UNIVERSITÀ CA' FOSCARI DI VENEZIA
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Specialistica in Informatica

Corso di Reti Neurali e Visione Artificiale

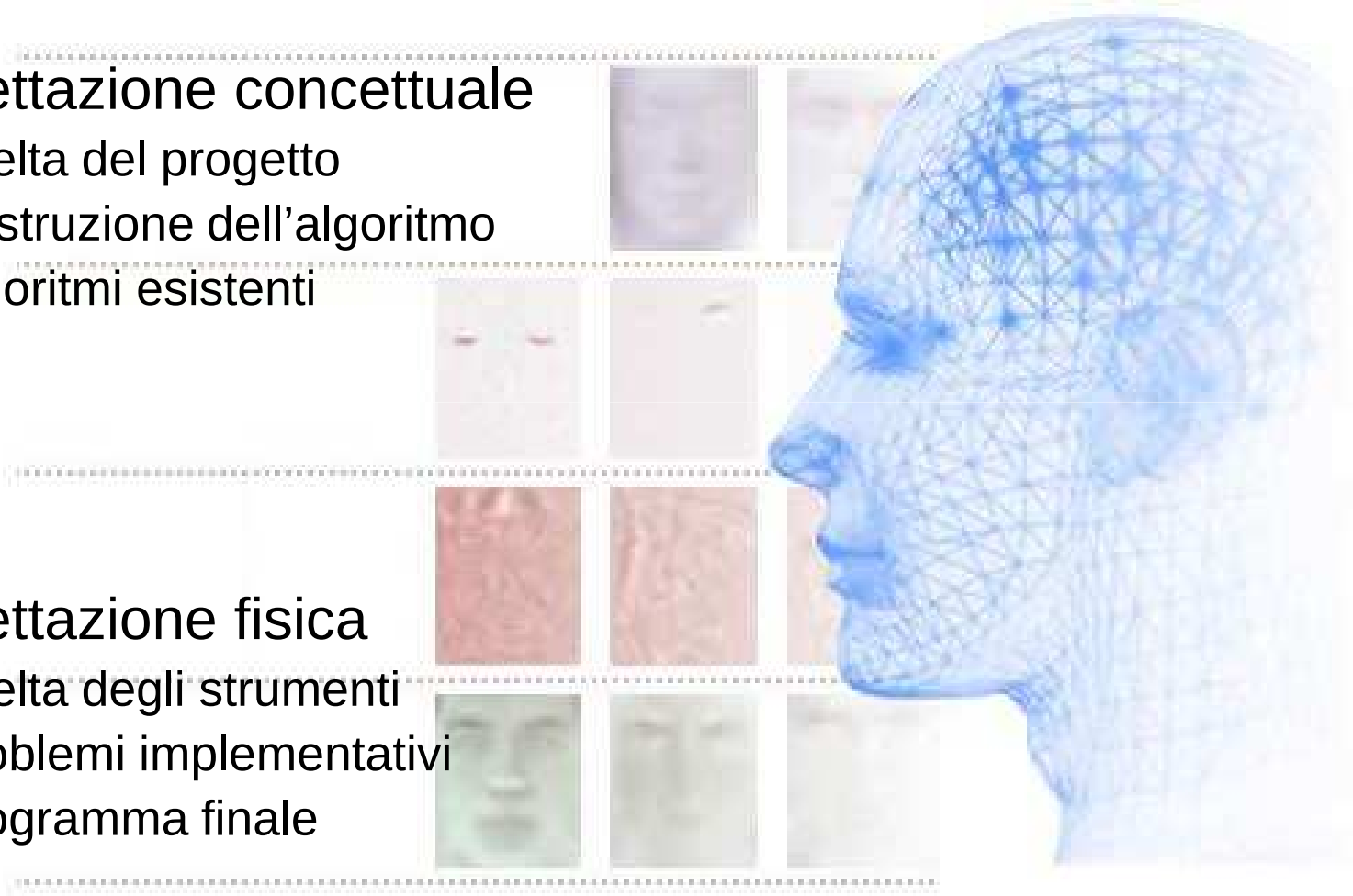
Computer Vision Project

Studente: Marco Lionello

Anno Accademico 2006-2007

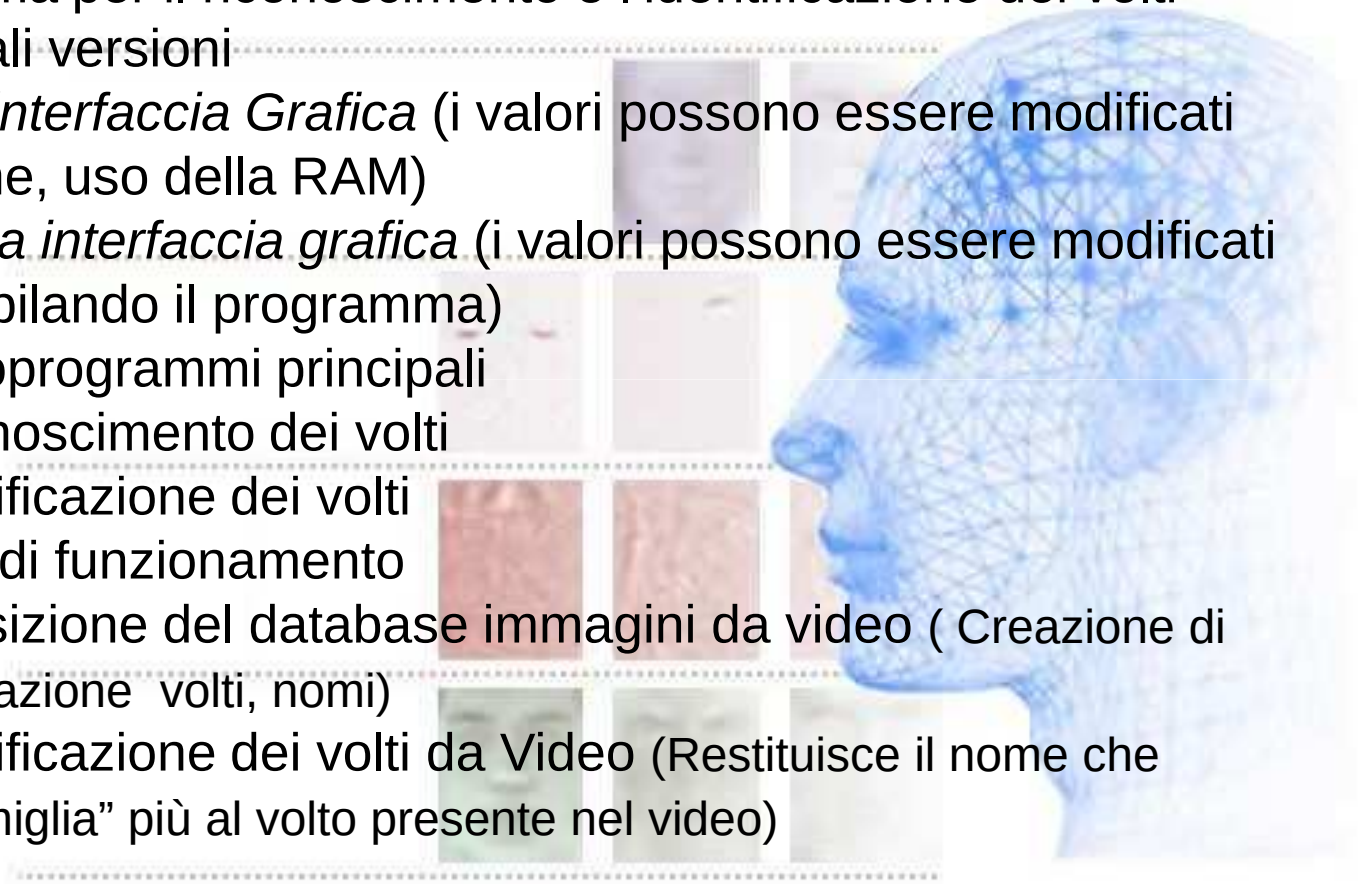
Introduzione

- Progettazione concettuale
  - Scelta del progetto
  - Costruzione dell'algoritmo
  - Algoritmi esistenti
- Progettazione fisica
  - Scelta degli strumenti
  - Problemi implementativi
  - Programma finale



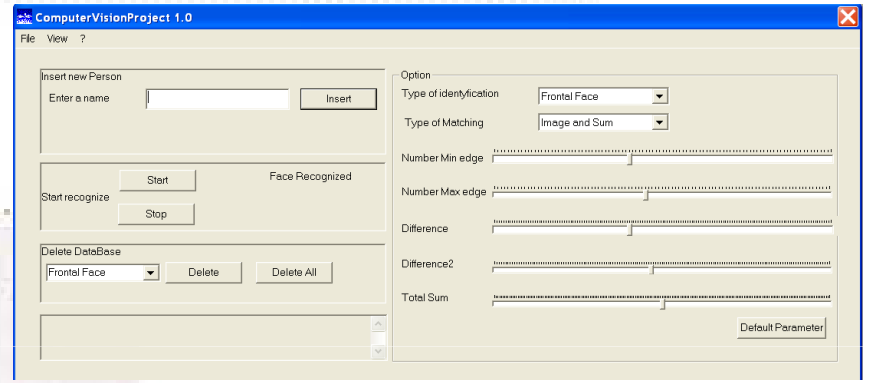
Computer Vision Project (CVP) (1/2)

- Programma per il riconoscimento e l'identificazione dei volti
- Due attuali versioni
  - *Con interfaccia Grafica* (i valori possono essere modificati runtime, uso della RAM)
  - *Senza interfaccia grafica* (i valori possono essere modificati ricompilando il programma)
- Due sottoprogrammi principali
  - Riconoscimento dei volti
  - Identificazione dei volti
- Principio di funzionamento
  - Aquisizione del database immagini da video (Creazione di associazione volti, nomi)
  - Identificazione dei volti da Video (Restituisce il nome che "assomiglia" più al volto presente nel video)



Computer Vision Project (CVP) (2/2)

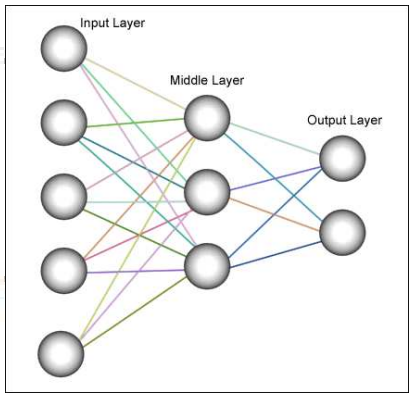
- Opzioni attualmente disponibili
  - Riconoscimento frontale
  - Riconoscimento laterale
  - Riconoscimento Mezzo busto
  - Riconoscimento a tutto busto
  - Miglior Immagine, Migliore insieme, Entrambi



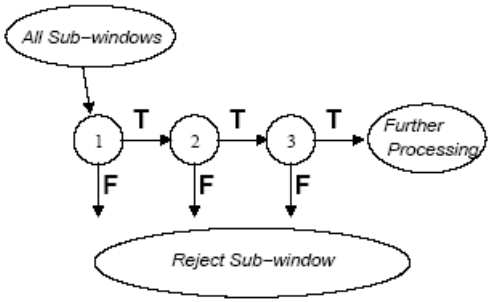
- Possibili utilizzi
  - Identificazione dell'utente
    - Per accesso a dati
    - Per accesso a sistemi
  - Motore di ricerca per volti
  - Altre applicazioni ….

Riconoscimento del volto

- Riconoscimento tramite ***reti neurali***
  - Riconoscimento estremamente lento
  - Addestramento della rete molto lento



- Riconoscimento tramite ***cascate di classificatori***
  - Estremamente rapido
  - Ottimo per il riconoscimento da video o da webcam che operano con molti frame per secondo
  - Addestramento relativamente lento



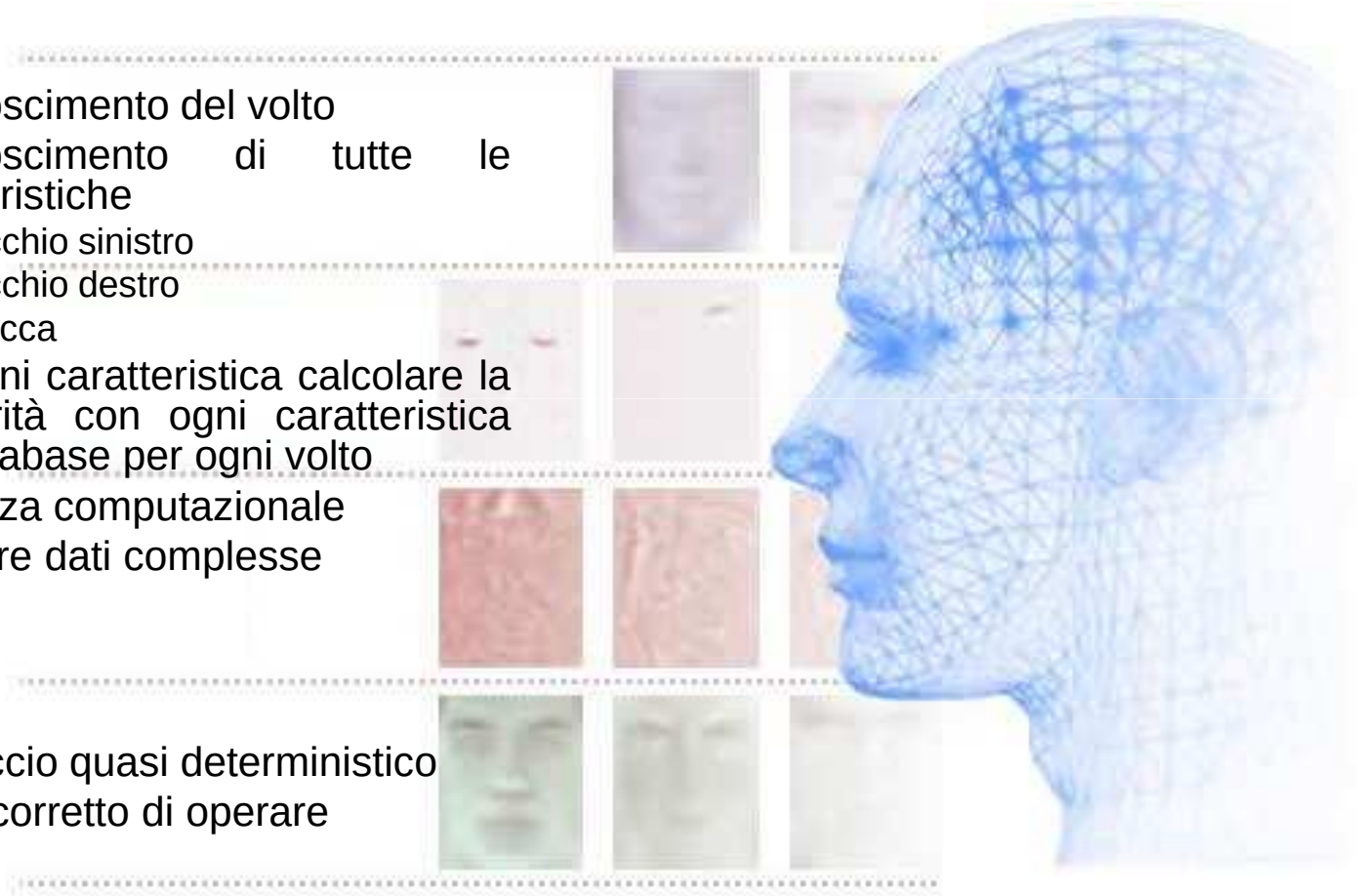
Identificazione dei volti (1/3)

Esempio tipico di identificazione:

- Estrazione di particolari es. naso, bocca, ecc. ecc.
- Estrazione delle caratteristiche:
  - Estrazione delle distanze e posizioni relative
  - Estrazione di informazioni colore occhi, colore capelli ecc. ecc.
- Archiviazione delle caratteristiche
- Ricerca del volto, selezionando il volto che minimizza la somma delle distanze tra le varie caratteristiche
- http://www.face-rec.org
  - Database
  - Algoritmi
  - New, Interesting paper

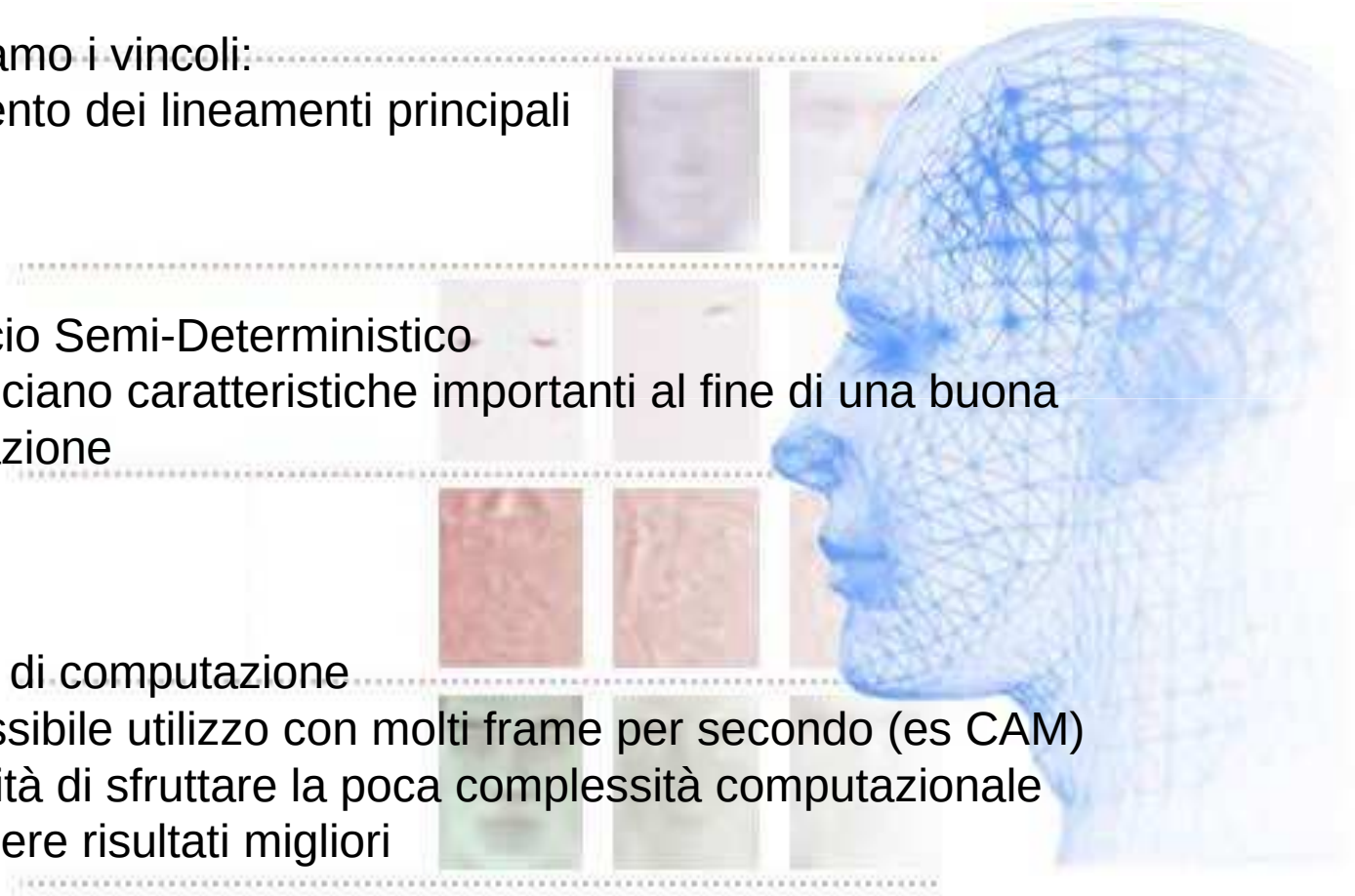
Identificazione dei volti (2/3)

- Svantaggi
  - Riconoscimento del volto
  - Riconoscimento di tutte le caratteristiche
    - Occhio sinistro
    - Occhio destro
    - Bocca
  - Per ogni caratteristica calcolare la similiarità con ogni caratteristica nel database per ogni volto
  - Lentezza computazionale
  - Strutture dati complesse
- Vantaggi
  - Approccio quasi deterministico
  - Modo corretto di operare



Identificazione dei volti (3/3)

Idea: Rilassiamo i vincoli:
•Riconoscimento dei lineamenti principali



Svantaggi

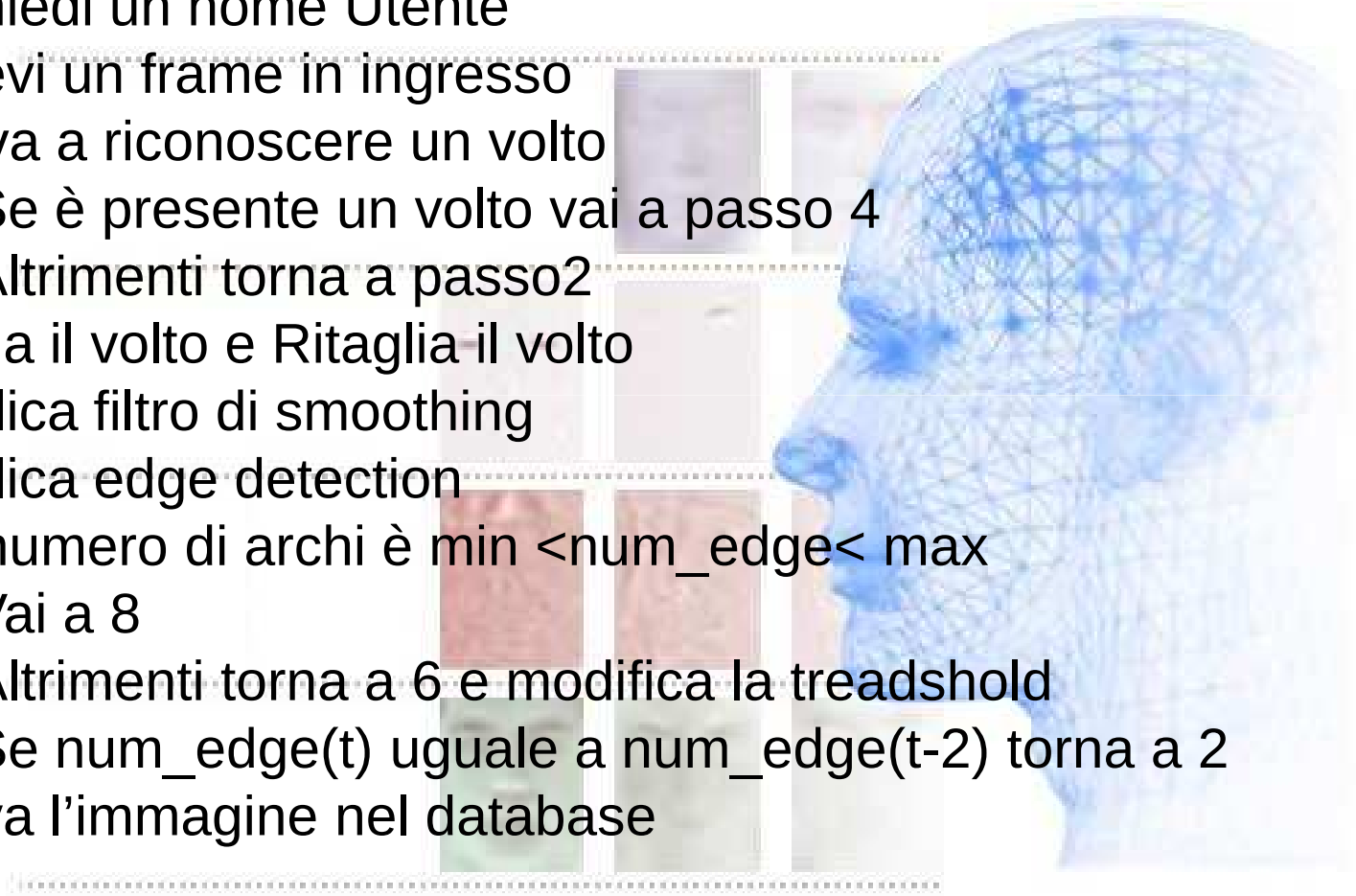
- •Approccio Semi-Deterministico
- •Si tralasciano caratteristiche importanti al fine di una buona identificazione

Vantaggi

- •Velocità di computazione
  - •Possibile utilizzo con molti frame per secondo (es CAM)
- •Possibilità di sfruttare la poca complessità computazionale per ottenere risultati migliori

Pseudo codice per il riconoscimento

1. Richiedi un nome Utente
2. Ricevi un frame in ingresso
3. Prova a riconoscere un volto
    1. Se è presente un volto vai a passo 4
    2. Altrimenti torna a passo2
4. Scala il volto e Ritaglia il volto
5. Applica filtro di smoothing
6. Applica edge detection
7. Se numero di archi è min <num_edge< max
    1. Vai a 8
    2. Altrimenti torna a 6 e modifica la treadshold
    3. Se num_edge(t) uguale a num_edge(t-2) torna a 2
8. Salva l'immagine nel database



Pseudo codice per l'identificazione

1. Ricevi un frame in ingresso
2. Prova a riconoscere un volto
   1. Se è presente un volto vai a passo 4
   2. Altrimenti torna a passo2
3. Scala il volto e Ritaglia il volto
4. Applica filtro di smoothing
5. Applica edge detection
6. Se numero di archi è min <num_edge< max
   1. Vai a 8
   2. Altrimenti torna a 6 e modifica la treadshold
   3. Se num_edge(t) uguale a num_edge(t-2) torna a 2
7. Per ogni immagine nella Base di Dati
   1. * Calcola le distanze tra le due immagini
   2. * Se la distanza è minore di una certa soglia hai trovato il volto corretto

Algoritmi utilizzati

Sottoproblemi principali:

- Riconoscimento del volto
- Edge detection
- Scelta della distanza tra immagini

Algoritmo per edge detection:

- Marr - Hildreth
- Canny
  1. Filtering
  2. Calcolo del gradiente
  3. NMS (non maxima suppression)
  4. Threadsholding

Perchè Canny?

- Tuttora è l'algoritmo più utilizzato nelle applicazioni di edge detection

- La fase di treadsholding risulta importantissima:
  - Permette la realizzazione dell'algoritmo
  - Permette di capire se sono presenti problemi di luminosità
  - Permette una semplificazione del calcolo della similitudine tra due immagini

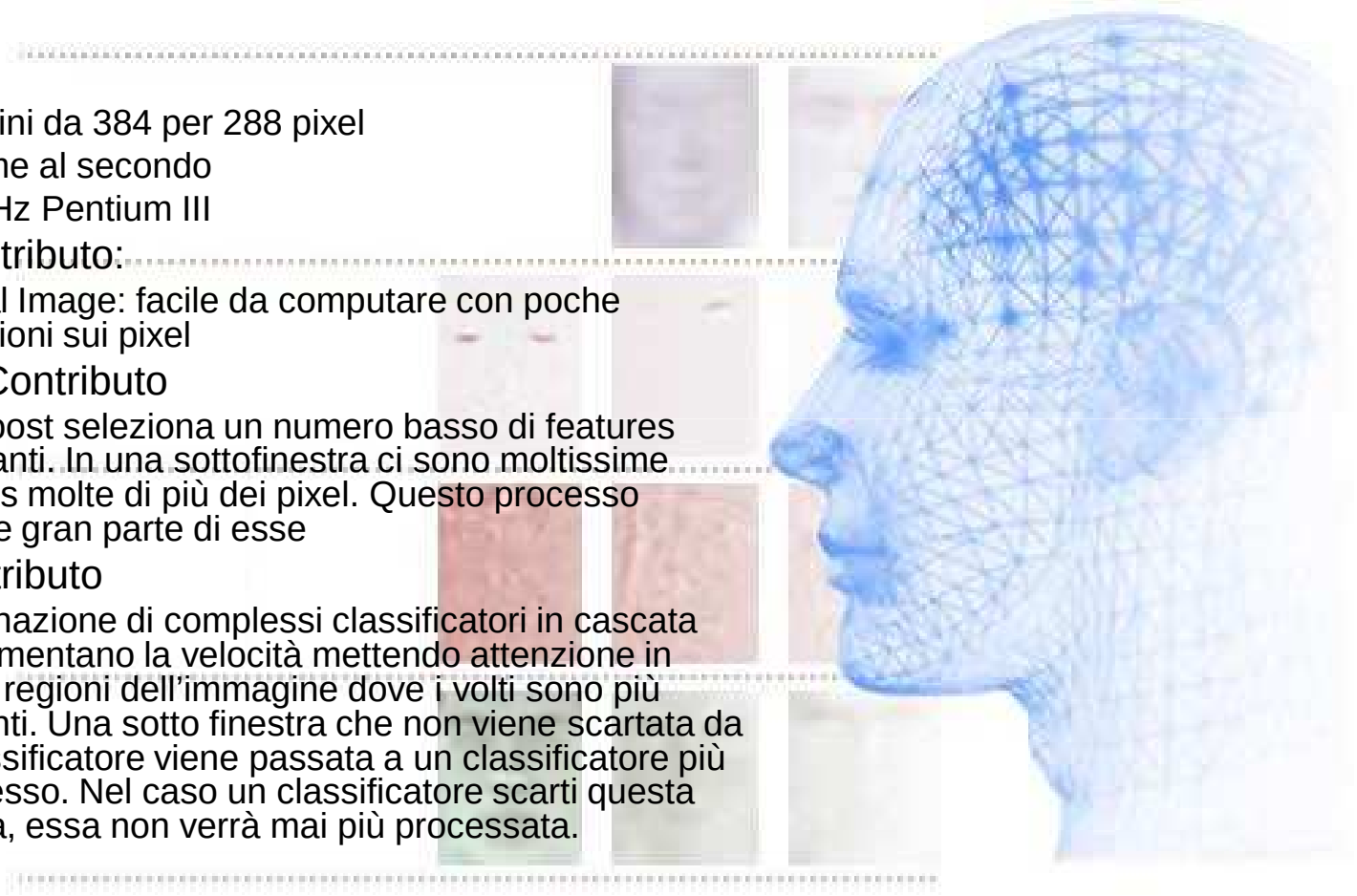
Riconoscimento dei volti

- Serve un algoritmo veloce in grado di computare velocemente le immagini da Video
  - **Rapid Object Detection using a Boosted Cascade of Simple Features.** *Paul Viola Mitsubishi Elettric Research Labs, Michael Jones Compaq CRT 2001*

- *Tre punti di forza:*
  - *Nuova rappresentazione delle immagini*
  - *Nuovo algoritmo di apprendimento "ada boost"*
  - *Metodo per combinare molti classificatori complessi in cascata*

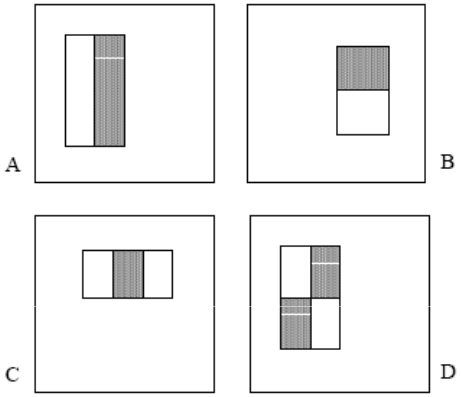
Rapid Object Detection: Introduzione

- Velocità
  - Immagini da 384 per 288 pixel
  - 15 frame al secondo
  - 700 MHz Pentium III
- Primo Contributo:
  - Integral Image: facile da computare con poche operazioni sui pixel
- Secondo Contributo
  - Ada Boost seleziona un numero basso di features importanti. In una sottofinestra ci sono moltissime features molte di più dei pixel. Questo processo esclude gran parte di esse
- Terzo contributo
  - Combinazione di complessi classificatori in cascata che aumentano la velocità mettendo attenzione in alcune regioni dell'immagine dove i volti sono più frequenti. Una sotto finestra che non viene scartata da un classificatore viene passata a un classificatore più complesso. Nel caso un classificatore scarti questa finestra, essa non verrà mai più processata.



Rapid Object Detection: Features

- Features
  - Possono essere usate come codifica ad-hoc di domini di conoscenza
  - Sono estremamente rapide rispetto all'uso dei pixel
- Tre tipi di features
  - *Two rettangle features* il valore è dato dalla differenza tra la somma dei pixel entro le due regioni rettangolari
  - *Tree rettangle features* il valore è dato dalla somma dei due esterni meno quello interno
  - *Four rettangle feature* differenza tra le diagonali
- Data una risoluzione di 24*24 pixel l'insieme dei rettangoli è circa 180.000



Rapid Object Detection: Integral Image
(1/2)

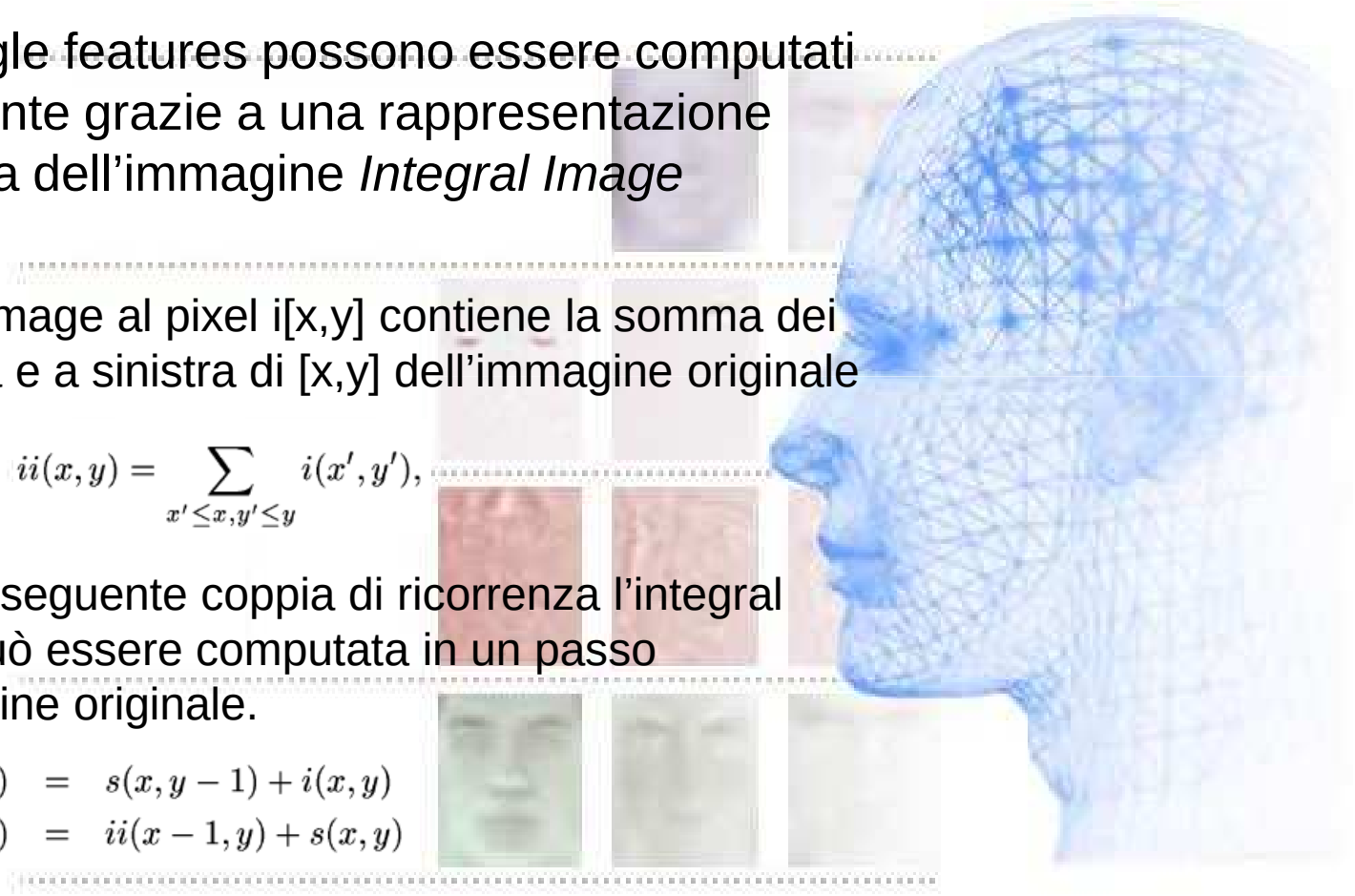
- Le rettangle features possono essere computati velocemente grazie a una rappresentazione intermedia dell'immagine *Integral Image*

- L'integral image al pixel i[x,y] contiene la somma dei pixel sopra e a sinistra di [x,y] dell'immagine originale inclusi

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

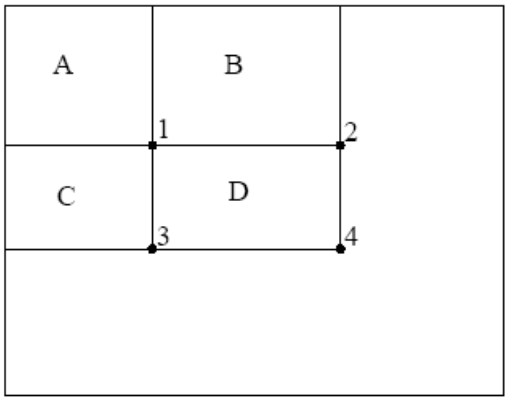
- Usando la seguente coppia di ricorrenza l'integral immage può essere computata in un passo dall'immagine originale.

$$\begin{aligned} s(x,y) &= s(x,y-1) + i(x,y) \\ ii(x,y) &= ii(x-1,y) + s(x,y) \end{aligned}$$



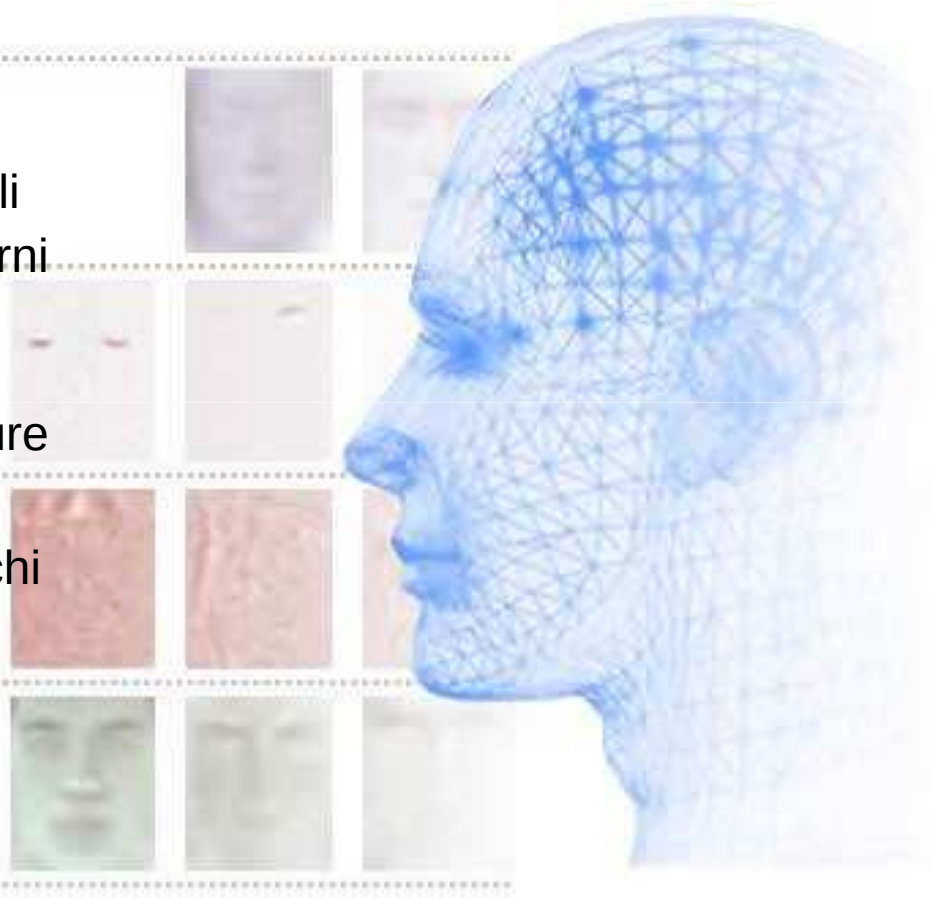
Rapid Object Detection: Integral Image (2/2)

- Usando l'integral image ogni rettangolo può essere computato in 4 riferimenti ad array
- Chiaramente la differenza tra la somma di due rettangoli può essere computata in 8 riferimenti ad array
- Quindi nel caso del two-rettangle siccome i rettangoli sono adiacenti servono solo 6 accessi ad array, 8 per il caso dei tree-rettangle e 9 pril four-rettangle



Discussione sulle Feauture

- Rettangoli sono primitivi
- Feauture con Filtri orientabili
  - Ottimi per l'analisi dei dettagli
  - Ottimi per l'analisi dei contorni
  - Ottimi per la compressione delle immagini
  - Ottimi per l'analisi delle texture
- Feauture Rettangolari
  - Sensibili alla presenza di archi
  - Sensibili a ogni struttura semplice (es barre)
  - Possibilità di usarli con l'integral image
  - Molto flessibili rispetto ai precedenti



Learning Classification function (1/2)

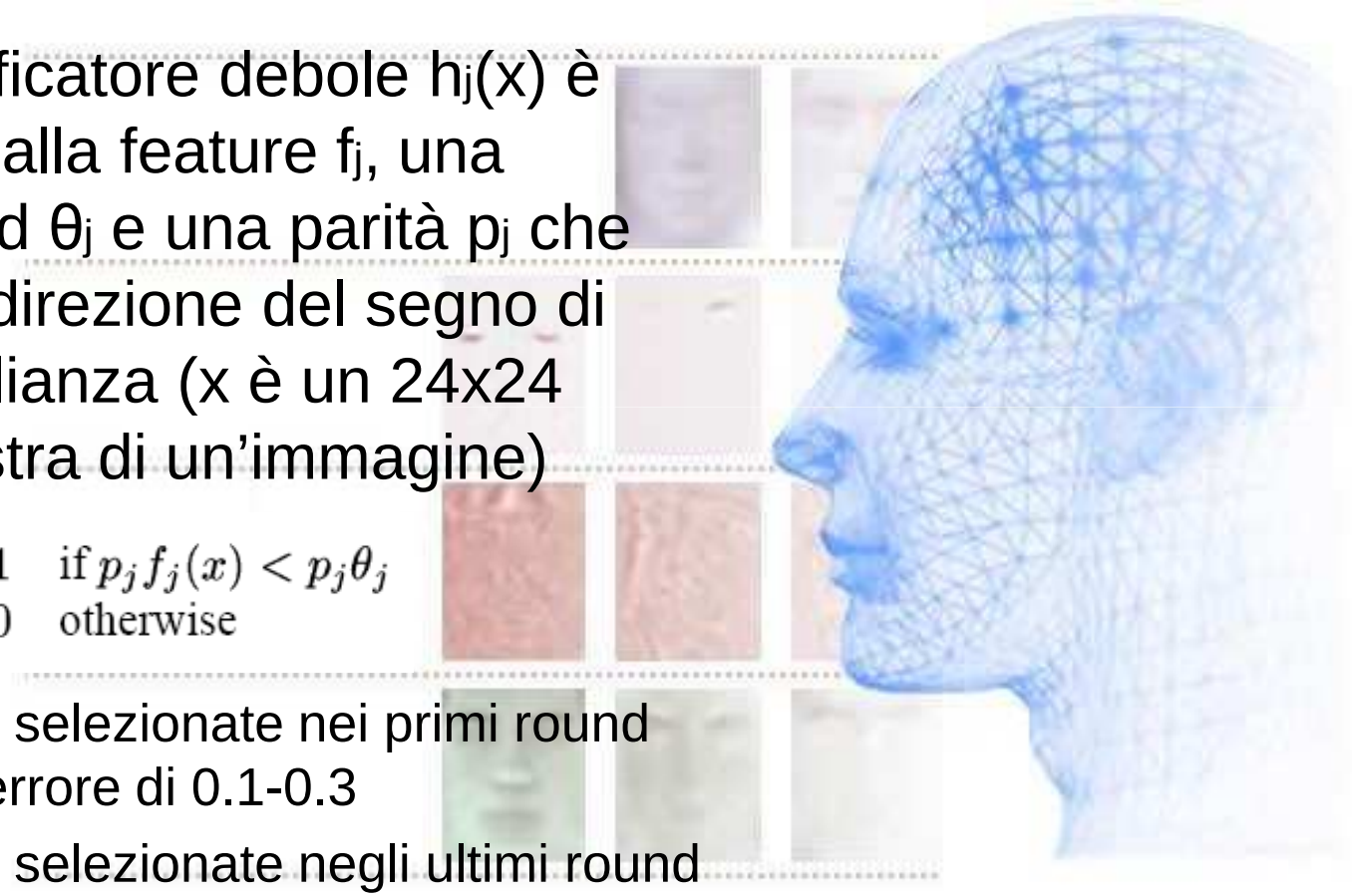
- Data un'insieme di feauture e un insieme di immagini positive e negative di training, le "Machine Learning" imparano una funzione di classificazione.
- Variante dell'AdaBoost
- Nella forma finale (Classificatore forte)
  - Amplifica le performance di classificazione
  - L'errore di training tende a zero esponenzialmente al numero di giri
  - Solo una parte delle 180.000 feauture viene effettivamente usato
- Nella forma debole
  - Si prende in considerazione il rettangolo che meglio separa i positivi e i negativi (Classificatore debole)
  - Per ogni classificatore debole determina la migliore soglia

Learning Classification function (2/2)

- Un classificatore debole $h_j(x)$ è formato dalla feature f_j, una thredshold θ_j e una parità p_j che indica la direzione del segno di diseguaglianza (x è un 24x24 sottofinestra di un'immagine)

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

  - Feature selezionate nei primi round hanno errore di 0.1-0.3
  - Feature selezionate negli ultimi round hanno errore di 0.4-0.5



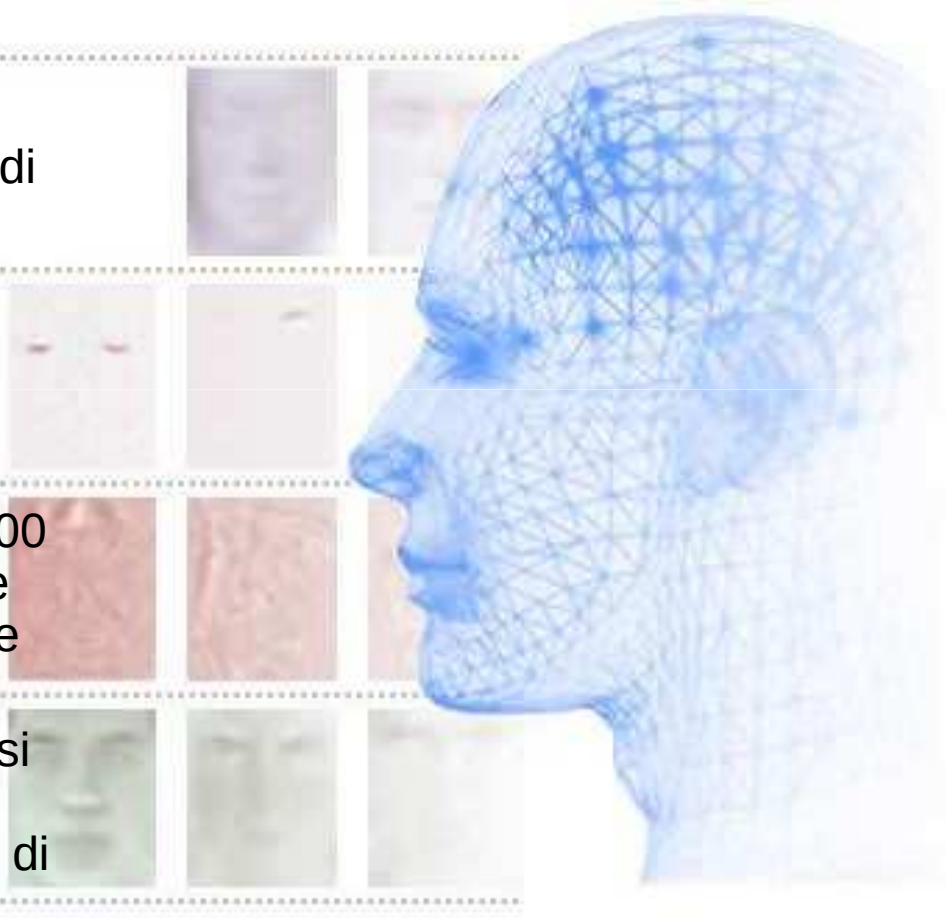
Algoritmo AdaBoost

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Discussione sull'apprendimento (1/2)

- Approccio finale molto aggressivo
  - La stragrande maggioranza di features viene eliminata
  - Buoni risultati con sole 37 features in un totale di 1734

- Alcuni risultati
  - Classificatore facciale con 200 feature con un detection rate del 95% con un false positive rate di 1 in 14084
  - Più veloce rispetto a qualsiasi altro sistema pubblicato 0.7secondi per un'immagine di 384x288

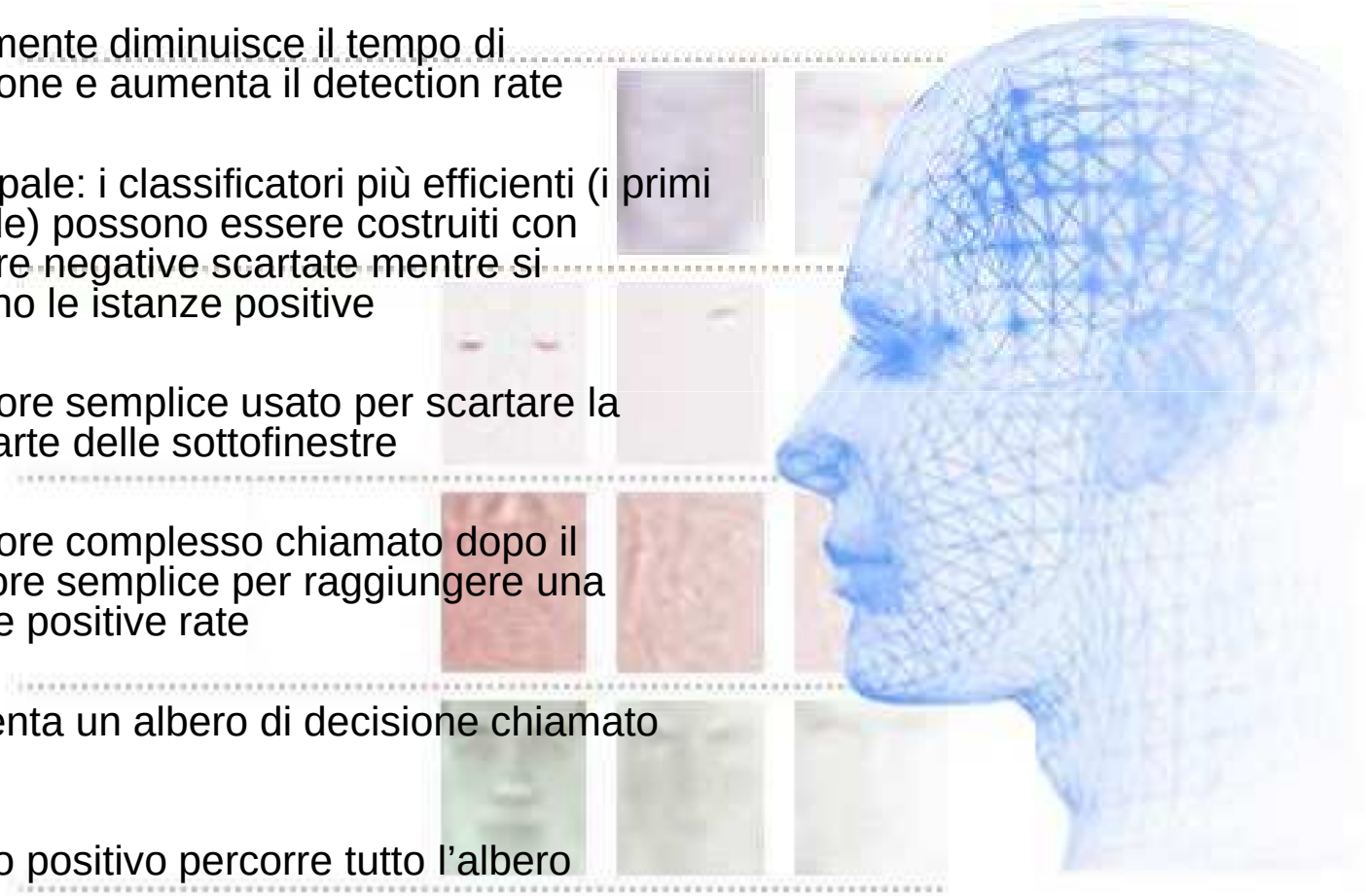


Discussione sull'apprendimento (2/2)

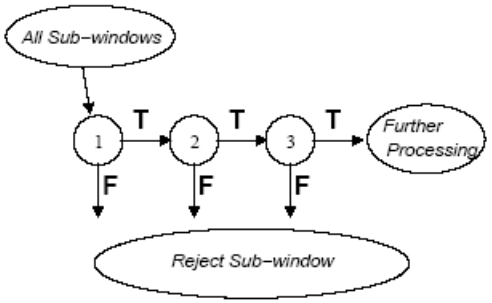
- Feature autoapprese da AdaBoost ricche di significato e facilmente interpretabili
  - Prima focalizzazione occhi solitamente scuri, guance solitamente chiare
  - Seconda focalizzazione occhi scuri, ponte del naso solitamente chiaro

Cascade of Classifier (1/2)

- Essenzialmente diminuisce il tempo di computazione e aumenta il detection rate
- Idea principale: i classificatori più efficienti (i primi nel cascade) possono essere costruiti con sottofinestre negative scartate mentre si determinano le istanze positive
- Classificatore semplice usato per scartare la maggior parte delle sottofinestre
- Classificatore complesso chiamato dopo il classificatore semplice per raggiungere una bassa false positive rate
- Il tutto diventa un albero di decisione chiamato ”cascade”
- Un risultato positivo percorre tutto l’albero

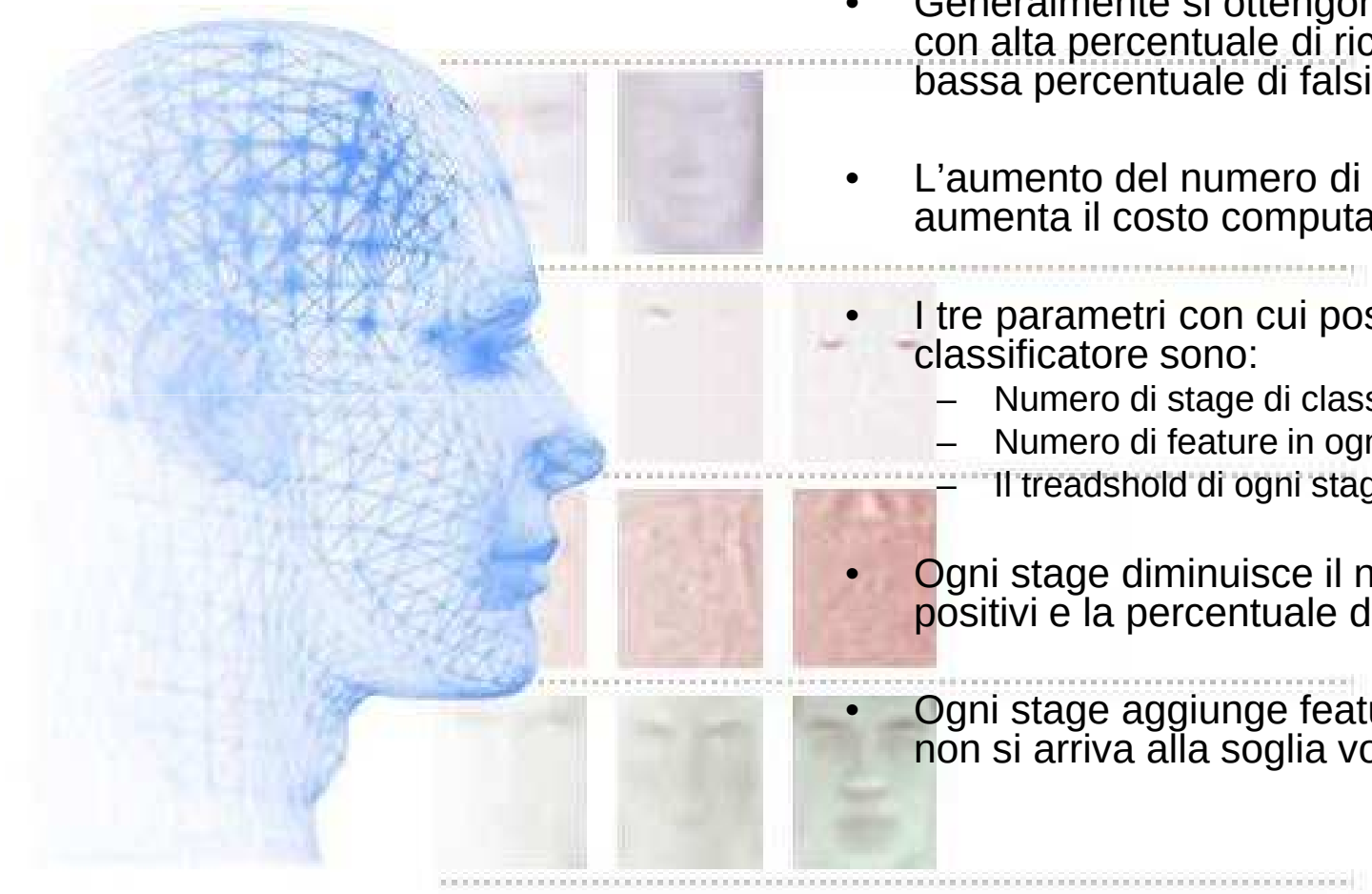


Cascade of Classifier (2/2)



- Stage creati grazie Ada Boost
  - Default AdaBoost designato per ottenere basse percentuali di errore nel training data (alto riconoscimento, alti falsi positivi)
- Modifica
  - I primi stage eliminano la maggior parte delle sottofinestre negative. Una volta eliminate non vengono mai più computate. Gli stage successivi ne eliminano via via sempre di meno aumentando la computazione.
  - Costo computazionale di 60 istruzioni di microprocessore per stage

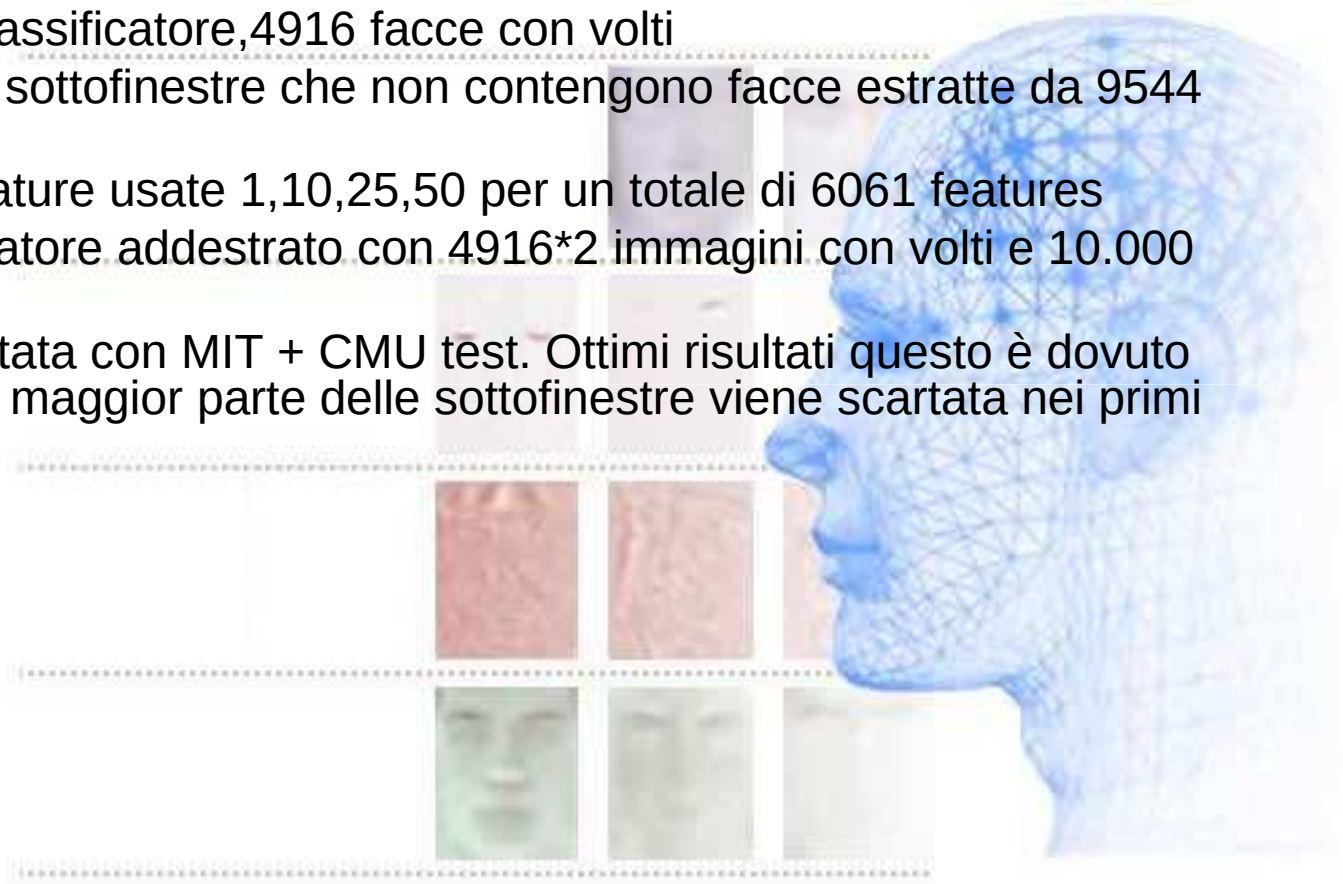
Addestrare una cascata di classificatori



- Generalmente si ottengono classificatori con alta percentuale di riconoscimento e bassa percentuale di falsi positivi
- L'aumento del numero di feauture aumenta il costo computazionale
- I tre parametri con cui possiamo creare il classificatore sono:
  - Numero di stage di classificatori
  - Numero di feature in ogni stage
  - Il treadshold di ogni stage
- Ogni stage diminuisce il numero di falsi positivi e la percentuale di riconoscimento
- Ogni stage aggiunge feature fino a che non si arriva alla soglia voluta

Detector finale

- 38 stage di classificatore,4916 facce con volti
- 350 milioni di sottofinestre che non contengono facce estratte da 9544 immagini
- Numero di feature usate 1,10,25,50 per un totale di 6061 features
- Ogni classificatore addestrato con 4916*2 immagini con volti e 10.000 senza volti
- ***Velocità*** valutata con MIT + CMU test. Ottimi risultati questo è dovuto al fatto che la maggior parte delle sottofinestre viene scartata nei primi stage



Paragone con altri algoritmi

- 15 volte più veloce dell'algoritmo di *Rowley* (usa una rete per fare uno prescreen per trovare le regioni di interesse e in queste fa partire una rete più accurata)
- *Amit and Geman* Usa il principio delle co-occorenze. Alcune semplici feature che se presenti fanno partire un classificatore forte in base alla occorrenza
- *Fleuret and Geman* "Catene di Test" fatte da disgiunzioni di archi a basse scale. Approccio diverso non interpretabile e tiene conto della "density evaluation e della density discrimination". Genera percentuali alte di falsi negativi

- Esempio di confronto:
  - 38 stages con 6000 feature
  - Dataset con 507 facce 75 milioni di sottofinestre

Detector \ False detections	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2 %	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

Table 2: Detection rates for various numbers of false positives on the MIT+CMU test set containing 130 images and 507 faces.

Funzione di edge matching

Una funzione di edge match interessante è decritta nel paper:

•M. Hu. ***Visual Pattern Recognition by Moment Invariants***, IRE Transactions on Information Theory, 8:2, pp. 179-187, 1962.

– Essa misura la similiarità tra due contorni A,B
– Il momento centrale normalizzato è:
– Hu ha dimostrato che l'insieme delle sette caratteristiche derivate dal secondo e terzo momento di un contorno sono restano invariati rispetto alla traslazione, rotazione e cambio di scala

$$A = \{(x_i, y_i), 1 \le i \le n\} \quad B = \{(u_i, v_i), 1 \le i \le m\}$$

$$\eta_{pq}, 0 \le p+q \le 3$$

$$h_1 = \eta_{20} + \eta_{02},$$

$$h_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,$$

$$h_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2,$$

$$h_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2,$$

$$h_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2],$$

$$h_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}),$$

$$h_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + -(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].$$

Funzione di edge matching

- Le tre funzioni di match che si possono calcolare sono quindi:

$$I_1(A,B) = \sum_{i=1}^{7} \left| -1/m_i^A + 1/m_i^B \right|,$$

$$I_2(A,B) = \sum_{i=1}^{7} \left| -m_i^A + m_i^B \right|,$$

$$I_3(A,B) = \max_i \left| (m_i^A - m_i^B)/m_i^A \right| ,$$

con $m_i^A = \mathrm{sgn}(h_i^A)\log_{10}\left|h_i^A\right|$, $m_i^B = \mathrm{sgn}(h_i^B)\log_{10}\left|h_i^B\right|$

- Per quanto riguarda l'uso in questo tipo di applicazione non dà buoni risultati (i risultati migliori si sono ottenuti con una semplice funzione di match)

Per quanto Riguarda la funzione di match usata in questa applicazione

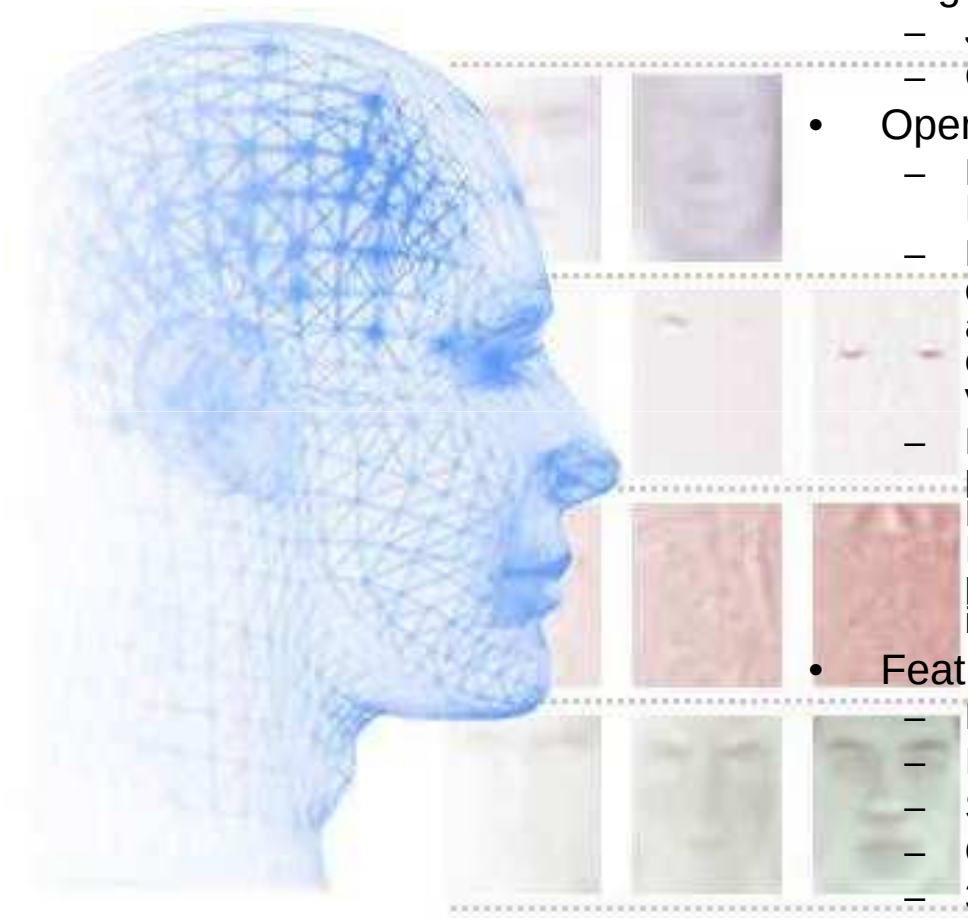
- Semplice
- Computazionalmente veloce

- Prendiamo la funzione che ci da 0 quando i due bit sono uguali, 1 se sono diversi!!!!!
- La calcoliamo come modulo della differenza tra i due array

Versioni Fallimentari

- Calcolo della similiarità tra volti grazie a funzioni di distanza tra immagini
  - Le immagini diventano complesse da computare(non più in bianco e nero)
  - Se si trasformano(es. bianco e nero) le immagini perdono informazioni importanti
  - EMD non tiene conto della località dell'informazione
- Uso del generatore dei classificatori per classificare un singolo volto
  - Addestrare il classificatore in maniera da riconoscere solo un volto
  - Per un buon addestramento servono migliaia di volti
    - Modifica, shifting, rotazione
  - Quale treadshold?
    - Troppo basso → molti falsi positivi
    - Troppo alto → tempo di esecuzione improponibile

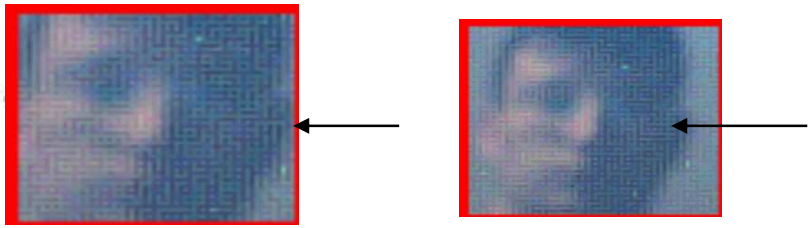
Scelte degli strumenti



- Linguaggio
  - Java
  - C++
- OpenCV (Computer Vision Library)
  - Libreria OpenSource sotto il controllo di Intel
  - È un'opportunità di stabilire una comunità che faccia un miglior uso degli "ultimi" algoritmi di visione allo scopo di far crescere lo sviluppo nell'ambiente di visione nei PC
  - Le funzioni sono ottimizzate per i processori intel usando IPP (Integrated Performance Primitives) a basso livello. IPP è multi piattaforma ed è particolarmente adatto nelle operazioni di image processing
- Features di OpenCV
  - Motion Analysis and Object Tracking
  - Image Analysis
  - Structural Analysis
  - Object Recognition
  - 3D Reconstruction
  - Basic Structures and Operations

Processo delle immagini

- Scala,dimensione e posizione

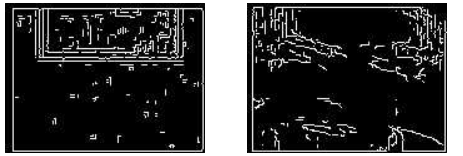


- Normalizzazzione della dimensione

- Applicazione del filtro e trasformazione in scala di grigi

Processo delle immagini

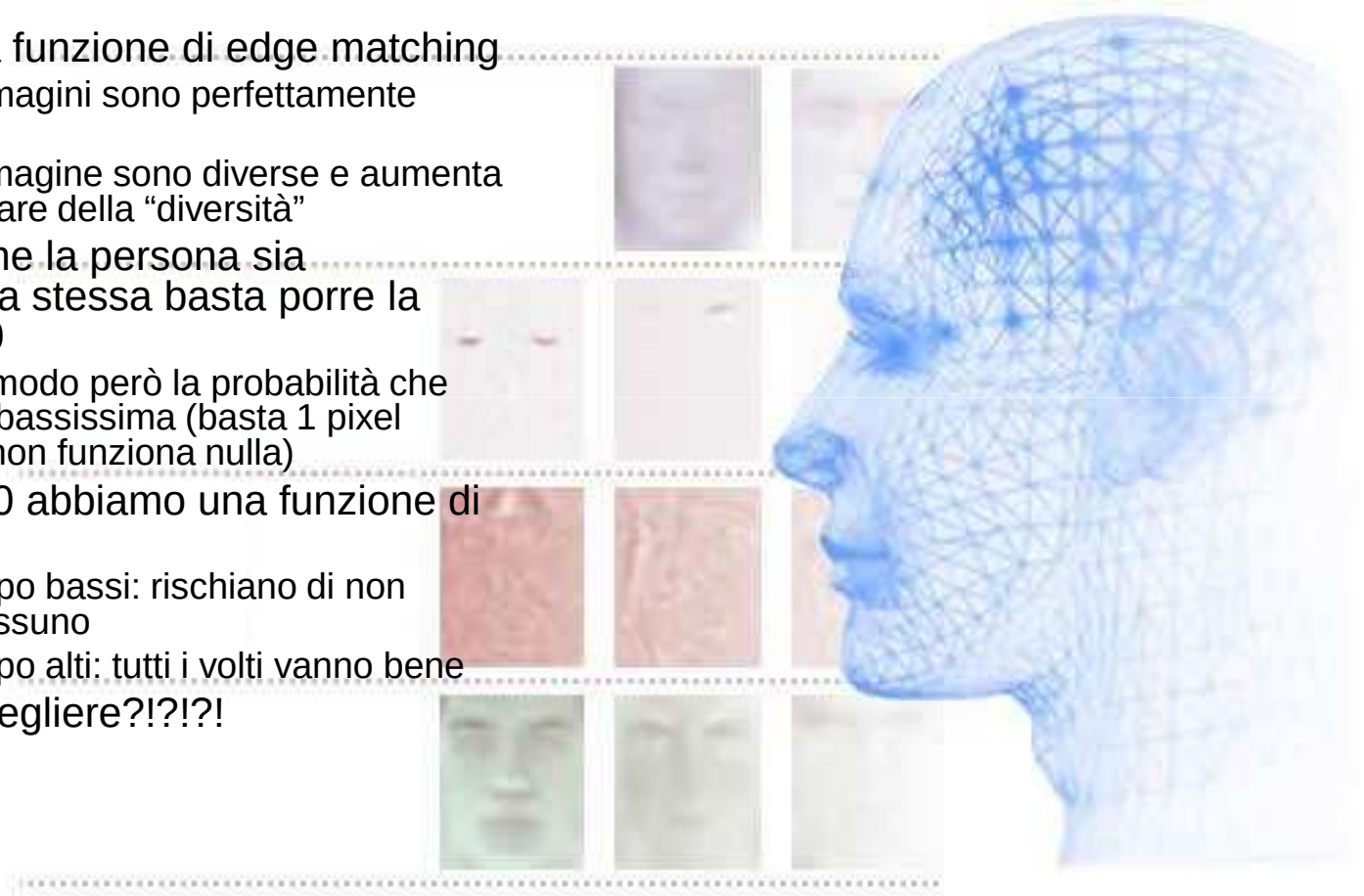
- Edge detection problemi:
  - Luminosità
  - Numero di archi



- Edge Detection Normalizzato:
  - Elimina problemi luminostità
  - Numero di archi limitato

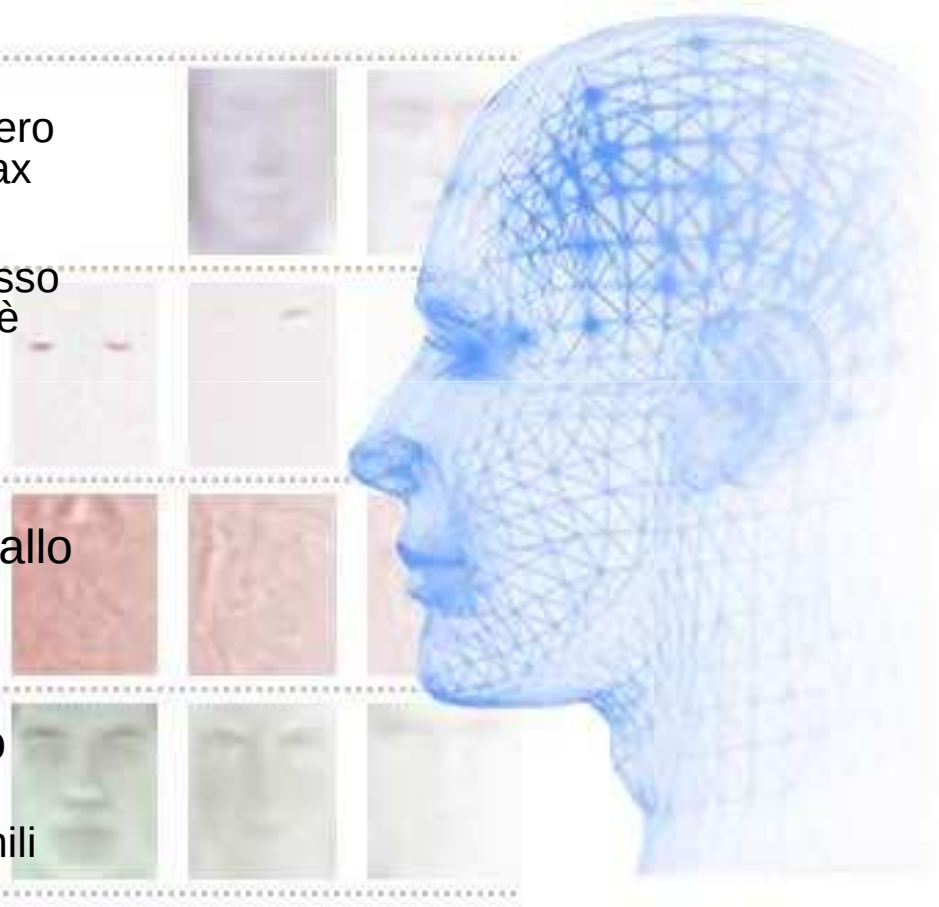
Agire sui parametri della funzione di matching

- Ricordiamo la funzione di edge matching
  - 0 se le immagini sono perfettamente uguali
  - >0 se l'immagine sono diverse e aumenta all'aumentare della "diversità"
- Imponendo che la persona sia esattamente la stessa basta porre la condizione =0
  - In questo modo però la probabilità che funzioni è bassissima (basta 1 pixel diverso e non funziona nulla)
- Imponendo >0 abbiamo una funzione di similiarità
  - Valori troppo bassi: rischiano di non trovare nessuno
  - Valori troppo alti: tutti i volti vanno bene
- Che soglia scegliere?!?!?!



Agire sui parametri dell'immagine

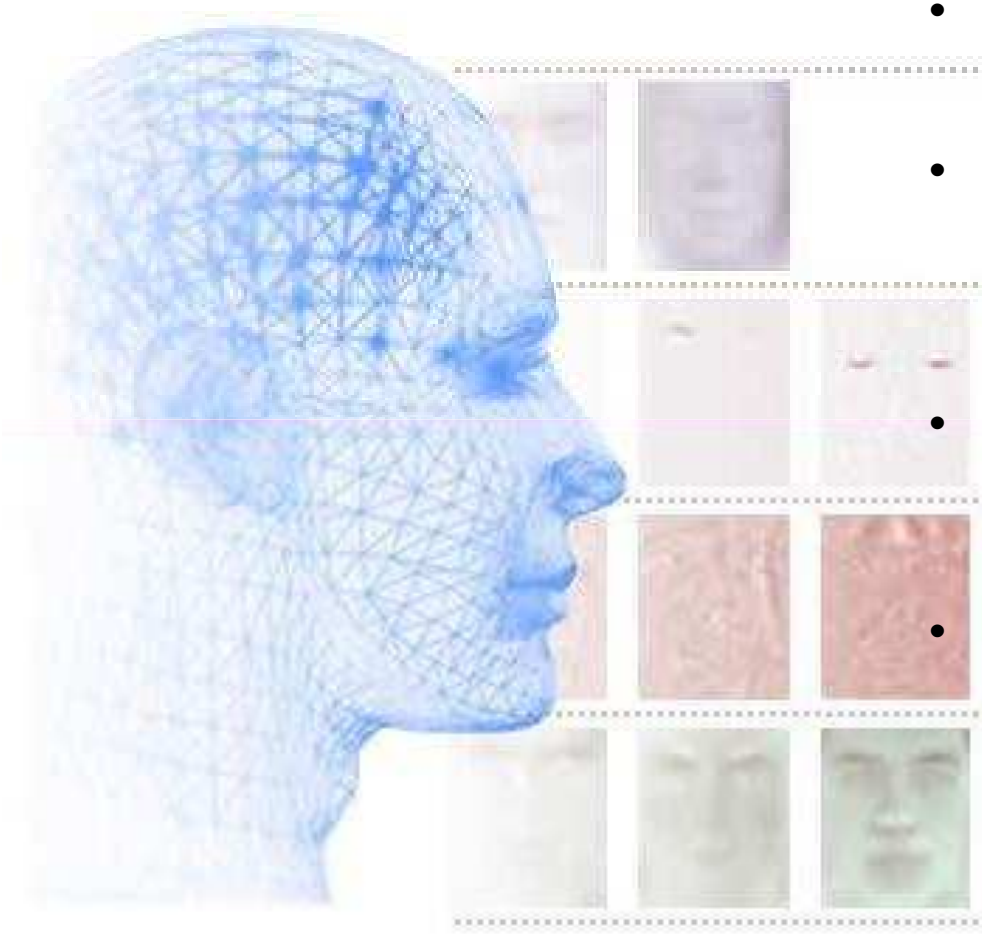
- Introduciamo il range
  - Quando prendiamo un nuovo volto esso deve avere un numero di archi compreso tra Min e Max
- Se per esempio Min=Max
  - Tutte le immagini hanno lo stesso numero di archi ma ogni volto è diverso
  - Non è realizzabile non si può arrivare ad un numero di archi preciso
- Se Max diverso da Min e intervallo grande:
  - Le immagini sono molto simili (tante collisioni tra pixel)
- Se Max diverso Min e intervallo piccolo:
  - Le immagini sono molto dissimili (poche collisioni tra pixel)



Ricapitolando

- Evitiamo queste situazioni (la funzione non avrebbe senso!!!!)
- Tutte le immagini saranno del tipo
- In pratica andiamo a calcolare una funzione di similiarità banale ma su immagini altamente dissimili

Sfruttiamo la velocità



- L’algoritmo è leggero e veloce
  - Usiamo la probabilità
- E’ possibile che un volto sia simile a un altro (es. posizione della testa nel momento del riconoscimento)
- Il detector scala. Una scala di un volto può essere simile alla scala di un altro diverso
- Ma cosa succede se nel database invece che una immagine ne salvo più di una e scelgo il volto che assomiglia più all’insieme?
  - La probabilità che scelga il volto sbagliato diminuisce!!

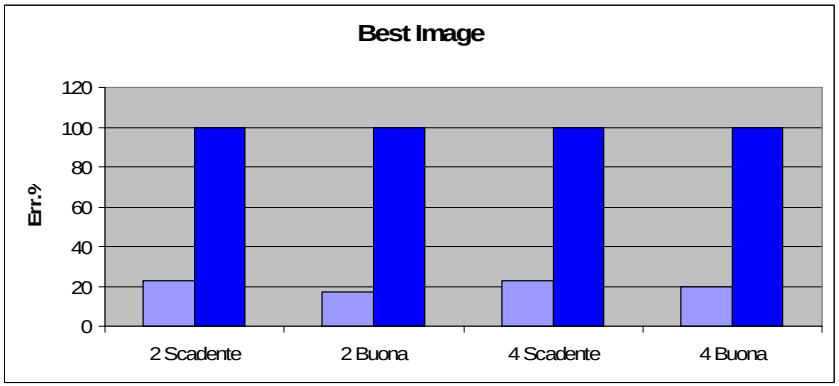
Ottimizzazione finale

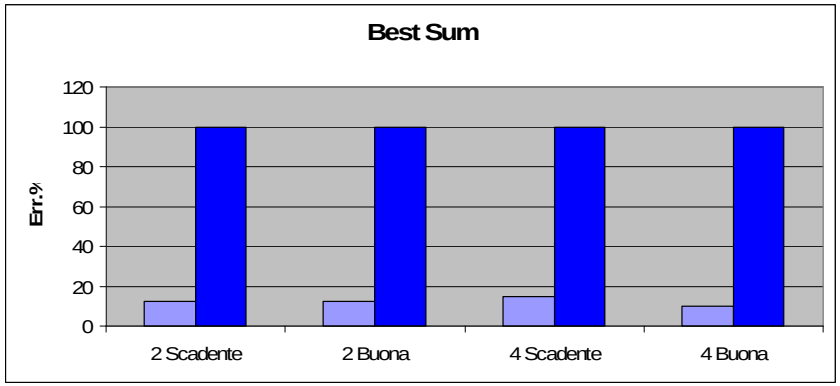
- Scegliamo il volto se:
  - L’insieme dei volti è più simile al volto da identificare (somma delle differenze tra tutte le immagini dell’insieme)
  - Se nel suo insieme esiste l’immagine che minimizza la funzione di matching rispetto a tutto il database

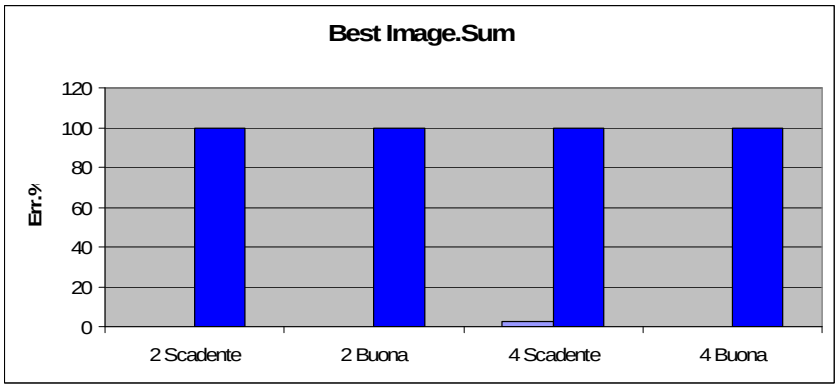
Performance (1/3)

Num Pers	Luminosità	Tipo Sel.	Prove	Prove Tot	Errore	Tempo Medio	Err. Perc.
2	Scadente	Best Image	20X2	40	9	29	22,5
2	Buona	Best Image	20X2	40	7	11	17,5
4	Scadente	Best Image	10X4	40	9	30	22,5
4	Buona	Best Image	10X4	40	8	10	20
2	Scadente	Best Sum	20X2	40	5	38	12,5
2	Buona	Best Sum	20X2	40	5	13	12,5
4	Scadente	Best Sum	10X4	40	6	40	15
4	Buona	Best Sum	10X4	40	4	14	10
2	Scadente	Image e sum	20X2	40	0	44	0
2	Buona	Image e sum	20X2	40	0	15	0
4	Scadente	Image e sum	10X4	40	1	46	2,5
4	Buona	Image e sum	10X4	40	0	15	0

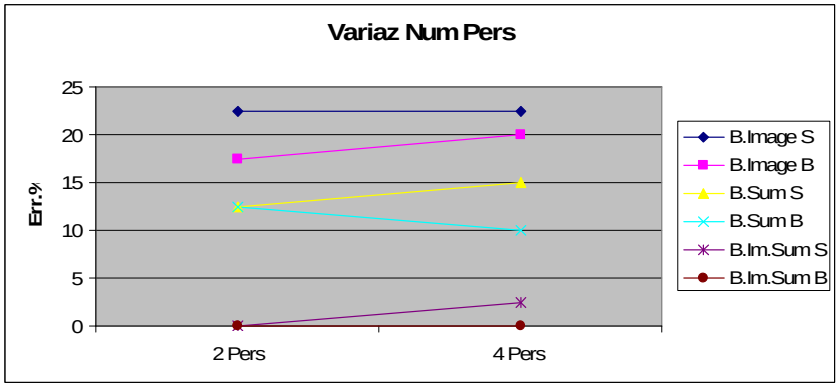
Performance (2/3)

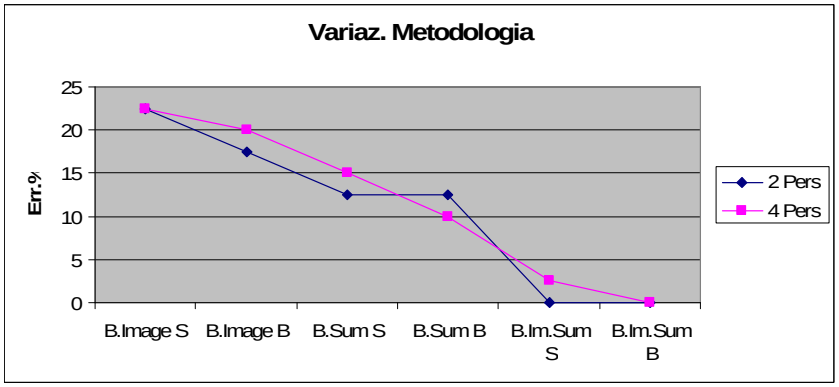


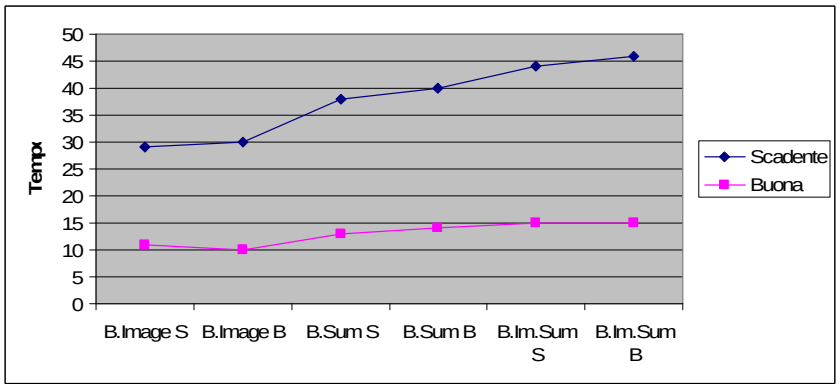


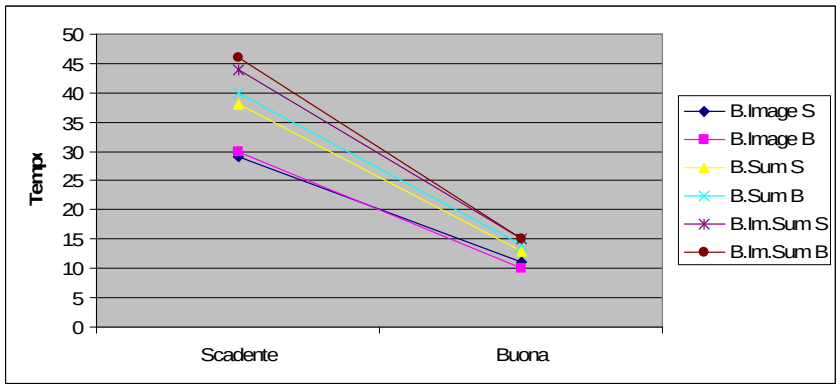


Performance (3/3)









Conclusioni

- L’algoritmo è complessivamente molto performante
  - Si raggiungono buoni risultati
  - La velocità dell’algoritmo permette nei processori attuali di effettuare ulteriore computazione per migliorare i risultati
  - Buona occupazione spaziale del DataBase
- Possibilità di parallelizzare il programma
  - Parallelismo di tipo Data Parallel
- Possibili sviluppi futuri
  - Calcolare una funzione che setti i parametri in automatico
  - Miglioramento del classificatore (esattamente frontali)
  - Calcolo delle caratteristiche facciali
  - Giochiamo sulla probabilità

Bibliografia

[1] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifi ers, 1997.

[2] Anonymous. Anonymous. In *Anonymous*, 2000.

[3] F. Crow. Summed-area tables for texture mapping. In *Proceedings of SIGGRAPH*, volume 18(3), pages 207–212, 1984.

[4] F. Fleuret and D. Geman. Coarse-to-fi ne face detection. *Int. J. Computer Vision*, 2001.

[5] William T. Freeman and Edward H. Adelson. The design and use of steerable fi lters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt '95*, pages 23–37. Springer-Verlag, 1995.

[7] H. Greenspan, S. Belongie, R. Gooodman, P. Perona, S. Rakshit, and C. Anderson. Overcomplete steerable pyramid fi lters and rotation invariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[8] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Patt. Anal. Mach. Intell.*, 20(11):1254–1259, November 1998.

[9] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[10] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *International Conference on Computer Vision*, 1998.

[11] D. Roth, M. Yang, and N. Ahuja. A snowbased face detector. In *Neural Information Processing 12*, 2000.

[12] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *IEEE Patt. Anal. Mach. Intell.*, volume 20, pages 22–38, 1998.

[13] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Stat.*, 26(5):1651–1686, 1998.

[14] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[15] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *International Conference on Computer Vision*, 2000.

[16] K. Sung and T. Poggio. Example-based learning for view-based face detection. In *IEEE Patt. Anal. Mach. Intell.*, volume 20, pages 39–51, 1998.

[17] J.K. Tsotsos, S.M. Culhane, W.Y.K. Wai, Y.H. Lai, N. Davis, and F. Nuflo. Modeling visual-attention via selective tuning. *Artifi cial Intelligence Journal*, 78(1-2):507–545, October 1995.

[18] Andrew Webb. *Statistical Pattern Recognition*. Oxford University Press, New York, 1999.