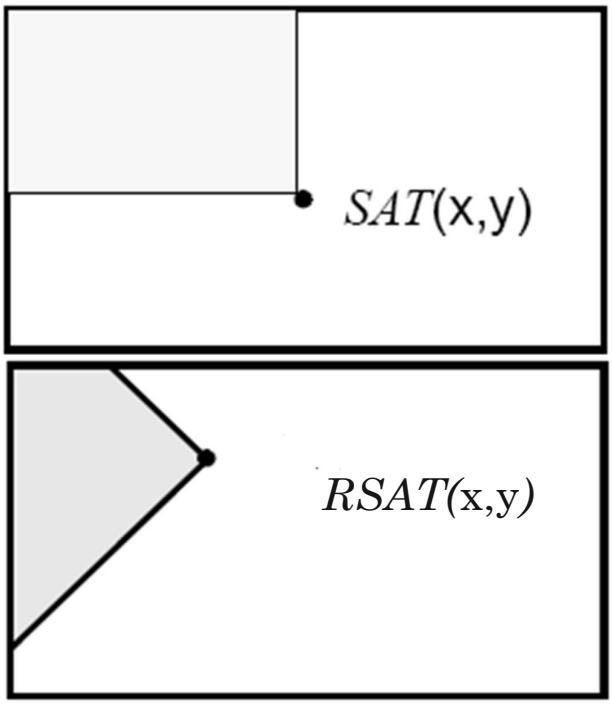
Ad esempio la somma all'interno del rettangolo D è data da:

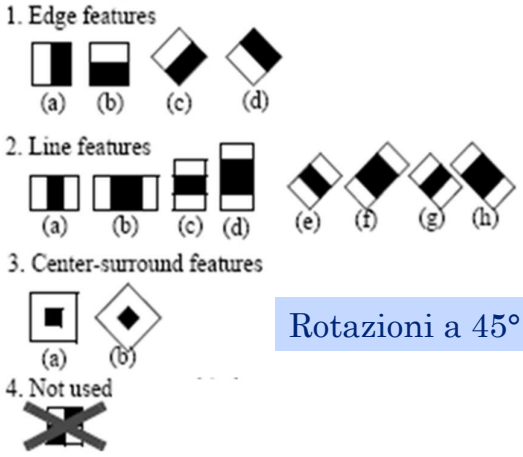
$$II(4) + II(1) - II(2) - II(3)$$

Le feature caratterizzate da 2, 3 o 4 rettangoli possono essere calcolate usando rispettivamente 6, 8 e 9 valori di riferimento.

Estensione delle feature (1)

set di feature introdotto da Lienhart e Maydt





Rotazioni a 45°

Vi sono ora due tipi di immagini integrali

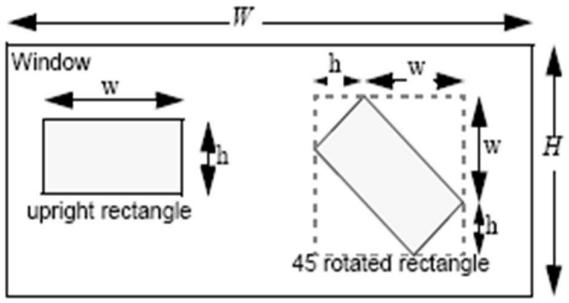
Sum Area Table

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

Rotated Sum Area Table

$$RSAT(x,y) = \sum_{x' \le x, x' \le x - |y-y'|} I(x',y')$$

Estensione delle feature (2)



Feature Type	w/h	X/Y	#
1a ; 1b	2/1 ; 1/2	12/24 ; 24/12	43,200
1c ; 1d	2/1 ; 1/2	8/8	8,464
2a ; 2c	3/1 ; 1/3	8/24 ; 24/8	27,600
2b ; 2d	4/1 ; 1/4	6/24 ; 24/6	20,736
2e ; 2g	3/1 ; 1/3	6/6	4,356
2f ; 2h	4/1 ; 1/4	4/4	3,600
3a	3/3	8/8	8,464
3b	3/3	3/3	1,521
Sum			117,941

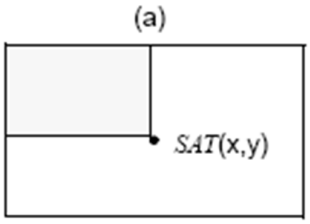
Dimensione della finestra di detection : $W \times H$

Fattori massimi di scala lungo x e y $X = \left\lfloor \frac{W}{w} \right\rfloor \quad Y = \left\lfloor \frac{H}{h} \right\rfloor$

•Il valore della caratteristica è calcolato come somma pesata di due componenti: la somma dei pixel corrispondenti alla regione scura e la somma di quelli all'interno della regione bianca.

•I segni dei due pesi sono opposti e i loro valori assoluti sono inversamente proporzionali alle aree delle rispettive regioni, in modo da effettuare un calcolo normalizzato.

Estensione delle feature (3)

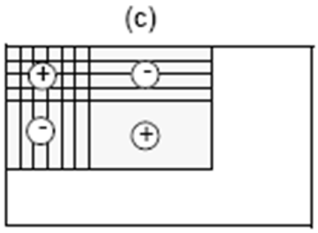


- **Calcolo di SAT in un solo passo**

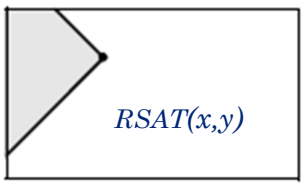
$SAT(-1,y) = SAT(x,-1)=0$
$SAT(x,y) = SAT(x,y-1) + SAT(x-1,y) + I(x,y) - SAT(x-1,y-1)$

- Per ogni rettangolo $r=(x,y,w,h,0)$ non ruotato la somma dei pixel può essere calcolata come:

$$RecSum(r) = SAT(x-1,y-1) + SAT(x+w-1,y+h-1) - SAT(x-1,y+h-1) - SAT(x+w-1,y-1)$$



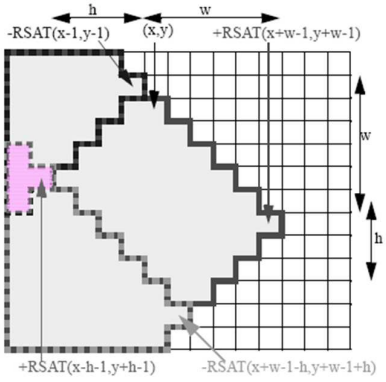
Estensione delle feature (4)



- **Calcolo di RSAT in due passi**
- Primo passo da sinistra a destra e dall'alto al basso

$RSAT(x,y) = RSAT(x-1,y-1) + RSAT(x-1,y) + I(x,y) - RSAT(x-2,y-1)$

con $RSAT(-1,y) = RSAT(-2,y) = RSAT(x,-1) = 0$

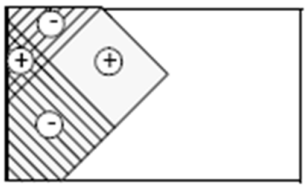


- Secondo passo da destra a sinistra e dal basso all'alto

$RSAT(x,y) = RSAT(x,y) + RSAT(x-1,y+1) - RSAT(x-2,y)$

- Per ogni rettangolo $r=(x,y,w,h,0)$ ruotato di 45° la somma dei pixel può essere calcolata come:

$$RecSum(r) = RSAT(x+w,y+w) + RSAT(x-h,y+h) - RSAT(x,y) - RSAT(x+w-h,y+w+h)$$



AdaBoost: il classificatore debole (1)

- Il più semplice classificatore debole è un albero di decisione con un singolo nodo.
- Supponiamo di aver costruito M-1 classificatori deboli $\{h_m(x)|m=1,..,M\text{-}1\}$ e di voler costruire $h_M(x)$. Il classificatore confronta il valore di una feature z_{k^*} con una soglia prefissata τ_{k^*} e assegna i valori +1 o -1 di conseguenza:

$$h_M(x) = +1 \quad \text{se} \quad z_{k^*} > \tau_{k^*}$$
$$= -1 \quad \text{altrimenti}$$

AdaBoost: il classificatore debole (2)

- In questa forma $h_M(x)$ dipende da due parametri:
  - la feature z_{k^*}
  - il valore di soglia τ_{k^*}
- I due parametri possono essere fissati in base al minimo errore pesato di classificazione:
  - Scegliamo il valore di soglia τ_k che minimizza l'errore di classificazione per ciascuna feature z_k;
  - La feature $\mathbf{z_{k^*}}$ scelta per il classificatore corrente è quella che permette di ottenere l'errore di classificazione complessivamente più basso.

AdaBoost: il classificatore robusto (1)

- AdaBoost apprende una sequenza di classificatori deboli h_m e li combina in un classificatore robusto H_M, minimizzando l'upper bound all'errore di classificazione di H_M. L'upper bound si ottiene con la seguente formula:

$$J(H_M) = \sum_i e^{-y_i H_M(x_i)} = \sum_i e^{-y_i \sum_{m=1}^{M} \alpha_m h_m(x_i)}$$

dove i è l'indice delle immagini di training.

- Dato il classificatore corrente $H_{M-1}(x) = \sum_{m=1}^{M-1} \alpha_m h_m(x)$ e il nuovo classificatore debole h_M, il miglior coefficiente α_M per il nuovo classificatore robusto $H_M(x) = H_{M-1}(x) + \alpha_M h_M(x)$ minimizza il costo:

$$\alpha_M = \arg\min_{\alpha} J(H_{M-1}(x) + \alpha_M h_M(x))$$

AdaBoost: il classificatore robusto (2)

- Il costo è minimo per:

$$\alpha_M = \log\frac{1-\varepsilon_M}{\varepsilon_M}$$

- dove ε_M è l'errore di classificazione pesato:

$$\varepsilon_M = \sum_i w_i^{M-1} 1 \cdot [sign(H_M(x_i)) \neq y_i]$$

- $1 \cdot [C]$ è 1 se C è vera, 0 altrimenti.
- A ogni iterazione è aggiornato il peso di ciascun pattern in base alle prestazioni di H_M:

$$w^{(M)}(x,y) = w^{(M-1)}(x,y)\exp(-y\alpha_M h_M(x)) = \exp(-yH_M(x))$$

AdaBoost: il classificatore robusto (3)

0. (*Input*)

(1) Pattern di training $\mathcal{Z} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, dove $N=a+b$;
a pattern hanno etichetta $y_i = +1$
b pattern hanno etichetta $y_i = -1$;

(2) Il numero M di classificatori da combinare.

1. (*Inizializzazione*)

$w_i^{(0)} = \frac{1}{2a}$ per i pattern con etichetta $y_i = +1$

$w_i^{(0)} = \frac{1}{2b}$ per i pattern con etichetta $y_i = -1$

2. (*Costruzione del classificatore*)

Per $m = 1,..,M$:

(1) Scegli il classificatore debole ottimale h_m che minimizzi l'errore pesato

(2) Scegli α_m

(3) Aggiorna i pesi $w_i^{(m)} \leftarrow w_i^{(m-1)} \exp[-y_i \alpha_m h_m(x_i)]$ e normalizzali $\left(\sum_i w_i^{(m)} = 1\right)$.

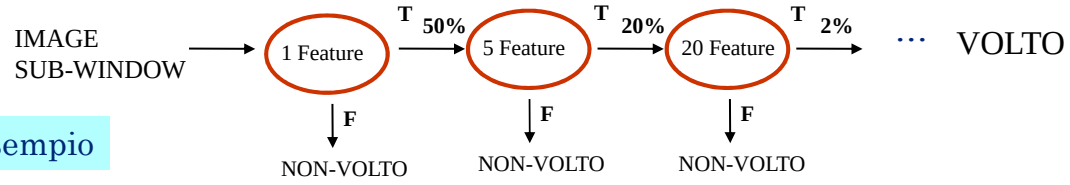
3. (*Output*)

(1) Classificatore $H_m(x)$

(2) Classificazione dei pattern di training $\hat{y} = \text{sign}[H_m(x)]$

Classificatori robusti in cascata

- Un solo classificatore robusto, per quanto elimini una grande porzione di sottofinestre che non contengono facce, non soddisfa i requisiti di applicazioni in termini d'efficienza e di percentuale molto bassa di falsi allarmi (ad esempio <= 10^{-6}). Una possibile soluzione consiste nell'impiego di classificatori in cascata, via via più complessi.



Esempio

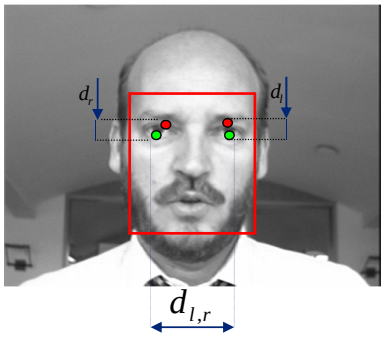
❑Un classificatore per una sola feature riesce a passare al secondo stadio la quasi totalità dei volti esistenti (circa 100%) mentre scarta al contempo il 50% dei falsi volti.
❑Un classificatore per 5 feature raggiunge quasi il 100% di detection rate e il 40% di false positive rate (20% cumulativo) usando i dati dello stadio precedente.
❑Un classificatore per 20 feature raggiunge quasi il 100% di detection rate con 10% di false positive rate (2% cumulativo).

Localizzazione dei volti

- La localizzazione dei volti avviene analizzando sottofinestre consecutive (sovrapposte) dell'immagine in input e valutando per ciascuna se appartiene alla classe dei volti.
- Il localizzatore di Viola e Jones è uno dei più robusti ed efficienti allo stato dell'arte
  - **L'addestramento è molto lento (può richiedere giorni);**
  - **La procedura di localizzazione è molto efficiente** (funzionamento real-time).

Valutazione delle prestazioni di localizzazione: indicatori

Correttezza della localizzazione



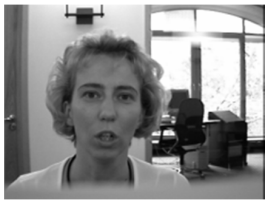
$$err = \max(d_l, d_r)/d_{lr}$$

Se ***err* < 0.25**, la localizzazione è considerata corretta

- Falsi positivi
  - Percentuale di finestre classificate come volto che in realtà non lo contengono.
- Facce non localizzate
  - Percentuale di volti che non sono stati individuati
- C-Error
  - Errore di localizzazione: distanza euclidea tra il reale centro della faccia e quello ipotizzato dal sistema, normalizzato rispetto alla somma degli assi dell'ellisse contenente il volto.

Database di test

- BioID face database
  http://www.bioid.com/downloads/facedb/index.php
- UCD Colour Face Image (UCFI) Database
  http://ee.ucd.ie/~prag/
- CMU face database
  http://vasc.ri.cmu.edu/idb/html/face/



- Face detection home page:
  http://www.facedetection.com/