Consiglio Nazionale delle Ricerche Istituto di Calcolo e Reti ad Alte Prestazioni

Riconoscimento e identificazione automatica dei volti: EigenFaces, 2DLaplacianFaces, MultiRegecFaces

Enzo Imposimato, Mario Rosario Guarracino

Data:

RT-ICAR-NA-03-09 *ottobre 2009*

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: www.na.icar.cnr.it

Consiglio Nazionale delle Ricerche Istituto di Calcolo e Reti ad Alte Prestazioni

Riconoscimento e identificazione automatica dei volti: EigenFaces, 2DLaplacianFaces, MultiRegecFaces

Enzo Imposimato, Mario Rosario Guarracino

Data:

RT-ICAR-NA-03-09 *ottobre 2009*

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: www.na.icar.cnr.it

Introduzione

In questo elaborato verranno analizzati tre algoritmi per il riconoscimento dei volti: il classico "Eigenfaces", sviluppato nel 1991 da Matthew Turk del **Computer Science Department dell'Università della California** e da Alex Pentland del Mit Media Laboratory, il recente algoritmo denominato "2D Laplacianfaces method", proposto dai ricercatori Ben Niu e Simon Chi Keung Shiu del Dipartimento di Calcolo dell'Università Politecnico di Hong Kong, da Qiang Yang del Dipartimento di Informatica ed Ingegneria dell' 'Università di Scienze e Tecnologie di Hong Kong e da Sankar Kumar Pal dell'Istituto Indiano di Statistica di Kolkata, implementato interamente durante il periodo di tirocinio trascorso al CNR ed infine il "MultiRegecFaces" che nasce da alcuni cambiamenti che sono stati apportati al metodo multiclasse "MultiRegec", ideato dal prof. Mario Guarracino ricercatore all' ICAR, per ottenere un nuovo ed ulteriore riconoscitore di volti da confrontare con i primi due.

Conoscere e riconoscere. Per riconoscere bisogna innanzitutto conoscere. Sono stati effettuati molti studi su come l'uomo riesca, fin dai primi mesi di vita, a distinguere un volto da altri. Esiste anche una patologia denominata prosopagnosia (inabilità nel riconoscere volti familiari) che ha aiutato a comprendere che i volti sono "processati" dal cervello in maniera differente rispetto agli oggetti non volti, piuttosto vi è una evidente analogia con il modo i cui "riconosciamo" le parole quando leggiamo.

Il riconoscimento di un volto non avviene scorporato da altre informazioni: spesso conosciamo molte cose di una persona, ma non ne ricordiamo il nome; alcune volte il fatto di possedere delle informazioni aggiuntive aiuta nell'identificare la persona; altre informazioni come ad esempio contesto, postura, atteggiamenti corporei, espressioni emotive, movimenti nell'eloquio, manierismi ecc. contribuiscono nel riconoscimento di un volto.**[1]**

Alcuni esperimenti mostrano che, nel riconoscimento di volti familiari, un livello medio di accuratezza del 95% crolla al 50-60% quando gli stessi volti vengono presentati a testa in giù. Questo suggerisce che l'identità di un volto è codificata soprattutto dalle relazioni spaziali fra i suoi tratti (naso, bocca, occhi), quindi tra tutte le caratteristiche collegate assieme.
La 4 figure che seguono (le prime due foto sono uguali alle seconde due eccetto che per la sola rotazione di 180°) dovrebbero dimostrare che, effettivamente, quando osserviamo un volto lo confrontiamo nel suo insieme (e non a "pezzi") con un'immagine modello che costruiamo e immagazziniamo nella nostra mente.

Se non dimentichiamo un volto, probabilmente è perché da qualche parte, nel nostro cervello, viene conservata un'immagine di quella persona. Questo è suggerito anche dai risultati di una recente ricerca pubblicata su "Nature Neuroscience": l'immagine "registrata" dal cervello ci aiuta a riconoscere un vecchio amico. Le teorie sul riconoscimento dei volti sono concordi: dobbiamo avere un meccanismo di "rappresentazione", nella nostra memoria, con il quale confrontiamo, quasi istantaneamente, i volti che incontriamo, e che ci permette di riconoscere le persone che incontriamo.
Una nuova ricerca condotta da David Leopold, del Max-Planck-Institut di Tubinga, accredita l'ipotesi che queste immagini siano costruite sulla base di un "volto guida" che è la media di tutte le face che si incontrano. Ma è stato osservato che questo prototipo non è fisso, si può adattare. Quando il modello cambia, muta di conseguenza anche la capacità di riconoscere i volti familiari. I risultati che emergono da questo studio sono promettenti, e la conferma arriva da Anya Hurlbert, che studia il riconoscimento dei volti alla Medical School dell'Università di

Newcastle upon Tyne. "È possibile che il prototipo di confronto cambi in modo permanente in funzione dei volti che incontriamo più spesso". Insomma, se ci si trasferisce da Napoli ad un villaggio della Cina centrale, probabilmente il modello di faccia che abbiamo nel nostro cervello subirà dei mutamenti. Per la ricerca, il gruppo di Leopold ha sfruttato la tecnica del *morphing*, spesso utilizzata nell'industria del cinema. I ricercatori hanno creato un volto modello con caratteristiche che comprendevano le dimensioni del naso, la distanza tra gli occhi e l'ampiezza della fronte, basato su una media tra 100 diverse persone. Poi hanno scelto quattro persone reali, per il confronto, ma hanno anche generato un' "antifaccia" per ciascuna di esse. Per esempio, se Adam ha un naso più voluminoso del 50 per cento rispetto alla media, la sua anti-immagine ce l'aveva più sottile della metà, e così via. I ricercatori hanno anche creato versioni "indebolite" di queste persone e dei loro opposti, costruendo una striscia di volti che passavano da quello medio. E poi hanno fatto la stessa cosa con altri tre volontari. Dopo questa fase di preparazione, hanno chiesto ai soggetti in esame di analizzare le diverse versioni dei quattro volti, individuando fino a quando erano in grado di riconoscere gli individui originali. Con Adam, per esempio, si è verificato che il volto risultava riconoscibile fino a quando manteneva soltanto un terzo delle caratteristiche originali. Ma, una volta osservato il volto dell'anti-Adam, i soggetti si sono dimostrati in grado di riconoscere Adam da appena un decimo delle sue caratteristiche, mentre avevano maggiori difficoltà con gli altri volti. Questo effetto di rimodellamento del prototipo si annulla entro breve tempo, afferma il gruppo di Leopold, ma se cambia il contesto dei volti che si vedono "normalmente", l'effetto potrebbe divenire permanente, e ciò renderebbe più difficile riconoscere volti familiari. L'interpretazione di questo fenomeno, però, potrebbe essere differente, e saranno necessarie ulteriori verifiche prima che questa teoria possa essere confermata. **[2]**

Negli ultimi 10 anni sono stati fatti molti passi in avanti nel campo del riconoscimento automatico delle immagini. Quello del riconoscimento dei volti è un settore in cui si sta investendo particolarmente tanto per le innumerevoli applicazioni soprattutto nel settore della sicurezza. In particolare, può essere utilizzato nella sorveglianza, nella validazione di transazioni commerciali o finanziarie automatizzate, nel campo legale e giudiziario, nel controllo dell'accesso ai computers, ad ambienti sottoposti a restrizioni (installazioni militari, uffici, etc.) o dove sia comunque richiesto un elevato grado di sicurezza (aeroporti, banche, supermercati), nella realizzazione di applicazioni multimediali con interfacce uomo-macchina adattative, nell'autenticazione della posta elettronica, nella catalogazione automatica delle immagini e delle sequenze video, ultimamente è stato utilizzato un algoritmo di AFR (*Automatic Face Recognition*) per il riconoscimento dei volti delle salme.

Il continuo miglioramento delle tecnologie presenti nelle apparecchiature ottiche e nei calcolatori elettronici permette di immaginare un futuro, forse non troppo lontano, in cui molte delle telecamere presenti ad esempio negli stadi o negli aeroporti non saranno più dei registratori passivi di immagini di persone e delle loro azioni ma, collegate ad un computer e ad un apposito software, saranno in grado di avvisare, ad esempio, della presenza di eventuali ricercati o di riconoscere e segnalare in qualunque istante la presenza di imminenti situazioni di pericolo. In questo caso si parla più specificatamente di riconoscimento delle azioni.

Per avvicinarsi al mondo del riconoscimento dei volti bisogna innanzitutto capire in che modo un calcolatore può essere "istruito" a svolgere tale compito. Come sempre è utile "prendere spunto" da quello che già conosciamo (o presumiamo di conoscere) dei metodi utilizzati dal nostro stesso cervello per ottenere tale scopo. Infatti, quando si scrive un algoritmo, può capitare di credere di inventare delle tecniche senza rendersi conto che esse sono già esistenti in natura e magari perfezionate da secoli di evoluzione.

Nel primo capitolo, dopo una breve introduzione alla classificazione in genere, sono stati descritti due algoritmi particolarmente rilevanti per riconoscimento dei volti: EigenFaces, che

si colloca tra gli algoritmi più veloci ed utilizzati finora, ed il nuovo 2D-LaplacianFaces, che ha generato le accuratezze più elevate e di cui si è implementata una versione per MatLab durante il periodo di tirocinio al CNR.
Il secondo capitolo descrive in che modo è stato trasformato il metodo sperimentale di classificazione MultiREGeC per ottenere un nuovo riconoscitore di volti da confrontare.
Il terzo ed ultimo capitolo contiene l'analisi del codice utilizzato per i tests e le tabelle delle accuratezze ottenute con databases differenti.

1 - Tecniche di riconoscimento dei volti: "EigenFaces" ed il nuovo "2D-Laplacianfaces"

1.1 *La classificazione*

Il termine classificazione viene utilizzato in tutte quelle attività che si possono ricondurre alla gestione delle conoscenze. Le attività di classificazione hanno lo scopo di organizzare le diverse entità presenti del dominio in esame in modo che possano essere rappresentate servendosi di criteri riconducibili ad alcune proprietà intrinseche degli elementi, estrapolando da essi in certi casi anche alcune regole precise e procedurali.
I modelli di classificazione si collocano tra i metodi di apprendimento supervisionato, infatti, a partire da un insieme di osservazioni riferite al passato per le quali è già nota la classe di appartenenza, i "classificatori" si propongono di generare un insieme di regole che consentano di predire la classe di appartenenza anche di osservazioni future.

1.2 Le tre fasi principali della classificazione

Per creare un modello di classificazione è necessario procedere secondo tre fasi principali: fase di training, fase di test e fase di predizione.

Fase di training. Nella fase di training l'algoritmo di classificazione viene applicato alle entità appartenenti ad un sottoinsieme dei dati a disposizione, denominato training set, con lo scopo di ricavare delle regole che consentano di attribuire a ciascuna osservazione x la corrispondente classe target y. Nei modelli per il riconoscimento dei volti l'osservazione coincide con un'immagine di un individuo e la classe target è la sua stessa identità.

Fase di test. Nella fase di test, le regole prodotte nel corso della fase precedente, che stabiliscono le relazioni che intercorrono tra le osservazioni e le loro classi di appartenenza, vengono impiegate per classificare le osservazioni dell'insieme di dati a disposizione (D) non incluse nel training set (T), per le quali è noto il valore della classe target. Per valutare l'accuratezza del modello di classificazione la classe di appartenenza di ciascun elemento dell'insieme $V = D - T$, denominato *test set,* viene confrontata con la classe predetta da parte del classificatore. Per evitare di incorrere in una sovrastima dell'accuratezza del classificatore è necessario che il training-set ed il test-set siano disgiunti, cioè si deve far in modo che i due insiemi non abbiano alcuna osservazione in comune. In certi casi è possibile eseguire comunque la fase di test sull'insieme utilizzato per il training per accertarsi che il modello che si sta costruendo non alteri in maniera disastrosa le proprietà intrinseche degl'elementi.

Fase di predizione. La fase di *predizione* corrisponde all'effettivo utilizzo del modello di classificazione per assegnare la classe target alle nuove osservazioni che si ricevono in input. La

predizione viene ottenuta applicando le regole generate in fase di training alle variabili esplicative che descrivono la nuova istanza. **[3]**
Oggi i modelli di classificazione si applicano a moltissimi campi, forse a tutti i campi della conoscenza e molti di questi modelli hanno raggiunto ampia affidabilità. Molti classificatori sono sottoposti a procedimenti di standardizzazione, alcune diventano standard ufficiali, altri standard de facto. La definizione dei modelli di classificazione richiede grande impegno, ma spesso bisogna trovare dei compromessi ed è successo che abbiano causato duri scontri fra gruppi di potere con diversi interessi e posizioni ideologiche.

1.3 Tecniche di riconoscimento

I primi lavori di ricerca sul riconoscimento automatico di volti sono iniziati più di venti anni fa; negli ultimi anni l'attività in questo campo ha conosciuto un nuovo rilancio essenzialmente perché l'hardware utilizzato è di più elevata potenza ed ha costi relativamente contenuti ed inoltre sono cresciute le esigenze legate allo sviluppo di algoritmi di AFR per la videosorveglianza. Tale maggior interesse ha quindi spinto la ricerca a proporre (o, in molti casi, riproporre) una vasta varietà di metodi basati su presupposti di volta in volta differenti in base alle problematiche specifiche da affrontare.
Se in input vengono fornite delle immagini scattate controllando la posa e l'illuminazione, le tecniche di AFR finora conosciute permettono di conseguire un livello di riconoscimento solitamente superiore al 90% anche su vasti database. Anche se questo valore puo' essere considerato un buon tasso di accuratezza per un sistema automatico, in realtà si è ancora lontani dall'efficienza che ha un essere umano nell'effettuare un riconoscimento. Tuttavia ci si può affidare agli algoritmi di AFR per numerose applicazioni come l'investigazione giudiziaria, il controllo di accesso o per le applicazioni multimediali con interfacce uomo-macchina adattative. I recenti progressi della ricerca permettono comunque di poter sperare che, in un futuro non troppo lontano, le prestazioni dei sistemi automatici possano eguagliare, se non addirittura superare, quelle dell'uomo.

1.4 Riconoscimento o identificazione?

E' necessario fare una distinzione tra riconoscimento e identificazione. L'identificazione consiste nell'utilizzo di opportuni algoritmi per stabilire, con una certa accuratezza, se un soggetto sia presente o meno tra le immagini di volti già identificati utilizzati nella fase di apprendimento, nel caso in cui non si riesca ad identificarlo per ragioni di natura diversa come cattiva illuminazione della foto, espressione innaturale del viso, cattiva angolatura dell'inquadratura, di fronte ad incertezze sulle generalità di una persona, etc.

Il riconoscimento è un processo 1 ad 1 che ci consente di stabilire se, tra le immagini che sottoponiamo all'algoritmo, sia presente o meno un particolare soggetto di cui già supponiamo di conoscere l'identità. Per riconoscimento di volti, in senso più generale, si può anche intendere l'appurare se nell'immagine in input sia effettivamente presente l'immagine di un volto di una persona oppure no, distinguendolo dallo sfondo e dagl' altri oggetti eventualmente presenti nella foto.

1.5 Il metodo Eigenfaces

Il metodo Eigenfaces, conosciuto anche come "metodo delle autofacce", è stato sviluppato nel 1991 da Matthew Turk del Computer Science Department of University of California e da Alex Pentland Ph.D. del Mit Media Laboratory. In questa tecnica le osservazioni sono viste come set di immagini di facce prese sotto le stesse condizioni di luce, normalizzate per allineare occhi e bocca, e campionate alla stessa risoluzione. Le eigenfaces saranno gli autovettori usati come base del nuovo spazio e nelle cui coordinate sono descritte tutte le facce. Ogni eigenface descrive una certa caratteristica come la linea dei capelli, la simmetria, la larghezza del naso o i contorni di quelle componenti non chiaramente distinguibili. Se consideriamo, ad esempio, immagini 100x100 pixel avremo 10000 possibili componenti ma in realtà la maggior parte degli individui può essere riconosciuto con un numero tra le 100 e le 150 eigenfaces. E' come se ogni faccia umana potesse essere considerata come una combinazione lineare di queste "facce standard" derivate dall'analisi di un grosso insieme di immagini di facce.

Eigenfaces è un metodo estremamente rapido: la sua complessità di tempo nella fase di training è $O(n^2m^2L)$ e nella fase di test di O(LMN), con n, m, L, M, N, numero di righe e di colonne della matrice immagine, numero di vettori proiezione, elementi di test e di taining, rispettivamente. La complessità di spazio è $O(n^2m^2)$.

1.6 Il metodo 2D Laplacianfaces

In questa sezione sarà fornita una descrizione dettagliata dell'algoritmo 2D Laplacianfaces, e verrà descritto in che modo si differenzia dal metodo Laplacianfaces mono-dimensionale. Nel terzo capitolo verrà analizzata la parte relativa al codice per la preparazione dell'input, per l'estrazione delle caratteristiche e per effettuare i tests di classificazione.
Il metodo Laplacianfaces è stato sviluppato recentemente proprio per il riconoscimento automatico dei volti. E' una naturale generalizzazione dell'algoritmo LLE (Locally Linear Embedding) che già aveva dimostrato di riuscire a controllare efficacemente la non linearità dello spazio immagine riducendone le dimensioni. L'idea principale del Laplacianfaces è infatti quella di scovare una rappresentazione dei dati in poche dimensioni che possa preservare il più possibile la loro località.
A differenza dell' Eigenfaces e del Fisherfaces, che ricercano le proiezioni ottimali analizzando i patterns globali delle densità dei dati, il metodo Laplacianfaces trova le sue soluzioni ottimali esaminando più da vicino la geometria locale delle immagini di training. Le caratteristiche che vengono apprese sono utili a preservare la località dei dati di training, rendendoli robusti nei casi più misclassificanti del training set ed adatti alla classificazione basata sul metodo k-nearest neighbor. E' stato osservato che il Laplacianfaces "batte" il più famoso Eigenfaces ed il Fisherfaces sui databases Yale, MSRA e PIE. Questi ultimi due algoritmi, infatti, sono poco efficienti nel momento in cui, nel manipolare le equazioni degli autovalori, la complessità di memoria e computazionale cresce esponenzialmente con i vettori delle immagini di training.
Alcuni ricercatori hanno provato a migliorare l'efficienza dei metodi Eigenfaces e Fisherfaces utilizzando la tecnica della proiezione delle immagini. Liu et al. e Yang et al. hanno sviluppato la versione a due dimensioni del metodo Eigenfaces, Xiong et al. e Jing et al. quella del Fisherfaces. Entrambi i metodi riducono drasticamente la complessità degli algoritmi da $O(m^2 \times n^2)$ ad O(m

^{2}) o ad O(n^2). Questi metodi riducono anche la dimensione delle matrici nelle equazioni degli autovalori consentendone, successivamente, una valutazione più accurata.
Per i metodi 2D Eigenfaces e Fisherfaces la tecnica della proiezione delle immagini ha portato buoni risultati, in questo elaborato verrà applicata anche al Laplacianfaces. Il nuovo algoritmo ha mostrato di sbaragliare gli altri algoritmi di riconoscimento dei volti come il Laplacianfaces monodimensionale, il 2D Eigenfaces ed il Fisherfaces.

1.7 L'idea e l'algoritmo 2D-Laplacianfaces

Sia *X* un vettore colonna unitario *n*-dimensionale. *A* rappresenta un'immagine di *m* righe ed *n* colonne. Nel metodo 1D Laplacianfaces l'immagine campione *A* andava trasformata in un vettore di dimensione
m x *n* prima della fase di training. Invece nel nuovo algoritmo 2D Laplacianfaces la matrice associata all'immagine viene proiettata direttamente lungo il vettore *X:*

$$Y = AX. \qquad (1)$$

Il vettore *m*-dimensionale *Y* così ottenuto è chiamato *feature vector* e
corrisponde alla proiezione orizzontale dell'immagine *A.* Dato un insieme
di immagini di training T = {A_1, …, A_i, …, A_j, …,A_n} la funzione obiettivo è così definita:

$$\min_X \sum_{ij} \|Y_i - Y_j\|^2 S_{ij}, \qquad (2)$$

dove Y_i è il feature vector proiezione corrispondente all'immagine A_i, $||\cdot||$
è la norma due ed S_{ij} è la matrice di similarità tra le immagini A_i ed A_j nello spazio osservazione ed è definita così:

$$S_{ij} = \begin{cases} \exp(-\|A_i - A_j\|^2/t) & \text{se } x_i \text{ è tra i k-nearest neighbors di } x_j \text{ o } x_j \text{ è tra i k-nearest neighbors di } x_i \\ 0 & \text{altrimenti} \end{cases} \qquad (3)$$

dove k è la dimensione del local neighborhood, e t è la larghezza della finestra che determina il tasso di decadimento della funzione di similarità.
Si può notare dall' equazione *(2)* che la funzione obiettivo impone una grossa penalità se due arbitrarie immagini campione sono mappate distanti nello spazio d'origine. Minimizzare questa

funzione assicura che se A_i ed A_j sono vicine tra loro, i loro feature vectors proiezione Y_i e Y_j sono anch'essi vicini. Pertanto la località dello spazio campione può essere massimamente preservata nel feature space attraverso le proiezioni. Effettuando alcuni passaggi algebrici, il metodo 2D Laplacianfaces è formulato in modo da minimizzare la seguente funzione obiettivo:

$$\begin{aligned}\min_X \sum_{ij} \|Y_i - Y_j\|^2 S_{ij} &= \sum_{ij} \|A_i X - A_j X\|^2 S_{ij} \\ &= \sum_{ij} [X^{\mathrm{T}}(A_i - A_j)^{\mathrm{T}}(A_i - A_j)X] S_{ij} \\ &= X^{\mathrm{T}} \left[\sum_i A_i^{\mathrm{T}} A_i \sum_j S_{ij} - \sum_{ij} A_i^{\mathrm{T}} S_{ij} A_j \right] X \\ &= X^{\mathrm{T}} A^{\mathrm{T}} (D - S) A X = X^{\mathrm{T}} A^{\mathrm{T}} L A X,\end{aligned} \tag{4}$$

dove $A^T = [A_1^T, ..., A_n^T]$ e $A = [A_1, ..., A_n]^T$ prendono le operazioni matematiche come blocchi di matrice di dimensioni 1 x N e N x 1, le cui righe e colonne consistono nella matrice immagine A_i^T e A_i, con i=1, ..., N, rispettivamente.
D è la matrice diagonale a blocchi di dimensioni N x N, i cui elementi diagonali sono d_{ii}, con $d_{ii} = \sum_j S_{ij}$, che corrispondono alle somme dei valori di similarità di tutte le immagini campione con la i-sima immagine nello spazio d'origine. S è la matrice di similarità ed L è chiamata matrice Laplaciana. Entrambe queste matrici sono di dimensioni N x N. Ogni elemento della matrice D indica quanto sia importante ciascun punto. Inoltre viene imposto questo vincolo:

$$X^{\mathrm{T}} A^{\mathrm{T}} D A X = 1. \tag{5}$$

Quindi, il metodo 2D Laplacianfaces verrà formulato come il

$$\begin{aligned}\min_X \quad & X^{\mathrm{T}} A^{\mathrm{T}} L A X, \\ \text{s.t.} \quad & X^{\mathrm{T}} A^{\mathrm{T}} D A X = 1.\end{aligned} \tag{6}$$

Nell'equazione (6), la matrice D fornisce una misura naturale dell'importanza delle immagini di training.
Nello spazio d'origine dei dati, gli elementi misclassificanti hanno

pochi oggetti vicini rispetto a quelli delle regioni di alta densità di distribuzione. Un po' di distorsione della geometria locale nei pressi di questi elementi anomali, dopo la trasformazione, è improbabile che abbia un impatto significativo sul risultato della classificazione. Quindi, nel determinare le direzioni delle proiezioni, sono meno importanti di quegl'elementi che hanno neighbors più vicini. Nell'equazione (6), utilizzando il vincolo, si è in grado non solo di rimuovere l'arbitrario fattore di scala dei vettori proiezione, ma anche di prendere in considerazione l'importanza di ogni singolo elemento per l'ottimizzazione.

Applicando il metodo del moltiplicatore di Lagrange, siamo in grado di ridurre l'equazione (6) ad un problema generalizzato di autovalori come mostrato nell'equazione (7)

$$A^{\mathrm{T}} LAX = \lambda A^{\mathrm{T}} DAX, \qquad (7)$$

dove le matrici A^TL A e A^TD A sono entrambe di dimensione N x N, ed L e D sono simmetriche e positive semidefinite. Possiamo calcolare il vettore proiezione ottimale X risolvendo proprio questa equazione. Gli autovettori associati ai primi d autovalori più piccoli saranno utilizzati per la selezione delle caratteristiche.

1.8 Estrazione delle caratteristiche

Denotiamo i vettori proiezione ottimali con X_1, ..., X_d. Data un'immagine *A* in input, sia $Y_i = AX_i$, con i = 1, ...,d. Possiamo allora ottenere un insieme di feature vectors proiezione, Y_1, ..., Y_d. Notiamo che le caratteristiche (features) estratte nel metodo 2D Laplacianfaces sono vettori, mentre nell'algoritmo originario monodimensionale erano scalari. I vettori proiezione sono adoperati per formare una matrice $B = [Y_1, ..., Y_d]$ di dimensioni m x d denominata la matrice delle features dell'oggetto immagine A.

1.9 2D-Laplacianfaces: predizione

Una volta ottenuta la matrice delle caratteristiche di tutte le immagini di training, viene utilizzato il classificatore one-nearest neighbor per la fase di classificazione. La distanza tra ogni coppia di matrici di features
$B_i = [Y_{i1}, ..., Y_{id}]$ e $B_j = [Y_{j1},...,Y_{jd}]$ è definita da

$$d(B_i, B_j) = \sum_{p=1}^{d} \|Y_{ip} - Y_{jp}\|. \qquad (8)$$

Supponiamo che le matrici delle features siano B_1, ..., B_N ed ognuna di esse sia associata ad una classe denominata C. Data in input un'immagine di test B, se $d(B, B_1) = \min d (B, B_j)$ e B_1 appartiene alla classe C, allora B è classificata come appartenente a C. **[4]**

2 - Il metodo MultiReGEC

2.1 Introduzione al metodo

Il MultiReGEC (Multiclass Regularized Eigenvalue Classifier), ideato e realizzato dai ricercatori Mario Rosario Guarracino, Antonio Irpino e Rosanna Verde, è un nuovo modello di classificazione che estende il modello GePSVM (Generalized Proximal Support Vector Machine) ai problemi di classificazione multiclasse. La nuova tecnica si basa su alcune considerazioni statistiche e di geometria che forniscono delle solide basi all'algoritmo. Verrà dimostrato, attraverso le analisi degli esperimenti effettuati, che l'accuratezza del GePSVM può essere ben confrontata con quella delle altre tecniche finora utilizzate.
La classificazione multiclasse si differisce dalla classificazione binaria in quanto permette di operare su un insieme di dati che appartengono a diversi tipi di classi. Si riferisce alla capacità di un sistema di imparare, dagli esempi forniti in input, come classificare le entità in classi multiple. Il sistema quindi impara sempre da un insieme di casi (training-set) proprio come accade nella classificazione binaria. Ogni caso nel training set è descritto da una serie di features (caratteristiche) e da un etichetta di classe. Per ogni nuova entità il sistema addestrato predice l'etichetta della classe di appartenenza.
E' stato dimostrato che gli algoritmi multiclasse possono essere applicati con successo in numerosi casi, come il riconoscimento dei caratteri, dei rischi del menagement, delle prognosi e diagnosi mediche. Nel mio lavoro studierò i suoi risultati nel riconoscimento dei volti. Non esiste un'unica tecnica che possa risolvere efficacemente tutti i problemi sopra citati. Ne esistono varie ed ognuna di esse si comporta in maniera più o meno diversa delle altre a seconda dei problemi su cui viene applicata. Le SVMs (Support Vector Machines) rappresentano lo stato dell'arte dei metodi di classificazione per la loro accuratezza nella previsione in numerosi problemi differenti. Le SVMs cercano un piano discriminante nell'insieme dei support vectors. Un altro esempio degno di nota è il Fisher Linear Disciscriminant Analysis (LDA), che cerca il piano discriminante utilizzando una tecnica di trasformazione delle features. E' stato dimostrato che il modello decisionale ottenuto mediante SVMs nella classificazione binaria è equivalente alla soluzione ottenuta mediante LDA.
Tutte queste tecniche hanno pero' anche diversi svantaggi. Il k-nearest Neighbors (kNN), che è il più semplice algoritmo di machine learning, ad esempio ha la tendenza ad assegnare i nuovi elementi alla classe più grande del dataset. Gli alberi di decisione possono generare un gran numero di regole per classificare i dati anche per insiemi di dati abbastanza semplici. LDA, nei problemi multiclasse, massimizzando la media della distanza quadratica tra le classi in uno spazio di dimensione pari al numero di classi meno uno, non considera il tasso d'errore nella classificazione. Risulta particolarmente evidente che quando ci sono coppie di classi che si trovano a grandi distanze tra loro nello spazio, poichè la trasformazione che viene effettuata preserva proprio le distanze delle classi in partenza già ben separate, avremo una bassa accuratezza di classificazione.
La classificazione binaria è solitamente il mattone che costruisce i metodi multiclasse. One-versus-one (uno contro uno) e One-versus-the-rest (uno contro tutti gli altri) sono alcune tecniche per decomporre problemi multiclasse in un insieme di problemi di classificazione binaria. Tutti questi metodi sono limitati.
La tecnica utilizzata nel MultiReGEC è interessante sotto numerosi punti di vista. Innanzitutto è possibile implementarlo in diversi ambienti per il problem-solving come MatLab, R e Weka, poichè risolve semplicemente un problema generalizzato di autovalori ed i kernels che effettuano questa computazione sono già stati sviluppati negli ambienti appena citati. In secondo luogo, poichè basato sulla soluzione di un problema generalizzato di autovalori ed alla decomposizione dei vettori singolari, la sua esecuzione in parallelo su computers multicore e

multicomputers puo' essere efficientemente codificata in linguaggi di programmazione ad alto livello utilizzando le librerie software matematiche già disponibili. In fine l'accuratezza di questa nuova tecnica si puo' ben comparare con quella dei metodi già esistenti e fornisce un approccio costruttivo per l'estensione dei metodi binari ai problemi multiclasse.

2.2 Formulazione dell'algoritmo MultiRegec

Nel caso in cui due classi sono separate linearmente, l'algoritmo di classificazione SVM (Support Vector Machines) cerca un iperpiano che separa per gli elementi appartenenti alle due classi distinte. L'iperpiano separatore è solitamente scelto in modo da minimizzare lo spazio tra le due classi. Il margine puo' essere definito come la massima distanza tra due iperpiani paralleli $w'x - b = \pm 1$ che ponga tutti gli elementi delle due classi in zone separate. L'iperpiano di classificazione è quello situato nel mezzo e parallelo a quello che massimizza il margine.
I punti che sono più vicini all'iperpiano sono chiamati *support vectors*, e sono gli unici punti necessari per addestrare il classificatore.
Si considerino due matrici $A \in R^{n \times m}$ e $B \in R^{k \times m}$, che rappresentano le due classi, ogni riga corrisponde ad un punto nello spazio delle features.
La formula quadratica vincolata linearmente per ottenere gli iperpiani ottimali (w, b) è

$$\begin{aligned} &\min f(w) = \frac{w'w}{2} \qquad (1) \\ &(Aw + b) \geq e \\ &(Bw + b) \leq -e, \end{aligned}$$

dove e è il vettore unità delle dimensioni appropriate.
Mangasarian et al. propongono di classificare questi due insiemi di punti A e B utilizzando due iperpiani, ciascuno più vicino ad un insieme di punti e più lontano dall'altro. Sia $x'w - \gamma = 0$ un iperpiano in R^m. Per soddisfare la condizione precedente per i punti in A, l'iperpiano puo' essere ottenuto risolvendo questo problema di ottimizzazione:

$$\min_{w,\gamma \neq 0} \frac{\|Aw - e\gamma\|^2}{\|Bw - e\gamma\|^2}. \qquad (2)$$

L'iperpiano per B puo' essere ottenuto minimizzando l'inverso della funzione obiettivo della (3). Ora, sia

$$G = [A \quad -e]'[A \quad -e], \quad H = [B \quad -e]'[B \quad -e], \quad z = [w' \quad \gamma]', \qquad (3)$$

quindi l'equazione (2) diventa:

$$\min_{z \in R^m} \frac{z'Gz}{z'Hz}. \qquad (4)$$

2.3 Support Vector Machines Multiclasse

La formulazione di SVM nel paragrafo precedente è stata basata su un problema di classificazione binaria. Come accennato nell'introduzione, si possono applicare strategie differenti agl'algoritmi base per generalizzare il problema di classificazione alle classi multiple. Essenzialmente esistono due strategie: la prima consiste nel suddividere il problema in sottoproblemi, ognuno dei quali diventa un problema di classificazione binaria. Una possibilità è quella di costruire k classificatori, uno per ogni classe classificata rispetto alle altre. Questo metodo ha però lo svantaggio di ottenere classi sbilanciate, anche se tutte le classi sono composte dallo stesso numero di elementi. In più i k problemi hanno tutti la stessa complessità di quello iniziale, nel senso che ogni modello di classificazione è valutato utilizzando tutto il training-set. Una seconda possibilità è usare $k(k-1)/2$ classificatori binari uno-contro-uno, ed utilizzare un voto a maggioranza o a coppie per etichettare i casi del test. Quest'ultimo riduce la complessità computazionale del classificatore singolo, dal momento che è utilizzato soltanto su due classi per volta. Resta ancora quindi il problema di decomporre un problema multiclasse in molteplici problemi binari indipendenti.
In fine è possibile estendere la formulazione di SVM ai problemi multiclasse per esempio definendo un margine per i problemi multiclasse.
Come verrà evidenziato nel prossimo paragrafo, è stato ridotto il problema multiclasse in un problema di classificazione binaria ed è stato costruito un modello di classificazione per ciascuna classe come media di tutti i modelli costruiti rispetto alle altre classi. Questa tecnica scompone il problema in sottoproblemi piccoli, e fornisce una soluzione che dipende da ciascuno di essi.

2.4 L'algoritmo MultiRegecFaces

Per spiegare meglio l'idea che sta alla base della tecnica multiclasse, inizierò con un esempio. Supponiamo che il problema sia costruire un modello di classificazione per un dataset linearmente divisibile, composto da due caratteristiche e diviso in quattro classi A_i, con i = 1,2,3,4.
Seguendo l'idea del GePSVM per separare la classe A_1 dalle altre tre classi, è possibile costruire tre piani $w_l' x - \gamma_l = 0$, con l = 2,3,4. E' possibile stimare una media di queste rette, valutando la media $\tilde{w}$ dei vettori normali con coefficienti w_i e imponendo che il baricentro del triangolo formato dai tre piani sia soluzione del piano ortogonale. Questo è raffigurato nella Figura 1, in cui è disegnata la media dei quattro piani, una per ogni classe,.
Per estendere il precedente esempio a k classi in uno spazio ad *m* dimensioni, possiamo avvalerci del seguente algoritmo di addestramento:
Per ciascuna classe A_i, con i = 1,...,k:

a) calcola i k - 1 iperpiani discriminanti $w'_{i,l}x - \gamma_{i,l} = 0$, con l = 1,...i - 1, i + 1,...k e sia

$$Z_i = [z_{i,1}, \ldots, z_{i,i-1}, z_{i,i+1}, \ldots, z_{i,k-1}],$$

con $z_{i,l} = [w'_{i,l} \quad \gamma_{i,l}]'$

b) Calcola i vettori normalizzati $\bar{W}_i =$

$$[\bar{w}_{i,1}, \ldots, \bar{w}_{i,i-1}, \bar{w}_{i,i+1}, \ldots, \bar{w}_{i,k}]$$

con $\bar{w}_{i,l} = w_{i,l}/\|w_{i,l}\|_2$

c) Calcola la scomposizione tramite SVD di

$$\bar{W}_{i:}\bar{W}_i = USV'.$$

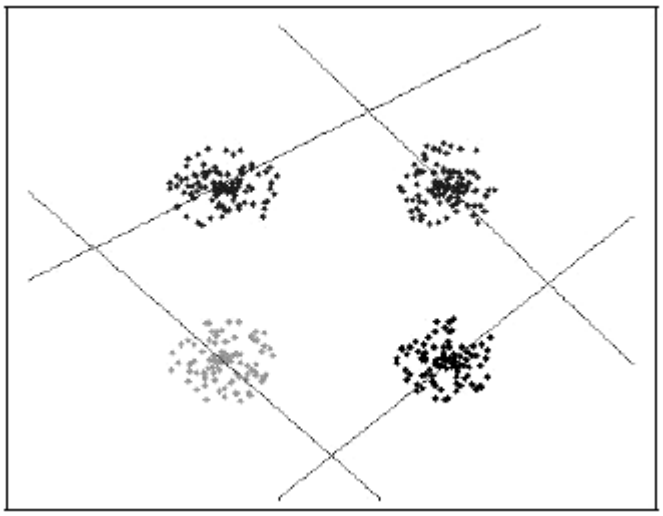


Figura 1. Classificazione di 4 classi in uno spazio a due dimensioni

d) Calcola l'iperpiano medio $[\tilde{w}_i \quad \tilde{\gamma}_i]' = Z_i U_1.$

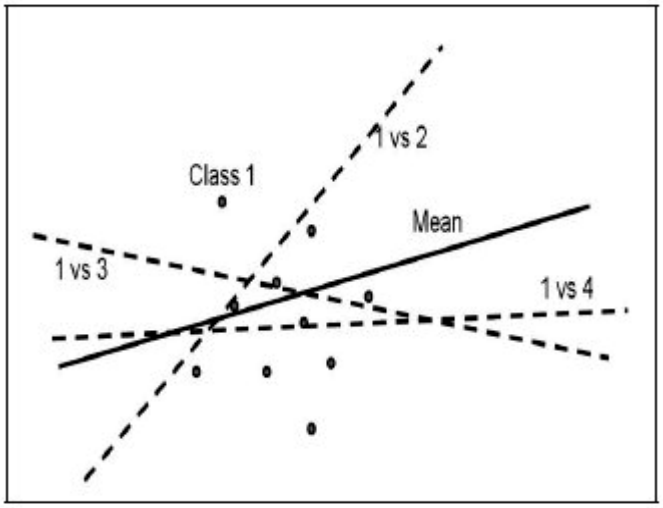


Figura 2. Iperpiano medio della classe A1

Nella decomposizione di SVD, S è la matrice diagonale che contiene i valori singolari in ordine decrescente, U è la base composta da autovettori dello spazio dei dati e V attraversa lo spazio della classe. Il vettore colonna U_1, dove U_1 è la prima colonna di U, rappresenta la direzione lungo la quale la correlazione tra i vettori normali $\bar{W}_i$ degli iperpiani è massima. La combinazione lineare dei coefficienti Z_i degli iperpiani con costanti U_1 è l'iperpiano con la massima correlazione con i piani Z_i.
Nella Figura 2 è tracciata una rappresentazione dell'iperpiano medio.
Per predire un'etichetta di classe di una nuova osservazione occorre soltanto selezionare l'iperpiano a distanza minima. Per evitare i casi in cui un iperpiano si interseca con l'inviluppo convesso di un altra classe, consideriamo la proiezione dei punti nella classe sul loro iperpiano. La distanza da un iperpiano diventa la distanza di un elemento del test dal punto proiettato più vicino di una classe data sul corrispondente iperpiano. Questo è rappresentato chiaramente in Figura 3, in cui l'elemento di test viene assegnato alla classe 4.**[5]**

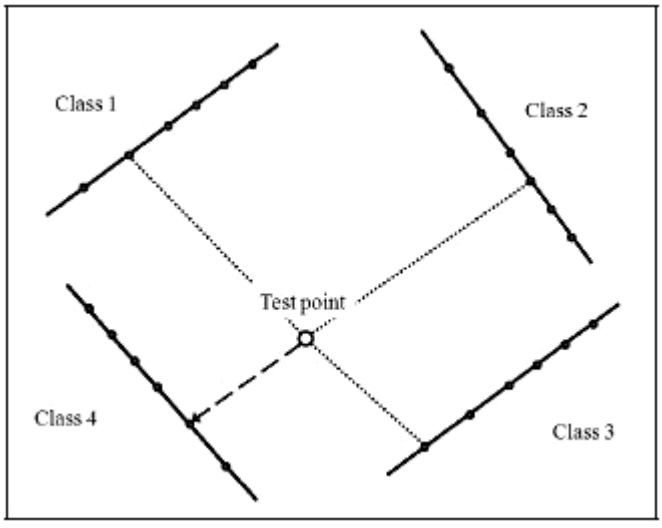


Fig. 3 Predizione della classe in uno spazio a due dimensioni

3. Analisi del codice e tests

3.1 Descrizione del codice di Eigenfaces

Per identificare i volti tramite questo algoritmo occore creare un set di "eigenfaces" per l'apprendimento: le facce che costituiscono il training-set devono essere preparate per essere processate, devono avere tutte la stessa dimensione ed in più andrebbero allineate fra loro rispetto ad occhi e bocca. Nei tests eseguiti non è stato effettuato quest'ultimo passaggio poichè le immagini utilizzate risultano già piuttosto allineate tra loro.
Ogni immagine è vista come un vettore, quindi è necessario concatenare per righe i pixels della matrice originale. Un'immagine in scala di grigi con r righe e c colonne è quindi rappresentata come un vettore con r x c elementi.
Le dimensioni delle singole foto di ogni dataset sono state ridotte per mantenere sempre il formato 100x100pxl circa.
Si assume che tutte le immagini del training set siano state memorizzate in una sola matrice "allsamples", dove ogni riga della matrice è un immagine. Questi passaggi vengono eseguiti dal codice riportato di seguito:

```
////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
allsamples=[];
soggetti=40;
rsz=0.4;
espressioniTr=2;
espressioniTe=8;
for i=1:soggetti
   for j=1:espressioniTr
      a=imread(strcat('C:\facerec\groups\by_orl\s',num2str(i),'\',num2str(j),'.pgm'));
      a=imresize(a,rsz);
      b=a(1:size(a,1)*size(a,2));
      b=double(b);
      allsamples=[allsamples;b];
   end
end
```

```
////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Successivamente occorre sottrarre la media: l'immagine media deve essere calcolata per colonne e sottratta da ogni colonna di ogni immagine originale in "allsamples".

```
////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
samplemean=mean(allsamples);

for x=1:size(allsamples,1),
   T(x,:)=allsamples(x,:)-samplemean;
end
```

```
////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

A questo punto bisogna calcolare gli autovalori e gli autovettori della matrice di covarianza "S":

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
S=T*T';
[v d]=eig(S);
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Ogni autovettore ha le stesse dimensioni dell'immagine originale e può essere di per sé considerato come un'immagine. Gli autovettori di tale matrice di covarianza sono chiamati "eigenfaces" o "autofacce". Corrispondono alle direzioni principali lungo le quali le immagini di apprendimento differiscono dall'immagine media. Ora è necessario effettuare la scelta del numero di componenti principali da utilizzare. Ogni autovettore rappresenta una direzione nello spazio immagine, è utile quindi conservare gli autovettori con gli autovalori associati più grandi e scartare i restanti. Per ottenere questo risultato si è scelto di procedere incrementando la variabile contatore *p* fino al raggiungimento di quel numero di autovalori necessario per preservare circa il 90% delle informazioni complessive. Questo ciclo, insieme all'estrazione e all'ordinamento della diagonale di autovalori *d*, è riportato nel codice seguente:

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
d1=diag(d);
%
dsort=flipud(d1);
vsort=fliplr(v);

%
dsum=sum(dsort);
dsum_extract=0;
p=0;

while(dsum_extract/dsum<0.9)
   p=p+1;
   dsum_extract=sum(dsort(1:p));
end
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Gli autovalori associati ad ogni autofaccia rappresentano quanto variano le immagini dell'insieme di addestramento rispetto all'immagine media in quella direzione.
Proiettare un'immagine su un sottoinsieme di autovettori provoca la perdita di alcune informazioni, questo effetto non desiderabile è però minimizzato dall'aver conservato quelle autofacce con autovalori più grandi associati.
Se le immagini sono in formato 100x100 pxl si ottengono 10.000 autovettori. Generalmente molte facce possono essere rappresentate utilizzando dalle 100 alle 150 eigenfaces quindi la maggior parte dei 10.000 autovettori può essere scartata.

La porzione di codice seguente illustra come viene creata la variabile *"base",* secondo la formula $\text{base} = \text{T' * vsort}(:, 1{:}\text{p}) * \frac{1}{\sqrt{diag(dsort(1:p)}}$, che, nella riga successiva, viene utilizzata per proiettare *allsamples* in uno spazio ridotto:

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
base = T' * vsort(:,1:p) * diag(dsort(1:p).^(-1/2));
allcoor=allsamples*base;
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Adesso è possibile utilizzare le eigenfaces per rappresentare nuove facce: siamo in grado di proiettare una nuova immagine (sottraendo sempre la media) sulle eigenfaces e quindi registrare quanto il nuovo volto si discosta dalla faccia media. Il codice seguente illustra che si è scelto di salvare gl'indici delle tre immagini a minor distanza candidate ad essere il volto corretto da predire. Se due o tre di questi indici puntano alla stessa immagine la scelta è ulteriormente accreditata:

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
for i=1:soggetti
   for j=espressioniTr+1:espressioniTr+espressioniTe
      a=imread(strcat('C:\facerec\groups\by_orl\s',num2str(i),'\',num2str(j),'.pgm'));
      a=imresize(a,rsz);
      b=a(1:size(a,1)*size(a,2));
      b=double(b);
      tcoor=b*base;
      for k=1:size(allsamples,1)
         mdist(k)=norm(tcoor-allcoor(k,:));
      end
      [dist,index2]=sort(mdist);
      class1=floor((index2(1)-1)/espressioniTr)+1;
      class2=floor((index2(2)-1)/espressioniTr)+1;
      class3=floor((index2(3)-1)/espressioniTr)+1;
      if class1~=class2 && class2~=class3
         class=class1;
      elseif class1==class2
         class=class1;
      elseif class2==class3
         class=class2;
      end
      tutte=[tutte;class];
      if class == i
         accu=accu+1;
      end
   end

end
```

//
////////////////

Eigenfaces è stato creato proprio per il riconoscimento dei volti. Per questo tipo di utilizzo, Eigenfaces presenta alcuni vantaggi rispetto ad altre tecniche disponibili, come velocità ed efficienza. Utilizzare Eigenfaces è molto rapido, funzionale ed in grado di operare su moltissime facce in poco tempo. Purtroppo, questo tipo di riconoscimento dei volti ha uno svantaggio da non trascurare: problemi nel riconoscere i volti quando sono fotografati con diversi livelli di luce o angolazioni. Per far funzionare bene il sistema bisogna in qualche modo cercare di rendere uniformi le immagini sotto questi due aspetti. Il riconoscimento del viso attraverso Eigenfaces ha dimostrato di essere comunque molto preciso.
Questo algoritmo è stato ampiamente sperimentato in letteratura proprio per verificare il variare dell'accuratezza sotto determinate condizioni: una media del 96% con variazione di luce, 85%, con variazione di orientamento ed il 64% con variazione delle dimensioni. **[6]**
Per completare Eigenfaces, è stato sviluppato un altro approccio chiamato Eigenfeatures. Questa tecnica combina metriche facciali (distanza di misurazione tra le caratteristiche del viso), con l' approccio di Eigenfaces. Un altro metodo, che è strettamente in competizione con Eigenfaces è denominato Fisherfaces. Questo metodo per il riconoscimento facciale è meno sensibile alla variazione di illuminazione e di posa, rispetto ad Eigenfaces.
Un' alternativa più moderna ad Eigenfaces ed a Fisherfaces è stata denominata Active Appearance Model, che scorpora le forma della faccia dalla sua texture: si procede scomponendo l'immagine del viso con Eigenfaces dopo averla distorta per ottenere una certa forma. Ciò consente di ottenere performance migliori su diverse proiezioni del viso, anche quando il volto è inclinato. **[7]**

3.2 Implementazione del 2d-Laplacianfaces

Implementare 2DLaplacianfaces è stato un altro punto chiave del lavoro svolto al CNR. La costruzione dell'algoritmo in linguaggio Matlab è stata preceduta da uno studio approfondito dell'articolo di riferimento dei ricercatori Ben Niu, Simon Chi Keung Shiu , Qiang Yang e Sankar Kumar Pal. Questo testo non riporta alcuna riga di codice ma soltanto le formule e i passaggi matematici che illustrano l'idea alla base del *"2D Laplacianfaces method"* per il riconoscimento dei volti.
Si è deciso di utilizzare la struttura *cell* per memorizzare comodamente tutte le immagini in input. In questo caso non è stato necessario trasformare le matrici in vettori.
La porzione di codice seguente mostra come avviene il caricamento delle immagini per l'apprendimento nella struttura cell *A{}{}.*

//
////////////////

```
for i=1:l,
   for j=1:totexp,
      if(train(j,1)==1),
         [img_name] = sprintf('%s/s%d/%d.%s',path,i,j,ext);
         A{1}{(i-1)*k+e}=imread(img_name);
         A{1}{(i-1)*k+e}=imresize(A{1}{(i-1)*k+e},rsz);
         e=e+1;
       end
   end
e=1;
```

```
end
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Nella fase appena descritta si noti che una foto prima di essere caricata in una cella di *A{}{}* viene ridimensionata da *imresize* di un fattore *rsz* che è possibile modificare in base alle dimensioni dell'immagine originale per ottenere il formato desiderato. Nei nostri tests abbiamo utilizzato quel valore massimo di *rsz* che consentisse di processare più agevolmente le immagini ma senza perdere molti punti percentuali di accuratezza rispetto all'immagine non ridimensionata. Nel ciclo *for* più interno è possibile aggiungere, ove necessario, alcune chiamate a funzioni aggiuntive quali *rgb2gray,* nel caso il dataset non fosse in scale di grigi, *imadjust* e *histeq* per regolare intensità e contrasto.
A questo punto bisogna costruire la matrice di similarità *S* con le sole *k* immagini di training a disposizione per ciascun soggetto: la matrice che vogliamo ottenere è diagonale a blocchi, la dimensione dei blocchi varia al variare di *k*; come indicato dagli autori del metodo *S* conterrà $\exp(-\|A_i - A_j\|^2/t)$ se A_i e A_j corrispondono allo stesso soggetto, 0 altrimenti.

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

```
for i=1:l*k,
  for j=1:l*k,

    if (floor((i-1)/k)==floor((j-1)/k)),
      S(i,j)= exp((-norm(double(A{1}{i})-   double(A{1}{j}),2)^2)/t);
    else
      S(i,j)=0;
    end
  end
end
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

La matrice *S* così costruita permette ora di crearne un'altra, *D*, ottenuta dalla somma delle righe di *S.* D è una matrice diagonale a blocchi di dimensioni N x N, i cui elementi diagonali sono d_{ii},
con $d_{ii} = \sum_j S_{ij}$, che corrispondono alle somme dei valori di similarità di tutte le immagini campione j con la i-sima immagine.
A questo punto possiamo utilizzare *S* e *D* per ottenere la matrice laplaciana *L* secondo la formula L = S-D.

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
for i=1:k*l,
  for j=1:k*l,

    D(i,i)=sum(S(i,:));

  end
end

L=D-S;
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Una volta ottenuta L si può procedere alla risoluzione dei passaggi, già descritti nel cap. 1, necessari per ricavare tutti i vettori X che saranno utilizzati nella fase di predizione per proiettare le immagini di training e di test.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////

m=size(A{1}{1},1);
n=size(A{1}{1},2);

for i=1:l*k, AT{1}{i}=A{1}{i}'; end
for i=1:l*k, AA{i}{1}=A{1}{i}; end

ATL=cell(1,k*l);
for i=1:k*l,
   ATL{1}{i}=zeros(n,m);
   for j=1:k*l
      ATL{1}{i} = double(ATL{1}{i})+double(AT{1}{j})*L(j,i);
   end
end

G=zeros(n,n);

for i=1:k*l,
   G=G+double(ATL{1}{i})*double(AA{i}{1});
end

for i=1:k*l,
   ATL{1}{i}=zeros(n,m);
   for j=1:k*l,
      ATL{1}{i} = double(ATL{1}{i})+double(AT{1}{j})*D(j,i);
   end
end
H=zeros(n,n);
for i=1:k*l,
   H=H+double(ATL{1}{i})*double(AA{i}{1});
end

[X, Lambda]=eig(G,H);

////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

A questo punto non resta che caricare le pose destinate al test e proiettarle lungo i vettori X ottenuti nel passaggio precedente.
Il volto proiettato lungo quel vettore X tale che la distanza con la proiezione di un volto del training-set lungo lo stesso X sia minima sarà così assegnato alla sua relativa classe di appartenenza.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////

for i=1:l,
```

```
   for j=k+1:totexp,
   [img_name] = sprintf('%s/s%d/%d.%s',path,i,j,ext);
   B=imread(img_name);
   B=imresize(B,1);
   dist=[];
   for z=1:k*l, dist(z)=norm(double(A{1}{z})*X(:,1:3)-double(B)*X(:,1:3)); end
   [bogus, imin]=min(dist);
   flooransw(i,c)=floor((imin-1)/k)+1;
   c=c+1;
   end
    c=1;
end

////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

3.3 Implementazione del MultiReGECFaces

MultiReGEC non nasce specificamente come riconoscitore di volti; apportando alcune modifiche al classificatore è stato possibile utilizzarlo anche per tale scopo: MultiReGECFaces riceve in input una matrice, le cui righe corrispondono ai valori singolari dei volti del training-set con il relativo vettore delle etichette di classe ed un'altra matrice di immagini con relative etichette per il test.

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////

[class,acc]=multiregecfaces(data(train,:)',label(train,1),data(test,:)',
label(test,1),sigma,delta);

////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Con le sole informazioni del training-set genera le equazioni degli iperpiani regolarizzandole con il sistema uno contro tutti come mostra la parte di codice seguente:

```
////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////

for i=1:classi;
   c=0;
   for j=1:classi;
      if (i~=j)
         c=c+1;
         APPOtrain=[train(trainL==i,:);train(trainL==j,:)];
         APPOtrain_l=[ones(sum(trainL==i),1); -    ones(sum(trainL==j),1)];
[class,acc,Z,W,T1]=regec2multi(APPOtrain,APPOtrain_l,APPOtrain,APPOtrain_l, C, sigma,delta);
         APPOpiani(:,c)=W(:,1);
      end
   end
   IPmedi(:,i)=generaIperPiani(APPOpiani');
end

////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

Ottenute le equazioni dei piani si procede al calcolo della distanza tra queste e gli elementi del test-set.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
for i=1:dimTest
  %calcola proiezioni e distanze dai piani
  appo=projTrain;
  for k=1:size(projTrain,2);
     appo(:,k)=appo(:,k)-test(i,k);
  end
  appo=appo.^2;
  distanza=sum(appo,2);
  for j=1:Classi
    % [projTest(i,:), dTest(i)]=ProjectPoint(test(i,:), planes(:,j));
     distaTestTrain(i,j)=min(distanza(trainL==j));
  end
  [bogus, testL(i)]=min(distaTestTrain(i,:));
end
testL=testL';

////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

In questo modo, gli elementi del test più vicini al segmento di iperpiano che rappresenta una singola classe verranno attribuiti proprio a quella classe attraverso un' etichetta. Quindi vengono confrontate le etichette ottenute nella fase di predizione con quelle fornite in input per valutare l'accuratezza ottenuta dall'algoritmo:

```
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
acc=sum(class==test_l)/size(class,1);
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////
```

3.4 Descrizione dei datasets

I datasets di volti che abbiamo utilizzato per i tests sono in tutto cinque. Due di questi sono stati ottenuti, come verrà specificato in seguito, da sottoinsiemi di un dataset più grande, il Feret Database. I restanti 3 datasets sono Orl, Indians e IMM Face Database.

3.4.1 Il database Feret

Questo dataset, a differenza degl'altri 3 utilizzati, è protetto, il che significa che non è scaricabile liberamente dalla rete; è stato necessario inoltrare una richiesta di download al National Institute of Standards and Technology per ricevere un account temporaneo di accesso via email.
Lo scopo del programma Feret è stato quello di costruire un grande insieme di immagini di volti raccolte indipendentemente dagli sviluppatori di algoritmi di AFR. Il dott. Harry Wechsler della George Mason University è stato scelto per dirigere la raccolta delle immagini. La costruzione del database è stata frutto della collaborazione tra il dott. Wechsler ed il dott. Phillips. Le immagini sono state raccolte in un ambiente semicontrollato. Per preservare un certo grado di coerenza in tutto il database sono state utilizzate le stesse impostazioni in ogni sessione di fotografia. Dato che l'apparecchio utilizzato doveva essere riassemblato ad ogni

sessione, vi è stata qualche piccola variazione nelle immagini raccolte in date diverse. Il FERET è stato costruito in 15 sessioni tra agosto '93 e luglio '96. La banca dati contiene 1564 insiemi di immagini per un totale di 14.126 immagini che comprendono 1199 persone e 365 gruppi di immagini replicate. Per alcune persone sono trascorsi più di due anni tra la prima e l'ultima sessione fotografica, e molti soggetti sono stati fotografati più volte. Questo lasso di tempo è stato importante che trascorresse poichè ha permesso ai ricercatori di studiare, per la prima volta, il cambiamento che si verifica in un soggetto nel corso di un anno. **[8]**

3.4.2 Il database IMM

Il database IMM è composto da 240 immagini di 40 differenti individui, tutti senza occhiali. La distribuzione tra i sessi è di 7 donne e 33 uomini. Le immagini sono state acquisite nel gennaio 2001, utilizzando il formato 640 x 480 JPEG con una videocamera DV Sony, DCR-TRV900E PAL. Le immagini sono state raccolte in base a sei tipologie differenti:

1. Faccia frontale, espressione neutra, luce diffusa.
2. Faccia frontale, espressione "felice", luce diffusa.
3. Faccia ruotata di ca. 30 gradi a destra rispetto ai volti frontali, espressione neutra, luce diffusa.
4. Faccia ruotata di ca. 30 gradi a sinistra, espressione neutra, luce diffusa.
5. Faccia frontale, espressione neutra, luce posizionata a sinistra della persona.
6. Faccia frontale, espressione arbitraria, luce diffusa. **[9]**

3.4.3 Il database Indians

Questo database contiene immagini di 40 diversi soggetti con undici diverse pose per ogni individuo catturate nel febbraio 2002 nel campus Indian Institute of Technology di Kanpur. Tutte le immagini hanno uno sfondo brillante ed omogeneo ed ogni soggetto è stato posizionato in piedi frontalmente rispetto alla macchina fotografica. Per ogni individuo, sono state utilizzate le seguenti pose: guardando avanti, guardando a sinistra, guardando a destra, guardando in alto, guardando in alto verso sinistra, guardando in alto verso destra, guardando verso il basso. In aggiunta alla variazione di posa, sono state incluse quattro espressioni emozionali: neutro, sorridente, ridente, triste / disgustato. A titolo di esempio, le immagini corrispondenti ad una singola persona sono riportate di seguito nella Figura 4.

Figura 4. Esempio di un individuo nel database Indians

I files sono in formato JPEG. La dimensione di ogni immagine è 640x480 pixels, con 256 livelli di grigio per pixel. Tutte le foto sono organizzate in due principali directory una per i maschi ed una per le femmine. In ciascuna di queste cartelle vi è una sottodirectory per ogni un individuo. In ciascuna sottodirectory vi sono le undici diverse pose di un singolo soggetto e le immagini

sono etichettate nella forma n.jpg, dove n è il numero che corrisponde ad una delle undici pose registrate. **[10]**

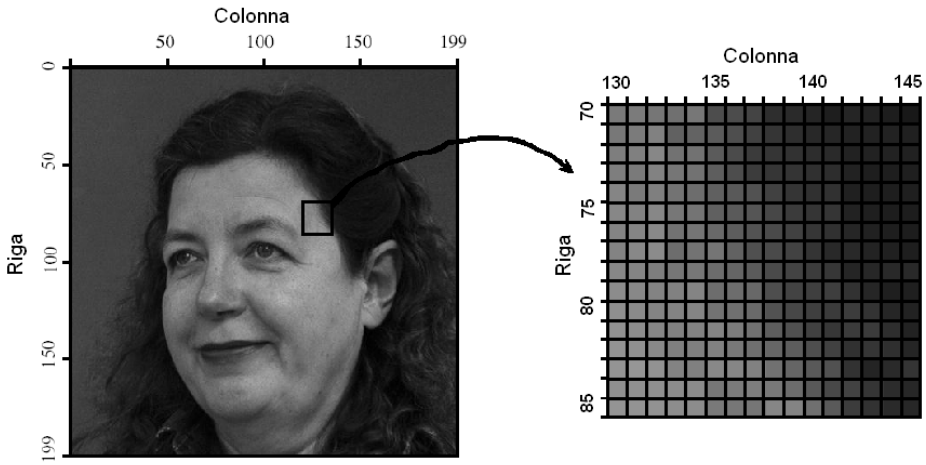
3.4.4 Il database ORL

Questo database di volti, formalmente 'The ORL Database of Faces', contiene una serie di immagini di volto scattate tra aprile 1992 e aprile 1994 presso i laboratori AT&T. Il database è stato utilizzato per il riconoscimento dei volti in un progetto realizzato in collaborazione con Speech, Vision and Robotics Group della Cambridge University -
Engineering Department.
Vi sono dieci immagini distinte per ciascuno dei 40 soggetti esaminati. Per alcuni, le immagini sono state prese in giorni differenti, variando l'illuminazione, le espressioni facciali (occhi aperti/chiusi, sorridente/non sorridente) e dettagli del viso (occhiali/senza occhiali). Tutte le immagini sono state riprese con uno sfondo scuro omogeneo e tutti i soggetti sono stati posti in piendi in posizione frontale (con una certa tolleranza di qualche piccolo movimento laterale).

I files sono in formato PGM, e possono comodamente essere visualizzati su UNIX (TM) o dai sistemi che utilizzano il programma 'xv'. La dimensione di ogni immagine è di 92x112 pixel, con 256 livelli di grigio per pixel. Le immagini sono organizzate in 40 cartelle (una per ogni soggetto), che hanno nomi in forma sx, dove X indica l'id del soggetto (tra 1 e 40). In ciascuna di queste directory, vi sono dieci diverse immagini dello stesso individuo, e sono etichettate nella forma Y.pgm, dove Y è il numero (tra 1 e 10) identificativo della singola posa. **[11]**

3.5 Preparazione dei dati

Per codificare ed importare le immagini nel workspace abbiamo utilizzato la funzione *imread*. Questa funzione, già disponibile in Matlab, restituisce una matrice n x m nel caso in cui l'immagine abbia n pixel di altezza ed m di larghezza.



Row	150	155	160	165
50	183 183 181 184 177 200 200 189 159 135 94 105 160 174 191 196			
	186 195 190 195 191 205 216 206 174 153 112 80 134 157 174 196			
	194 196 198 201 206 209 215 216 199 175 140 77 106 142 170 186			
	184 212 200 204 201 202 214 214 214 205 173 102 84 120 134 159			
55	202 215 203 179 165 165 199 207 202 208 197 129 73 112 131 146			
	203 208 166 159 160 168 166 157 174 211 204 158 69 79 127 143			
	174 149 143 151 156 148 146 123 118 203 208 162 81 58 101 125			
	143 137 147 153 150 140 121 133 157 184 203 164 94 56 66 80			
	164 165 159 179 188 159 126 134 150 199 174 119 100 41 41 58			
	173 187 193 181 167 151 162 182 192 175 129 60 88 47 37 50			
60	172 184 179 153 158 172 163 207 205 188 127 63 56 43 42 55			
	156 191 196 159 167 195 178 203 214 201 143 101 69 38 44 52			
	154 163 175 165 207 211 197 201 201 199 138 79 76 67 51 53			
	144 150 143 162 215 212 211 209 197 198 133 71 69 77 63 53			
	140 151 150 185 215 214 210 210 211 209 135 80 45 69 66 60			
65	135 143 151 179 213 216 214 191 201 205 138 61 59 61 77 63			

Figura 5. Esempio di immagine trasformata nella matrice dei valori corrispondenti alle gradazioni di grigio

Ogni elemento corrisponde alla tonalità di grigio del pixel corrispondente e quindi varia tra 0 e 255.
A seconda del dataset, abbiamo incontrato immagini di formati ed estensioni differenti. In tutti i tests abbiamo cercato di mantenere standard il formato delle immagini (90x90pxl circa) tramite la funzione *imresize.* In alcuni casi l'utilizzo delle funzioni *imadjust* e *histeq* ha contribuito ad incrementare di qualche punto percentuale il tasso di accuratezza medio poichè, rispettivamente, aiutano a regolare intensità e contrasto.
Ogni matrice, corrispondente ad una singola immagine, è stata inserita per comodità in una struttura particolare del matlab denominata *cell.* Questa struttura è utile perchè permette di gestire le matrici attraverso l'indicizzazione diretta: un elemento della struttura *cell* di riga *i* e di colonna *j* che in una matrice standard corrisponde ad un singolo valore, in questa struttura invece conterrà un'intera matrice di valori che, nel nostro caso, rappresenta la codifica di un' immagine di una persona.
Gli algoritmi EigenFaces e MultiRegecFaces operano solo con i vettori quindi è stato necessario accodare tutte le righe di ogni singola matrice corrispondente ad un' immagine ottenendo così un vettore che rappresenta una singola foto e non più una matrice come mostra la Figura 6.

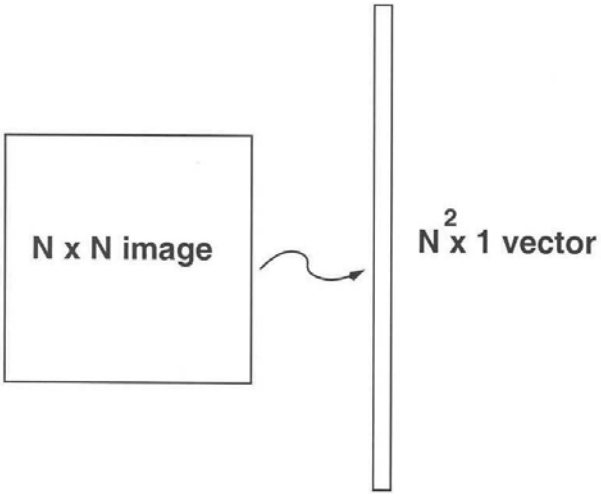


Figura 6. Rappresentazione grafica della trasformazione da matrice a vettore.

Per ottenere questo risultato abbiamo utilizzato in sequenza *squeeze*, *shiftdim* e *reshape* anche queste funzioni per la manipolazione di matrici già presenti in Matlab.

3.6 Estrazione delle caratteristiche nei tests

L'estrazione delle caratteristiche dai volti è stata uno dei punti chiave dell'intero lavoro svolto. Nel riconoscimento dei volti, così come in molti altri problemi di classificazione, è di fondamentale importanza cercare di ridurre quanto più possibile lo spazio di osservazione dei dati. Nel nostro caso abbiamo riscontrato effetti differenti sulla predizione, al variare dei dati e dei metodi applicati. Una corretta estrazione delle caratteristiche puo' provocare un miglioramento complessivo dell'accuratezza: in un volto le caratteristiche salienti possono essere rappresentate dagli occhi, dalle orecchie, dal naso. Abbiamo provato ad utilizzare alcune tecniche automatiche per l'estrazione delle caratteristiche appena elencate e ad effettuare il riconoscimento solo su questi tratti ma i risultati che abbiamo ottenuto non sono stati soddisfacenti come pensavamo. Queste tecniche infatti richiedono uno sforzo eccessivo per dare buoni risultati, non avendo avuto a disposizione il tempo necessario per poterle regolare opportunamente sulle nostre immagini per selezionare correttamente i tratti richiesti, abbiamo deciso di non utilizzarle. In un secondo momento abbiamo provato a ritagliare soltanto il volto (escludendo quindi lo sfondo, il collo ed i capelli) da un sottoinsieme di immagini di un dataset ed a centrare le immagini in base a bocca e naso manualmente, avvalendoci di un programma base di fotoritocco. Abbiamo constatato che i pixels eliminati dal ritaglio provocavano ancora un abbassamento dell'accuratezza generale e così si è deciso di procedere per altre vie: abbiamo sperimentato, tra le numerose tecniche già esistenti, la SVD (Singular Value Decomposition). SVD ci ha permesso di ridurre notevolmente lo spazio occupato dai singoli volti per l'analisi senza incidere troppo sull'accuratezza.

3.7 Risultati sperimentali

In questa sezione verranno valutati i risultati sperimentali ottenuti utilizzando Eigenfaces, il 2D Laplacianfaces e MultiReGECFaces su tre databases ben noti: FERET, ORL ed INDIANS.
Il FERET è un dataset molto vasto e di conseguenza è solitamente impiegato per testare la performance degli algoritmi per il riconoscimento dei volti poichè occorrono numerosi soggetti campione, mentre i databases ORL ed INDIANS sono più indicati per valutare l'accuratezza del riconoscitore su immagini di volti riprese da differenti angolazioni, condizioni di luce ed espressioni facciali.
Nelle prossime pagine seguono quattro tabelle che riportano le accuratezze ottenute negli esperimenti. Tutti i tests sono stati effettuati in ambiente Matlab. I risultati riportati negli schemi scaturiscono dai confronti fatti con ciascun algoritmo con dati di training e di test identici eccetto che per la forma, matriciale per il 2D-Laplacianfaces e vettoriale per gli altri due metodi. Per eseguire i tests e per validarli si è proceduto facendo scorrere una "finestra" variabile di pose per il training, in modo da predire la classe di appartenenza dei volti lasciati al di fuori di tale finestra e che costituiranno quindi il test-set.
Le medie delle accuratezze ottenute interando un certo numero di volte questa operazione, sono riportate nelle colonne delle quattro tabelle di risultati. Ad esempio, avendo a disposizione 10 pose per ciascun individuo, una validazione con tre volti per l'apprendimento è ottenuta considerando nel primo test le prime tre pose per il training, dalla seconda alla quarta posa come training-set del secondo test, dalla terza alla quinta per il terzo e così via fino ad utilizzare le ultime tre immagini per l'apprendimento e le restanti sette per il test.

Algoritmo	**Numero di volti utilizzati per ciascun individuo per l'addestramento**					
	1	2	3	4	5	6
Eigenfaces	15%	32%	41%	41%	48%	53%
2D LaplacianFaces	5%	15%	71%	85%	84%	83%
MultiRegecFaces	31%	50%	63%	71%	65%	63%

Dataset: Feret Frontali (40 soggetti / 7 espressioni)

Percorso locale: **C:\facerec\groups\by_feret_frontali**

Formato originale: **256x384**

Ridimensionamento: **40%**

Algoritmo	Numero di volti utilizzati per ciascun individuo per l'addestramento					
	1	2	3	4	5	6
Eigenfaces	33%	47%	53%	53%	53%	35%
2D LaplacianFaces	13%	20%	59%	53%	48%	67%(85%)
MultiRegecFaces	33%(34%)	48%	51%(63%)	51%	53%	67%(90%)

Dataset: Feret 7 pose (40 soggetti / 7 espressioni)

Percorso locale: **C:\facerec\groups\by_feret**

Formato originale: **256x384**

Ridimensionamento: **40%**

Algoritmo	Numero di volti utilizzati per ciascun individuo per l'addestramento									
	1	2	3	4	5	6	7	8	9	10
Eigenfaces	46%	44%	48%	53%	55%	60%	65%	73%	73%	84%
2D LaplacianFaces	16%	58%	69%	59%	31%	74%	75%	82%	75%	82%
MultiRegecFaces	44%(55%)	45%	41%	44%	52%	61%	79%	66%	75%	66%

Dataset: Indians (32 soggetti / 11 espressioni)

Percorso locale: **C:\facerec\groups\by_indians**

Formato originale: **128x96**

Ridimensionamento: **100%**

Algoritmo	Numero di volti utilizzati per ciascun individuo per l'addestramento								
	1	2	3	4	5	6	7	8	9
Eigenfaces	68%	75%	79%	85%	88%	91%	94%	94%	93%
2D LaplacianFaces	8%	84%	88%	91%	93%	94%	95%	94%	98%(100%)
MultiRegecFaces	66%	75%	80%	81%	87%(90%)	93%	95%	98%	98%(100%)

Dataset: ORL (40 soggetti / 10 espressioni)

Percorso locale: **C:\facerec\groups\by_orl**

Formato originale: **92x112**

Ridimensionamento: **100%**

Conclusioni

Il lavoro svolto al CNR ha prodotto certamente risultati validi ed incoraggianti. Le operazioni che si possono effettuare sulle immagini di un volto, per l'estrazione delle sue caratteristiche, sono tuttavia numerose e differenti tra loro ed è quindi sempre possibile ipotizzare nuove strade che consentano un tasso di riconoscimento sempre più accurato. Un altro dei propositi di questo elaborato, oltre a quello di evidenziare le differenze tra i diversi metodi analizzati, è stato quello di mettere in luce come sia possibile trasformare un' immagine ad alta risoluzione in una sequenza numerica di pochi elementi utili però a rappresentare anche una classe complessa come quella del volto di essere umano.

In definitiva è possibile ottenere il riconoscimento automatico di un volto, con una discreta accuratezza e con poche informazioni in input, attraverso diversi criteri di classificazione. Ogni metodo analizzato ha terminato l'esecuzione dei tests, su 40 individui differenti, con tempi di completamento piuttosto accettabili: circa 30 secondi, nel caso di poche immagini per l'apprendimento, pochi minuti con oltre sette immagini di training. I tre algoritmi hanno mostrato limiti e qualità differenti che potranno essere sfruttate a proprio vantaggio in base alle caratteristiche specifiche del dataset da analizzare a disposizione:
Eigenfaces è risultato certamente il metodo più rapido in termini di prestazioni ma con percentuali di accuratezza non molto elevate rispetto agli altri due; il 2D-LaplacianFaces si è dimostrato un ottimo riconoscitore ma non nel caso in cui si dispone di una sola immagine di ciascun individuo per il training.
MultiReGECFaces ha generato risultati molto validi, in particolare in condizioni difficili: ad esempio in presenza di occlusioni nell'immagine, che sono state applicate in alcuni tests artificialmente utilizzando rettangoli neri per simulare la presenza di elementi di "disturbo" (come una sciarpa o un cappello), ha superato sempre di oltre 10 punti percentuali le accuratezze prodotte da EigenFaces e dal 2D-LaplacianFaces.

Il codice implementato, e l'insieme di procedure utili alla preparazione delle immagini, saranno a breve resi disponibili in rete all'indirizzo *http://www.utenza.it* per poter essere maggiormente analizzati e quindi migliorati in prestazioni ed affidabilità.

Bibliografia

[1] Elisa Frigerio, *"Psicologia delle Emozioni"* **-** lezioni di Psicologia delle Emozioni presso l'Università degli Studi di Pavia, Maggio 2008.

[2] Anya Hurlbert, *"Trading faces",* Nature Neuroscience vol. 4 n°1, Gennaio 2001.

[3] Carlo Vercellis, *"Business intelligence"*, McGraw-Hill pp. 212-216, 2006.

[4] Ben Niua, Qiang Yangb, Simon Chi Keung Shiua, Sankar Kumar Palc, *"Two-dimensional Laplacianfaces method for face recognition",* Pattern Recognition vol. 41, 2008

[5] M.R. Guarracino, C. Cifarelli, O. Seref, P. Pardalos, *"A Classification Method Based on Generalized Eigenvalue Problems"*, Optimization Methods and Software, vol. 22, n. 1 Pages 73-81, 2007.

[6] Turk & Pentland, *"Face recognition using eigenface",* p. 590, 1991

[7] T. F. Cootes, G. J. Edwards e C. J. Taylor, *"Active appearance models"* IEEE TPAMI, 23(6): pp. 681–685, 2001

[8] Feret Database: P.J. Phillips, H. Wechsler, J. Huang, P. Rauss,
"*The FERET database and evaluation procedure for face recognition algorithms*", Image and Vision Computing J, Vol. 16, No. 5, pp. 295-306,1998.
P.J. Phillips, H. Moon, S.A. Rizvi, P.J. Rauss,
"*The FERET Evaluation Methodology for Face Recognition Algorithms*", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 22, pp. 1090-1104, 2000.

[9] Imm Database of Faces: M. B. Stegmann, B. K. Ersbøll, and R. Larsen. FAME - a flexible appearance modelling environment. *IEEE Trans. on Medical Imaging*, 22(10):1319-1331, 2003.

[10] The Indian Face Database: Vidit Jain, Amitabha Mukherjee,
http://vis-www.cs.umass.edu/~vidit/IndianFaceDatabase , 2002.

[11] ORL Database of Faces: AT&T Laboratories Cambridge - Aprile 1992 e Aprile 1994.