Cristina Segalin
Massimo Gambin

Sistema di rilevamento automatico e riconoscimento volti: aspetti metodologici e pratici

Tesi di Laurea Triennale

Università di Verona

Dicembre 2010

Università di Verona
Facoltà di Scienze MM.FF.NN

Corso di Laurea Triennale in INFORMATICA
Indirizzo INFORMATICA MULTIMEDIALE

Sistema di rilevamento automatico e riconoscimento volti: aspetti metodologici e pratici

Tesi di

Cristina Segalin e Massimo Gambin

Relatore

Prof. Umberto Castellani .....

Controrelatore

Prof. Marco Cristani .....

Candidato

[VR076481] Cristina Segalin .....

[VR076757] Massimo Gambin .....

Sessione Laurea 14 Dicembre 2010
Anno Accademico 2009/2010
Consultazione consentita

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

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 in cui riconosciamo le parole quando leggiamo.

Il riconoscimento di un volto non viene 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.

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 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 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 facce che si incontrano. Ma è stato osservato che questo prototipo non è fisso, si può adattare. Quando il mo-

dello 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. "É 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 antiimmagine 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 (Auto-

matic 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.

Data l'importanza di questo tipo di attività nella vita di tutti i giorni, sembra che l'evoluzione abbia guidato noi esseri viventi a sviluppare perciò un sistema altamente efficace e flessibile.
Il riconoscimento facciale fa riferimento ad un ambito di ricerca molto più vasto, che studia i *sistemi biometrici*. Questi sistemi sono in grado di verificare o riconoscere l'identità di un individuo in base a una caratteristica fisiologica o comportamentale. Oltre al riconoscimento del volto, il mondo della *biometria*, comprende anche altri tipi di discipline quali ad esempio il riconoscimento delle impronte digitali, del colore e della dimensione dell'iride, della retina, della sagoma della mano o della forma dell'orecchio. Tutte queste caratteristiche appartengono all'insieme delle caratteristiche fisiologiche. Tuttavia i sistemi biometrici sono anche in grado di operare riconoscimenti di caratteristiche di tipo comportamentale, quali ad esempio l'impronta vocale, la scrittura grafica, lo stile della battitura su di una tastiera, la firma di una persona o i movimenti del corpo.

Scopo della tesi è lo sviluppo di un sistema automatico per il riconoscimento di volti.
Il riconoscimento di volti è un ambito molto interessante ed affascinante della visione computazionale che ha ricevuto negli ultimi anni sempre maggiore attenzione. Ciò è dimostrato, ad esempio, dalla nascita di conferenze dedicate esclusivamente ad esso. Questo grande interesse è dovuto principalmente alle molte applicazioni in ambito commerciale e di sicurezza che il riconoscimento automatico di volti può avere.

Sebbene esistano diverse tecniche biometriche per il riconoscimento di esseri umani, alcune delle quali molto efficaci (ad esempio la lettura delle impronte digitali), il riconoscimento basato su immagini del volto è oggetto di particolare interesse per la sua estrema flessibilità e poca invasività.
D'altra parte, il riconoscimento di volti in contesti non controllati rappresenta una sfida piuttosto complessa, data l'estrema variabilità dei soggetti e la difficile modellizzazione del problema. Le tecniche e gli algoritmi proposti per questo scopo sono molti e di varia natura; alcuni di essi hanno raggiunto ottimi livelli di efficacia e robustezza[12].
Ciò che ha motivato il nostro lavoro non è stato tanto lo studio di nuove tecniche o algoritmi per il riconoscimento dei volti, quanto l'idea di creare un sistema semplice e realmente utilizzabile che lavorasse su una sequenza d'immagini acquisita in tempo reale (real time).
Se da un lato l'obiettivo di arrivare ad un sistema real time completo e funzionante può essere stato un limite nello studio di tecniche innovative o maggiormente sofisticate, dall'altro ha prodotto una piattaforma ideale sulla quale valutare l'efficacia applicativa di nuovi metodi di apprendimento statistico. Il funzionamento in tempo reale consente infatti di effettuare un grande numero di test in breve tempo mettendo in luce gli aspetti critici che richiedano ulteriori fasi di ricerca.
Il sistema implementato, a partire dalla memorizzazione di immagini campione di una o più persone, è in grado di verificare l'identità delle stesse in momenti successivi. Grazie a tecniche base di visione computazionale il sistema ottenuto si adatta alle condizioni ambientali in cui viene installato. Il software è corredato di una semplice interfaccia che permette la registrazione di nuovi utenti (inserimento dati e acquisizione di un insieme di immagini di training di buona qualità) e la validazione automatica di utenti registrati. In entrambe le modalità di lavoro l'elaborazione dei frame avviene in tempo reale.
Alcuni esperimenti effettuati hanno dimostrato l'efficacia del sistema in condizioni di poca variabilità (ad esempio delle condizioni di illuminazione) e con un numero piuttosto ristretto di soggetti da riconoscere. In contesti particolarmente controllati, il sistema ottenuto ha dato risultati soddisfacenti.
Lo scopo di questo progetto è di realizzare un sistema di riconoscimento facciale automatico attraverso l'uso della libreria OpenCV e utilizzarlo come sistema di accesso al laboratorio VIPS.
Si analizzeranno in particolare la libreria OpenCV, il sistema di riconoscimento facciale nelle sue diverse tecniche e in particolare quello usato per il progetto, lo sviluppo dell'applicazione, l'installazione del dispositivo nel laboratorio, i risultati ottenuti da questo esperimento.

Capitolo 1

Verso un sistema di riconoscimento

1.1 Introduzione ai Sistemi biometrici

Un **sistema biometrico** è un dispositivo automatico per la verifica di identità o l'identificazione di una persona sulla base di caratteristiche biologiche. Queste caratteristiche possono essere di varia natura e sono generalmente suddivise, come riportato nella figura seguente, in **fisiologiche** (impronta digitale, volto, mano, retina, iride, DNA, ...) e **comportamentali** (voce, calligrafia, stile di battitura, ...). Come è facilmente intuibile, i sistemi basati su caratteristiche fisiologiche sono generalmente più affidabili di quelli basati su caratteristiche comportamentali, tuttavia questi ultimi possono risultare a volte più semplici da integrare in alcune specifiche applicazioni. In figura sono riportate le principali caratteristiche biometriche.[13]

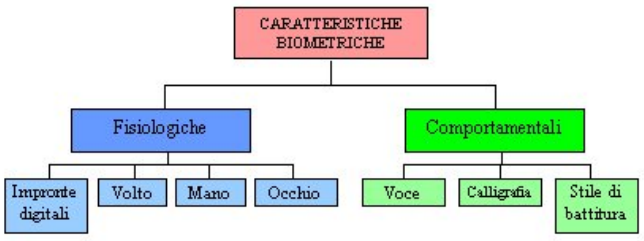

Figura 1.1: Le principali caratteristiche biometriche

Un sistema biometrico può essere generalmente impiegato per la **verifica di identità** o per l'**identificazione**:

- problema della **verifica di identità** consiste nello stabilire se un individuo è veramente colui che dichiara di essere; a tal fine l'utente

deve fornire al sistema, oltre alla caratteristica biometrica da esaminare, anche il proprio nome o un codice di identificazione personale che rappresenta la sua dichiarazione di identità.

- Il problema dell'**identificazione** consiste invece nel determinare se una persona può essere associata (corrisponde) a una di quelle presenti in un archivio (non è richiesto all'individuo di dichiarare la propria identità).

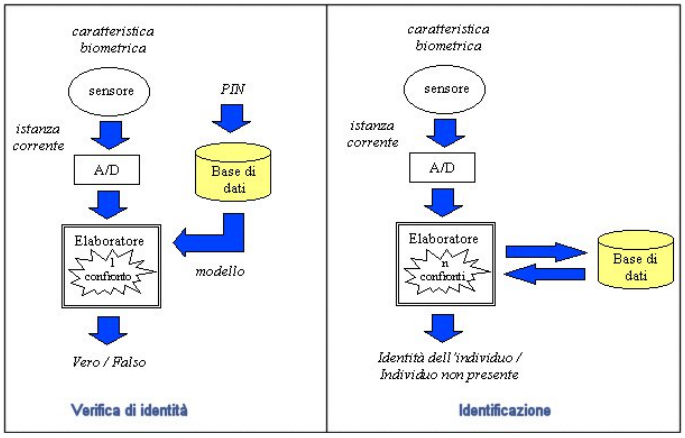

Figura 1.2: Differenza tra verifica di identità e identificazione

Affinché un utente possa utilizzare un sistema biometrico è necessario che sia sottoposto a una fase di registrazione iniziale (detta **enrolment**), durante la quale vengono acquisite una o più istanze della caratteristica biometrica; le caratteristiche numeriche da esse estratte vengono combinate in un modello che viene memorizzato nel sistema.

1.2 Introduzione alla Face Detection e Face Recognition

Face Recognition è un'area molto attiva nella Computer Vision e nel campo biometrico, poiché è stata studiata per 25 anni e sta producendo infine applicazioni per sicurezza, robotica, HCI, camere digitali, giochi e intrattenimento. *Face Recognition* in genere include due stadi:

- **Face Detection**, dove si cerca una qualsiasi faccia in una foto, mostrata qui con una rettangolo, e poi l'elaborazione delle immagini elimina l'immagine del volto per rendere più facile il riconoscimento.
- **Face Recognition**, dove la faccia rilevata e processata è confrontata con un database di facce note, per decidere chi è la persona ritratta (come mostrato dalla scritta in rosso nell'immagine)

Figura 1.3: Face Detection

Dal 2002, la Face Detection può essere realizzata in modo abbastanza affidabile come una OpenCV Face Detector, che lavora in circa il 90-95% delle foto chiare di una persona cercando nella fotocamera. Di solito è più difficile da rilevare un volto di una persona quando è visto di lato o in un angolo, e a volte questo richiede stima della posa della testa in 3D. Può anche essere molto difficile da rilevare se la foto non è molto luminosa, o se una parte del volto è più luminoso di un altro o ha ombre o è sfocata o indossando occhiali, ecc.
Tuttavia, Face Recognition è molto meno affidabile del Face Detection, in genere 30-70% di affidabilità. Face Recognition è stata un forte campo di ricerca dal 1990, ma è ancora distante dall'essere affidabile, e nuove tecniche vengono inventate ogni anno.
Mostreremo come utilizzare Eigenfaces (detta anche "Analisi delle Componenti Principali" o PCA), un metodo semplice e popolare di 2D Face Recognition da una foto, a differenza di altri metodi comuni, quali reti neurali o Faces Fisher.
Per imparare la teoria di come Eigenface opera, si dovrebbe leggere Face Recognition Con Eigenface dal Servo Magazine (aprile 2007).[14]

1.3 Il riconoscimento del volto umano

Quando si vuole riconoscere una persona, la prima cosa che si osserva è il volto; non siamo certo abituati a riconoscere le persone analizzando la impronte digitali o osservando minuziosamente la conformazione delle iridi.
Sia le informazioni visive globali che le caratteristiche locali (morfologia di occhi, naso) sono fondamentali nella percezione e nel riconoscimento del volto. Studi sul riconoscimento del volto documentano infatti che gli uomini si concentrano subito, quando questi sono presenti, su elementi predomi-

nanti come grosse orecchie, naso aquilino, occhi strabici, ecc. Si è messo in evidenza che le caratteristiche interne (naso, bocca, occhi) ed esterne (forma della testa, capelli) sono utilizzate per riconoscere volti non familiari, mentre i volti familiari sono riconosciuti principalmente grazie alle caratteristiche interne. Si è anche dimostrato che vengono ricordati più facilmente i volti attraenti, seguiti dai volti meno attraenti, ed infine i volti né belli né brutti, più difficili da identificare poiché non accompagnati da informazioni supplementari sul gradimento estetico .

Figura 1.4: Una serie di immagini frontali di volti

Interessante è anche la teoria secondo la quale gli uomini riconoscono gente della propria razza meglio che gente di razze diverse. Questo si può spiegare con il fatto che gli uomini codificano nel proprio cervello i canoni di una faccia "media" della gente che sono abituati a vedere, e questa faccia media varia per razze diverse. Purtroppo ancora molte delle teorie che sono state formulate presentano punti contraddittori e si basano su insiemi relativamente ristretti di individui. Nonostante questo alcuni ricercatori hanno trovato in questi studi alcuni spunti per il progetto e la costruzione di sistemi automatici o semiautomatici basati sul riconoscimento del volto.
Queste applicazioni presentano un'ampia gamma di problematiche differenti e che richiedono soluzioni tecniche adeguate. Esistono importanti parametri per la valutazione di queste applicazioni. In qualsiasi problema di "pattern recognition" la qualità della risposta dipende direttamente dalle limitazioni che si impongono sulle immagini di input e sullo spazio di ricerca. I vincoli sulle immagini possono includere formati predefiniti, sfondi che semplificano la suddivisione dell'immagine in componenti elementari (segmentazione) e controlli di qualità dell'immagine. I vincoli sulle basi di dati possono includere limiti geografici e limiti su attributi legati all'immagine (descrittori). In generale il riconoscimento del volto richiede che il sistema fornisca in output un insieme di immagini rappresentanti il soggetto identificato. Se il sistema non è in grado di fornire queste immagini allora il soggetto "non è riconosciuto". L'individuazione di immagini simili a quelle descritte da una persona richiede che il concetto di similarità utilizzato dal sistema sia il più possibile corrispondente a quello comune dell'uomo.
Le tecniche di riconoscimento del volto, rispetto ad altre tecniche biometriche, presentano il vantaggio di non essere invasive, ovvero di richiedere poca o nulla cooperazione, non essendo soggette a eventuali modifiche di comportamento(volontarie o involontarie) da parte dell'individuo (passivo) sottoposto a riconoscimento. Grazie alla buona accettabilità da parte degli

individui, la tecnica di riconoscimento del volto (Face Recognition Technology) è diventata abbastanza popolare negli USA a partire dagli anni '90. Purtroppo ancora molte delle teorie che sono state formulate presentano punti contraddittori e si basano su insiemi relativamente ristretti di individui. Nonostante questo alcuni ricercatori hanno trovato in questi studi alcuni spunti per il progetto e la costruzione di sistemi automatici o semiautomatici basati sul riconoscimento del volto.

Capitolo 2

Sistemi automatici di riconoscimento del volto

Attraverso alcuni criteri è possibile tentare di classificare i problemi di riconoscimento automatico. L'acquisizione del volto può essere eseguita sia su una sequenza di fotogrammi, che da una immagine fissa: secondo questa distinzione i problemi si dividono in "riconoscimento statico" e "riconoscimento dinamico".
Il *riconoscimento statico* è caratterizzato da una buona qualità dell'immagine dove solitamente lo sfondo e l'illuminazione sono controllati. Inoltre, nei casi di foto relative a documenti, come patenti o passaporti si ha una buona cura nella posa del soggetto che è spesso frontale e senza particolari espressioni. Per tale motivo il problema di localizzare ed isolare il volto è relativamente semplice; le problematiche esistenti consistono nel mantenere una buona precisione anche per archivi di grandi dimensioni.
Il *riconoscimento dinamico* presenta invece sfondi e pose irregolari: talvolta questa difficoltà è in parte compensata dalla disponibilità di una sequenza di fotogrammi.

Figura 2.1: Immagine contenente un numero di volti tra cui quello da individuare

Una ulteriore distinzione la si può fare tra applicazioni che richiedono ricerche di somiglianza: si parlerà così di sistemi che svolgono confronti esatti (exact matching) e sistemi che si basano su un concetto di somiglianza.
In generale il processo di riconoscimento può essere scomposto in quattro fasi :

- **Fase di pre-elaborazione**: consiste nel garantire che l'immagine cui viene applicato il processo di riconoscimento soddisfi alcuni standard richiesti: ad esempio che il volto sia situato al centro dell' immagine e costituisca buona parte della stessa; che lo sfondo soddisfi certi vincoli, e così via. Solitamente questa fase è svolta dalle apparecchiature atte al prelevamento dell'immagine, tramite meccanismi che tendono ad impedire all'utente di fornire immagini distorte: un esempio possono essere i sensori che permettono di acquisire l'immagine solo quando il soggetto si trovi ad una distanza accettabile.

- **Fase di segmentazione o localizzazione**: consiste nella esatta localizzazione del volto o di alcune parti che lo compongono. Questa fase nasce dall'esigenza di caratterizzare, attraverso alcuni tratti caratteristici (occhi, naso, ecc.), la faccia di un soggetto. Le maggiori difficoltà nascono proprio dal fatto che le caratteristiche dell'immagine e del soggetto ripreso dipendono da un insieme di fattori:

  - difficoltà nel definire matematicamente e geometricamente cosa è una faccia: non è ancora stata data una definizione formale di cosa sia un volto, non esiste quindi un formalismo che possa indicare la strada per determinare un dato oggetto all'interno dell'immagine;
  - la distanza del soggetto dalla telecamera: la distanza dalla telecamera di un soggetto influisce decisamente sulle prestazioni del sistema di localizzazione;
  - le condizioni di illuminazione: un'immagine è strettamente influenzata dal tipo di illuminazione; se la luce è proiettata in modo uniforme tutte le parti del volto sono poste in risalto, ma se la luce non è uniforme un'immagine in cui i particolari sono poco distinguibili;
  - l'allineamento del volto con l'asse della verticale: la faccia nell'immagine può essere ruotata rispetto all'asse verticale; il sistema deve quindi essere in grado di riconoscere la rotazione ed anche di normalizzare la posizione riportandola nella direzione verticale;
  - lo sfondo dell'immagine: lo sfondo è uno dei fattori che maggiormente influenza la complessità della ricerca nell'immagine, infatti uno sfondo uniforme rappresenta una zona dell'immagine che è semplice da escludere dalla ricerca del volto, al contrario se

lo sfondo è complesso o casuale l'intera area dell'immagine deve essere attentamente analizzata.

- **Fase di normalizzazione**: in questa fase si cerca di standardizzare l'immagine rispetto alla base dei campioni contenuti nel database. Per fare ciò in generale si applicano all'immagine traslazioni lineari, rotazioni e cambiamenti di scala.

- **Fase di estrazione dei particolari**: è forse il cuore di tutto il processo di riconoscimento del volto. Un particolare è una caratteristica utile per distinguere un volto da un altro; può essere estratto dall'immagine tramite processi di varia natura. Solitamente più elevato è il numero di particolari estratti, più alta è la capacità di discernimento tra facce simili. Alcuni particolari interessanti sono, ad esempio, il colore degli occhi o dei capelli, la forma del naso o della bocca. Questi particolari vengono solitamente detti locali perché dipendono da e si riferiscono a una zona particolare e ristretta dell'immagine.

- **Fase di riconoscimento**: una volta che all'immagine è stato associato un vettore di valori, il problema del riconoscimento si riduce a problemi già ampiamente studiati in letteratura: la parte caratteristica di questi sistemi concerne dunque principalmente la modalità di estrazione dei particolari.

In funzione del tipo di applicazione, i sistemi di riconoscimento del volto devono essere progettati e realizzati a seconda del tipo di atteggiamento assunto dall'individuo, che possono essere di tre tipi:

- *cooperativo*: il soggetto è motivato a utilizzare il sistema per farsi riconoscere e accedere, attraverso appositi varchi, alle aree consentite;

- *non cooperativo*: il soggetto è distratto o comunque non si preoccupa di favorire o di ostacolare il riconoscimento;

- *ostile o reticente*: quando il soggetto si attiva per evitare il riconoscimento e assume comportamenti evasivi.

Il volto umano è composto da un complesso set di "*immagini multidimensionali*". Da un punto di vista biometrico, il riconoscimento del volto non è caratterizzato da un'elevata permanenza: le molteplici espressioni del volto, l'età, i radicali cambiamenti nel look, la presenza di occhiali, sono esempi di caratteri esteriori che possono mutare nel tempo rendendo difficoltoso il riconoscimento facciale. Le caratteristiche "non permanenti" del volto, implicano una notevole complessità di problemi tecnici da risolvere. Ciò nonostante, sono state sviluppate con successo alcune tecniche che consentono di conseguire sufficienti e pratici risultati di identificazione personale (Personal Identification) a prezzi accessibili. Oggi le tecniche di riconoscimento

del volto vengono utilizzate principalmente in modalità verifica, confrontando l'immagine del volto del dichiarante(acquisita in tempo reale) con quella pre-registrata nel sistema. In modalità identificazione l'impiego è limitato ai piccoli database.
Il riconoscimento di volti è un ambito molto interessante ed affascinante della visione computazionale e nel corso degli ultimi 10 anni ha ricevuto particolari attenzioni, come dimostra la nascita di conferenze ad esso dedicate, come AFGR e AVBPA[3]. Le motivazioni di questo crescente interesse sono da ricercare nel gran numero di applicazioni che il riconoscimento di volti ha in ambito commerciale e di sicurezza. Sebbene esistano altri metodi efficaci di identificazione biometrica, come ad esempio la lettura delle impronte digitali e la scansione della retina, questi metodi sono fortemente basati sulla cooperazione del soggetto da identificare, mentre il riconoscimento di volti può avvenire senza la cooperazione del soggetto o addirittura senza che esso se ne accorga.
Il problema può essere formulato in maniera generale nel modo seguente: *"Data una serie di immagini di una scena, identificare o autenticare una o più persone all'interno di essa"*. L'identificazione e l'autenticazione di un soggetto sono problemi leggermente diversi. Nell'identificazione si pone in ingresso al sistema l'immagine di un volto sconosciuto, e si ottiene in uscita l'identità del soggetto, scelta da un database di individui noti. Nell'autenticazione si pone in ingresso al sistema l'immagine di un volto, dichiarandone nel contempo l'identità; il sistema deve confermare o smentire l'identità dichiarata. Le varie applicazioni che coinvolgono tecniche di riconoscimento di volti pongono un'ampia serie di problematiche differenti, legate per esempio ai seguenti fattori:

- tipo di immagini che si hanno per i soggetti da riconoscere (si va da immagini statiche con posture controllate fino a video senza alcun vincolo);
- qualità delle immagini;
- dimensioni del volto rispetto allo sfondo;
- tipo di interazione da parte dell'utente;
- postura del soggetto;
- variazioni di illuminazione.

Sebbene, come vedremo in seguito, alcune di queste problematiche siano tuttora aperte, si può dire che il riconoscimento di volti sia divenuto negli ultimi 10 anni una delle applicazioni di maggior successo nell'ambito della visione computazionale.

2.1 Definizioni introduttive

In questa sezione vengono illustrate le tecniche di elaborazione di immagini e visione computazionale che si sono rivelate utili nello sviluppo del sistema. Esse riguardano l'eliminazione dello sfondo, la rilevazione di aree con pelle e l'algoritmo di riconoscimento. Prima però verranno date alcune definizioni introduttive utili alla comprensione dei paragrafi successivi.

Definizione 1: Per **acquisire un'immagine digitale** servono tre apparecchiature: una telecamera (solitamente a CCD), un Frame Grabber e un PC per il salvataggio e l'elaborazione delle immagini digitali. Il CCD (Charged Coupled Device) è una matrice con N righe ed M colonne di sensori fotosensibili, che trasforma i segnali luminosi incidenti in segnali elettrici. Il Frame Grabber si occupa di digitalizzare in una matrice N x M di numeri interi i segnali elettrici in uscita dal CCD. Questa matrice viene infine inviata ad un PC per l'elaborazione.

Definizione 2: Il **pixel** (acronimo di "picture element") è l'unità minima di informazione in uscita dal sensore della telecamera. É un elemento della matrice rappresentativa dell'immagine.

Definizione 3: Un'**immagine digitale a livelli di grigio** è rappresentata da una matrice A di numeri interi con N righe ed M colonne, corrispondente alla matrice di sensori del CCD. L'elemento $A_{i,j}$ della matrice denota il valore dell'immagine (la sua luminosità) nel pixel ij-esimo. N ed M sono due numeri interi che esprimono la dimensione di A in pixel. $A_{i,j}$, è un intero compreso fra 0 (nero) e 255 (bianco).

Figura 2.2: Esempio di immagine 3x3

Si consideri ad esempio l'immagine 3 x 3 della figura 2.2; ad essa corrisponde la matrice:

$$\mathbf{A} = \begin{pmatrix} 0 & 255 & 0 \\ 128 & 0 & 128 \\ 0 & 255 & 0 \end{pmatrix}$$

In modo molto simile un'immagine può essere pensata come un vettore, costruito con i valori della matrice letti da sinistra verso destra e dall'alto verso

il basso. Tornando all'immagine di esempio precedente, ad essa corrisponde il vettore di 9 componenti seguente:

$$V = [0 \quad 255 \quad 0 \quad 128 \quad 0 \quad 128 \quad 0 \quad 255 \quad 0]$$

Definizione 4: Una **sequenza di immagini** è una serie di N immagini, o frame, acquisite in istanti di tempo discreti $t_k = t_0 + k\Delta t$, dove t_0 è l'istante iniziale dell'acquisizione, Δ_t è un intervallo di tempo prefissato e k è un intero positivo. Δ_t vale tipicamente 1/24 secondo e viene detto velocità di acquisizione (o frame rate).

2.1.1 Rilevazione di cambiamenti tra immagini

Un importante capitolo della visione artificiale è dedicato ai metodi di *change detection* (rilevazione di cambiamenti) all'interno di sequenze di immagini. Tali metodi si occupano di studiare variazioni nel contenuto di ogni pixel in fotogrammi diversi. In questo ambito sono state proposte numerose soluzioni, più o meno complesse, a seconda che la sequenza di immagini sia acquisita da telecamere in movimento o da telecamere statiche. Noi concentreremo la nostra attenzione sul secondo caso, che meglio si adatta alla natura della nostra applicazione. Rilevare cambiamenti tra fotogrammi diversi permette di scoprire quali siano le variazioni occorse nel tempo all'interno di una scena osservata, adeguarsi a quelle definitive (la scena è cambiata), segnalando quelle durature (una persona si è posta di fronte alla telecamera), ignorando quelle temporanee (dovute al rumore o a movimenti veloci). Un metodo immediato per rilevare variazioni fra due immagini acquisite da telecamera statica consiste nell'effettuare una differenza pixel a pixel fra le due. Ogni pixel dell'immagine differenza si calcola cioè come differenza dei pixel corrispondenti nelle due immagini. A livello teorico applicando un procedimento di questo tipo ci si aspetta di ottenere valori diversi da zero solo in corrispondenza dei pixel che sono effettivamente cambiati da un'immagine all'altra. Per costruire una mappa binaria delle aree di immagine che hanno subito una variazione sarebbe dunque sufficiente considerare i pixel non nulli dell'immagine differenza. In pratica questo non è possibile, poiché sulle immagini acquisite inevitabilmente è presente del rumore, che causa una variazione nel livello di grigio dei pixel anche in assenza di cambiamenti fra le immagini. La soluzione più semplice a questo problema è quella di porre una soglia minima alla variazione del livello di grigio, creando la mappa binaria delle variazioni nel modo seguente:

$$m_{i,j} = \begin{cases} 1 & \text{se } d(i,j) > s \\ 0 & \text{se } d(i,j) < s \end{cases}$$

Dove $m_{i,j}$ e $d_{i,j}$ indicano il pixel ij-esimo rispettivamente della mappa binaria e dell'immagine differenza, mentre s indica la soglia impostata.

La soglia deve essere maggiore dell'ampiezza massima raggiunta dal rumore, ma non può essere impostata ad un valore troppo alto, per evitare che effettive variazioni nell'immagine non vengano rivelate. Essa può essere impostata empiricamente nel seguente modo:

- si fa in modo che la scena ripresa dalla telecamera sia statica, cioè non vi sia in essa alcuna variazione, se non quelle dovute al rumore;
- si cerca il valore minimo della soglia per cui i pixel della mappa binaria siano tutti nulli, ovvero in nessun pixel dell'immagine venga rilevato movimento.

Quali e quante siano le coppie di immagini da confrontare dipende dalle applicazioni e da diversi aspetti quali dimensione della memoria e velocità di esecuzione richiesta. In tipiche applicazioni di monitoraggio e sorveglianza si sceglie di mantenere in memoria un'immagine statica di riferimento, solitamente chiamata "sfondo", rispetto alla quale si confrontano i fotogrammi successivi. Questo approccio è proprio quello usato dal nostro sistema di riconoscimento.

2.1.2 Rilevazione di zone di pelle

La rilevazione automatica di zone di pelle all'interno di immagini è un'operazione comunemente usata come "primo passo" per diverse applicazioni di elaborazione di immagini legate in qualche modo alla presenza di esseri umani. La ricerca di zone di pelle pixel per pixel può essere effettuata molto rapidamente ed ha il grosso vantaggio di ridurre sensibilmente le aree di ricerca per successivi, più sofisticati algoritmi di classificazione [15] [16] [17]. La tecnica utilizzata per rilevare zone di pelle è molto semplice ed è stata tratta da un articolo di Brand e Mason [18]. Nel loro lavoro Brand e Mason sperimentarono diverse tecniche per individuare la pelle umana in un'immagine. Una di esse, proposta da Igawa [19], considera semplicemente il rapporto fra il livello di rosso (R) e il livello di verde (G) che costituiscono il colore. Igawa osservò che il colore della pelle umana contiene sempre un significativo livello di rosso. Normalizzando questo valore (ad esempio dividendo per il livello di verde) si può ottenere una buona indicazione della presenza di pelle. Wark [21] in seguito mostrò che filtrare i pixel di un'immagine a seconda del loro rapporto R/G, accettando solo quelli tali che:

$$L_i < R/G < L_s$$

poteva essere un buon metodo per individuare la pelle umana. Brand e Mason condussero esperimenti con oltre 12000 immagini del database Compaq

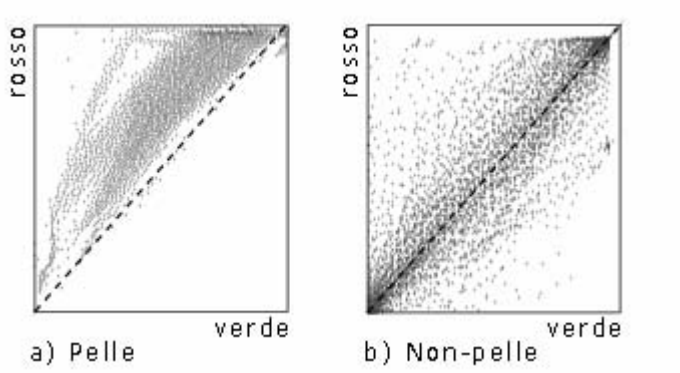

Figura 2.3: Grafici del livello di rosso vs. il livello di verde per: a) immagini di pelle b) immagini di non-pelle

[22] contenenti o non contenenti pelle umana. La figura 2.3 mostra due grafici in cui ogni punto rappresenta un pixel; sulle ascisse il livello di verde e sulle ordinate il livello di rosso.
Il grafico di sinistra è per i pixel provenienti da immagini di pelle del database Compaq, mentre quello di destra per i pixel di immagini non contenenti pelle. Possibile notare come i pixel color pelle abbiano un rapporto R/G ristretto ad un intervallo con limite inferiore pari a 1. La tecnica di filtrare i pixel a seconda del rapporto R/G si dimostra quindi molto efficace per individuare la pelle umana, sebbene - come si può intuire dal grafico di destra - abbia il difetto di accettare numerosi pixel di "non-pelle". Gli esperimenti di Brand e Mason dimostrarono che ben il 94.7% dei pixel provenienti da immagini di pelle umana veniva accettato, ma insieme ad esso anche il 35.6% dei pixel provenienti da immagini non contenenti pelle. Questa tecnica è comunque un buon compromesso fra efficacia e velocità di esecuzione e per questo è stata scelta per il sistema di riconoscimento.

2.2 Rilevazione di volti all'interno di immagini

La rilevazione di un volto all'interno di un'immagine è la prima, fondamentale, operazione da svolgere per il suo successivo riconoscimento.
Il problema assume un sapore diverso a seconda che l'analisi venga svolta su una singola immagine (nel qual caso il rilevamento è interamente basato su caratteristiche di forma, colore o tessitura) o su sequenze di immagini (nel qual caso è importante e utile l'ausilio fornito dalla componente del moto).
Qui diamo una breve panoramica dei principali contributi al rilevamento di volti all'interno di immagini singole. Fino a metà degli anni '90 la maggior parte del lavoro su questo aspetto si concentrò sulla rilevazione di un singolo volto in presenza di uno sfondo semplice, oppure complesso.
Gli approcci in questo senso erano basati su confronto con campioni, rile-

vazione delle zone color pelle e reti neurali. Negli ultimi anni sono stati proposti metodi più efficaci per rilevare diversi volti all'interno di immagini con sfondi complessi, funzionanti anche in presenza di parziali occlusioni dei volti e di rotazioni. In [30] Sung e Poggio usarono un approccio basato su apprendimento da esempi.
In [31][32] Kanade e altri usarono un approccio basato su reti neurali.
Più recentemente Osuna, Freund e Girosi [33], usarono una Support Vector Machine (SVM) per questo scopo. Per altri algoritmi usati per la rilevazione di volti si veda [34] [35] [36][37] [38][39], mentre per una valutazione generale delle tecniche si veda [40].

2.3 Estrazione di caratteristiche del volto

L'estrazione di caratteristiche è una operazione chiave sia nella rilevazione di volti che nel riconoscimento di essi. Per un'analisi dettagliata dell'argomento si veda [41], qui si vedranno solo alcune tecniche rappresentative.
Un rilevatore di simmetrie è usato in [42] per trovare gli occhi e la bocca in un volto. La motivazione di questo approccio deriva dalla pressoché simmetrica natura del volto intorno ad una linea passante per il naso. Un approccio puramente statistico per rilevare e riconoscere gli occhi umani in un'immagine di un volto ripreso frontalmente è descritto in [43].
In quel caso si usano modelli di occhi creati con due zone di livello di grigio costante; una è la regione dell'iride, mentre l'altra è la regione bianca che lo racchiude. In [44] viene proposto un approccio basato sulla rilevazione di bordi per estrarre con precisione forme bidimensionali da un'immagine.

2.4 Risultati psicofisici e neuroscientifici rilevanti per il riconoscimento di volti

Diversi studi nell'ambito della psicologia e della neuroscienza sui fattori importanti per il riconoscimento di volti da parte del cervello umano possono avere una diretta rilevanza per i ricercatori interessati a sviluppare algoritmi e sistemi per il riconoscimento di volti umani da parte di un computer.

- **Il riconoscimento di volti è un processo dedicato?**
  L'evidenza che il riconoscimento di volti da parte del cervello umano sia un processo dedicato viene da tre fonti:

  1. I volti vengono ricordati più facilmente da parte degli esseri umani rispetto ad altri oggetti quando vengono visti frontalmente;

2. Pazienti affetti da prosopagnosia [1] non sono in grado di riconoscere volti (anche familiari), ma solitamente non soffrono di altri tipi di agnosia [2] in modo rilevante. Essi riconoscono le persone dalla voce, dal colore dei capelli o dai vestiti, ma non dal volto. Sebbene essi riescano a percepire gli occhi, il naso, la bocca, i capelli, ecc., non riescono a "mettere insieme" queste caratteristiche per lo scopo di riconoscere un volto. È interessante notare come i pazienti affetti da prosopagnosia riescano a distinguere un volto da altri oggetti, ma non riescano ad identificarlo.
3. I neonati hanno una naturale propensione ad essere attratti dai volti umani. In particolare sembrano preferire stimoli visivi che abbiano la forma di un volto piuttosto che forme diverse.

Recenti studi su questo argomento hanno ulteriormente confermato che il riconoscimento di volti sia un processo dedicato, diverso dal riconoscimento di oggetti generici [22].

- **Intero volto o caratteristiche locali?**
  In [23] si dichiara che entrambi i tipi di informazione, globali e locali, vengano usate per il riconoscimento del volto umano. Alcuni studi suggeriscono che una descrizione globale del volto serva come primo passo per una successiva, più fine, percezione delle caratteristiche locali. Se una forte caratteristica locale è presente per il volto (ad esempio un grosso naso), la descrizione globale può anche non essere usata per il riconoscimento. Una prova del fatto che anche una descrizione globale del volto sia utile per il riconoscimento, viene da esperimenti effettuati con immagini di volti "invertiti". In queste immagini la bocca e gli occhi vengono scambiati, stravolgendo così la naturale struttura del volto. Si è dimostrato che in queste condizioni il volto viene riconosciuto con maggior difficoltà, sebbene le caratteristiche locali siano le stesse, solo poste in una struttura insolita.

- **Ordine di importanza delle caratteristiche locali**
  È stato dimostrato che i capelli, il profilo del volto, gli occhi e la bocca (non necessariamente in questo ordine), siano caratteristiche locali importanti per il riconoscimento di un volto. Alcuni studi hanno invece mostrato come il naso giochi un ruolo quasi insignificante nel riconoscimento di volti visti frontalmente, mentre assuma una certa importanza nei volti visti di profilo. È stato inoltre dimostrato che la parte superiore del volto sia maggiormente utile per il riconoscimento rispetto a quella inferiore.

- **Il ruolo della frequenza spaziale**
  Alcuni vecchi studi [24][25] conclusero che le informazioni a bassa frequenza spaziale giocassero un ruolo dominante nel riconoscimento di

volti. Studi successivi [26] hanno mostrato che le basse e le alte frequenze spaziali giochino ruoli differenti a seconda dell'applicazione. Ad esempio le basse frequenze sono dominanti nel riconoscere il sesso di una persona dal volto, mentre le alte frequenze sono più importanti per la sua identificazione. Riassumendo si può dire che le basse frequenze spaziali siano maggiormente utili per riconoscere caratteristiche globali del volto, mentre le alte frequenze giochino un ruolo più importante nel riconoscimento di caratteristiche locali più fini.

- **Il riconoscimento dipende dalla direzione di visualizzazione del volto?**
  Mentre per oggetti generici il dibattito se il riconoscimento sia invariante per cambiamenti del punto di vista è ancora aperto, ciò non vale per il riconoscimento di volti. Sembra che per i volti la memoria sia fortemente dipendente dalla direzione di visualizzazione, come mostrato da Hill e altri in [27].

- **Dipendenza dalla direzione di illuminazione**
  Alcuni studi [28] hanno dimostrato come una luce proveniente dall'alto renda più semplice il riconoscimento di un volto umano rispetto ad una luce proveniente dal basso.

- **Riconoscimento di volti in movimento**
  Un recente studio [29] ha dimostrato che volti "famosi" siano più semplici da riconoscere se visti in sequenze in movimento piuttosto che in immagini statiche. Questa osservazione è stata estesa [23] per mostrare che il movimento aiuti nel riconoscimento di volti familiari, anche se questi vengano sottoposti a diversi livelli di "degradazione", come la conversione in negativo o in bianco e nero. D'altra parte, esperimenti riguardanti il riconoscimento di volti non familiari hanno mostrato che il movimento non dia alcun beneficio rispetto ad immagini statiche.

- **Espressioni facciali**
  Da studi neurofisiologici [30] sembra che l'analisi delle espressioni facciali venga fatta separatamente rispetto al riconoscimento del volto. I pazienti affetti da prosopagnosia, i quali non sono in grado di riconoscere volti familiari, riescono comunque a riconoscere le espressioni facciali. D'altra parte, pazienti sofferenti di una sindrome cerebrale che non permette loro di analizzare le espressioni del volto, riescono a riconoscere i volti delle persone piuttosto facilmente.

2.5 Problemi aperti nel riconoscimento di volti

Sebbene diverse tecniche per il riconoscimento di volti siano state proposte e si siano dimostrate efficaci, esistono ancora alcune problematiche che acco-

munano un po' tutti i sistemi fin qui sviluppati. Una valutazione di diversi algoritmi per il riconoscimento di volti è stata effettuata nel 1996 con il test FERET [67], il quale ha evidenziato almeno due problematiche rilevanti: le variazioni di illuminazione e i cambiamenti di posa. Entrambi questi problemi possono causare serie perdite di efficacia per la maggior parte dei sistemi di riconoscimento esistenti. Sfortunatamente i cambiamenti di illuminazione e di posa sono inevitabili quando i volti sono acquisiti in un ambiente non controllato, come può essere un video di sorveglianza. In questo paragrafo verranno brevemente esaminati i due problemi ed elencati alcuni approcci per risolverli.

2.5.1 Il problema delle variazioni di luminosità

Il problema dell'illuminazione è illustrato nella figura 2.4, dove volti della stessa persona appaiono diversi a causa di variazioni di illuminazione.

Figura 2.4: Volti della stessa persona in condizioni di luce differenti

Le variazioni indotte da cambi di luce sono spesso superiori alle differenze fra i volti dei vari individui e dunque possono causare una classificazione scorretta. Questo è stato osservato sperimentalmente in [68] usando un insieme di immagini di 25 individui, ed è stato dimostrato teoricamente in [69] per sistemi basati sulla PCA. Il problema dell'illuminazione è piuttosto complesso ed ha ricevuto particolare attenzione nella letteratura della visione computazionale. Nel caso di riconoscimento di volti, diversi approcci specifici al problema sono stati proposti; essi possono essere suddivisi in quattro tipi [70], che vengono qui elencati con i relativi articoli di riferimento a cui si rimanda per maggiori dettagli:

- Metodi euristici ad esempio scartare gli autovettori corrispondenti ai tre autovalori maggiori nella costruzione dello spazio per la PCA [53], oppure basarsi sulla simmetria del volto umano [71];
- Metodi di confronto di immagini in cui vengono usate appropriate rappresentazioni dell'immagine e misure di distanza [68][72];

- Metodi di classificazione che usano molte immagini di un unico volto nella stessa posizione, ma in differenti condizioni di illuminazione [73][74][75][76];
- Approcci in cui si impiegano modelli 3D [69][70][77].

2.5.2 Il problema dei cambiamenti di posa

Le prestazioni dei sistemi di riconoscimento calano vistosamente anche in presenza di cambiamenti nella posa del volto da riconoscere. Per cambiamenti di posa si intendono rotazioni laterali e verticali del volto, come possibile vedere in figura 2.5

Figura 2.5: Volto della stessa persona in pose diverse

Questa difficoltà è stata chiaramente evidenziata dall'ultimo test FERET e la risoluzione del problema del riconoscimento in presenza di rotazioni è stato indicato come un importante fronte di ricerca. I ricercatori hanno proposto diversi metodi per gestire le rotazioni del volto; essi possono essere suddivisi in quattro tipi, che vengono qui elencati con i relativi articoli di riferimento a cui si rimanda per maggiori dettagli:

- Metodi in cui vengono raccolti diversi insiemi di immagini per ogni persona, con rotazioni diverse del volto [78] [79] [80] [81];
- Metodi "ibridi" in cui diversi database di immagini sono disponibili per ogni persona in fase di training, ma solo uno di essi viene usato in fase di riconoscimento [82][83][84][85];
- Metodi basati su una singola immagine campione; metodi cioè in cui non viene eseguita nessuna fase di training;
- Metodi basati sul riconoscimento delle singole componenti del volto, piuttosto che del volto intero [54].

Il secondo approccio sembra essere il più popolare, mentre il terzo non ha ad oggi ricevuto particolare attenzione. Il quarto ed ultimo approccio è il più recente e sembra essere piuttosto promettente.

Capitolo 3

Tecniche di riconoscimento dei volti

In questo capitolo si presentano alcune tecniche fra le più comuni per il riconoscimento di volti umani. La Principal Component Analysis è quella usata dal mio sistema di riconoscimento e verrà spiegata in maggior dettaglio. Le altre tecniche vengono presentate in termini generali, accennando quali siano le caratteristiche fondamentali dei diversi approcci. Per dettagli tecnici e approfondimenti si rimanda ai riferimenti citati. 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 può 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.

Gli algoritmi finora conosciuti si dividono in due categorie:

- Image-Based Face Recognition Algorithms
- Video-Based Face Recognition Algorithms

Il primo gruppo di algoritmi si suddivide a sua volta in :

- PCA
- ICA (Independent Component Analysis)[4]
- LDA
- EP
- EBGM
- Kernel Methods
- Trace Transform[8]
- AAM
- 3-D Morphable Model[5]
- 3-D Face Recognition
- Bayesian Framework
- SVM (Support Vector Machine)[7]
- HMM (Hidden Markov Model)[6]
- Boosting & Ensemble
- Algorithms Comparisons
- 2d-Laplacianfces
- Discriminanti di Fischer

Per il secondo rimandiamo al sito di riferimento
http://www.face-rec.org/algorithms/

3.1 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.

3.1.1 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 all'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 e 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 $V = D \cdot T$, 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 degli 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. 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.

Ora passeremo ad analizzare i metodi di riconoscimento in ordine di rilevanza.

3.2 Discriminati di Fisher

R. A. Fisher sviluppò i discriminanti lineari di Fisher negli anni '30 [49], ma solo recentemente essi sono stati utilizzati nell'ambito del riconoscimento di oggetti. Una spiegazione dei discriminanti di Fisher può essere trovata in [50]. Swets e Weng usarono i discriminanti di Fisher per applicazioni di identificazione nel 1996 [51]. Belhumeur, Hespanha e Kriegman li usarono per il riconoscimento di volti con diverse condizioni di illuminazione [52].

3.3 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. Il metodo Laplacianfaces è stato sviluppato recentemente proprio per il riconoscimento automatico dei volti. É 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. É 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. Il nuovo algoritmo ha mostrato di sbaragliare gli altri algoritmi di riconoscimento dei volti come il Laplacianfaces monodimensionale, il 2D Eigenfaces ed il Fisherfaces.

3.3.1 L'idea e l'algoritmo 2DLaplacianfaces

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

$$Y = AX \tag{3.1}$$

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 cosi definita:

$$\min_X \sum_{ij} \|Y_i - Y_j\|^2 S_{ij} \tag{3.2}$$

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 cosi:

$$S_{ij} = \begin{cases} exp(-\|A_i - A_j\|^2/t) & \text{se } x_i \text{ è tra i k-nearest} \\ & \text{neighbors di } x_j \text{ o } x_j \text{ è} \\ & \text{tra i k-nearest neighbors di } x_i \\ 0 & \text{altrimenti} \end{cases} \tag{3.3}$$

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 3.2 che la funzione obiettivo impone una grossa penalità se due arbitrarie immagini campione sono mappate distanti nello

spazio di 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:

$$\begin{aligned}
&\min_X \sum_{ij} \|Y_i - Y_j\|^2 S_{ij} = \sum_{ij} \|A_i X - A_j X\|^2 S_{ij} \\
&= \sum_{ij} [X^T (A_i - A_j)^T (A_i - Aj) X] S_{ij} \\
&= X^T \Big[\sum_i A_i^T A_i \sum_j S_{ij} - \sum_{ij} A_i^T S_{ij} A_i \Big] X \\
&= X^T A^T (D - S) A X = X^T A^T L A X
\end{aligned} \tag{3.4}$$

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 $1 \times N$ e $N \times 1$, 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 $N \times N$, 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-esima immagine nello spazio d'origine. S è la matrice di similarità ed L è chiamata matrice Laplaciana. Entrambe queste matrici sono di dimensioni $N \times N$. Ogni elemento della matrice D indica quanto sia importante ciascun punto. Inoltre viene imposto questo vincolo:

$$X^T A^T D A X = 1 \tag{3.5}$$

Quindi, il metodo 2D-Laplacianfaces verrà formulato come il

$$\begin{aligned}
&\min_X X^T A^T L A X, \\
&s.t \quad X^T A^T D A X = 1.
\end{aligned} \tag{3.6}$$

Nell'equazione (3.6)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 quegli elementi che hanno neighbors più vicini. Nell'equazione (3.6), 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 (3.6) ad un problema generalizzato di autovalori come mostrato nell'equazione (3.7)

$$A^T LAX = \lambda A^T DAX \tag{3.7}$$

dove le matrici $A^T LA$ e $A^T DA$ sono entrambe di dimensione $N \times N$, ed L e D sono simmetriche e semi-definite positive. 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.

3.3.2 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 $B = [Y_i.., ..., Y_d]$ di dimensioni $m \times d$ denominata la matrice delle features dell'oggetto immagine A.

3.3.3 2DLaplacianfaces: 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 $B_j = [Y_j.., ..., Y_{jd}]$ è definita da

$$d(B_i, B_j) = \sum_{p=1}^{d} \|Y_{ip} - Y_{jp}\| \tag{3.8}$$

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

3.4 Il metodo MultiReGEC

3.4.1 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 geometriche 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. É stato dimostrato che gli algoritmi multiclasse possono essere applicati con successo in numerosi casi, come il riconoscimento dei caratteri, dei rischi del management, delle prognosi e diagnosi mediche. 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 Discriminant Analysis (LDA), che cerca il piano discriminante utilizzando una tecnica di trasformazione delle features. É stato dimostrato che il modello decisionale ottenuto mediante SVMs nella classificazione binaria è equivalente alla soluzione ottenuta mediante LDA. Tutte queste tecniche hanno però 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 interessan-

te sotto numerosi punti di vista. Innanzitutto è possibile implementarlo in diversi ambienti per il problem-solving come MatLab, R e Weka, poiché 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, poiché 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 può 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 può ben comparare con quella dei metodi già esistenti e fornisce un approccio costruttivo per l'estensione dei metodi binari ai problemi multiclasse.

3.4.2 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 può 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 $A \in \mathbf{R}^{n\times m}$ e $B \in \mathbf{R}^{k\times m}$, 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

$$\begin{array}{l} \min f(w) = \frac{w'w}{2} \\ (Aw + b) \geq e \\ (Bw + b) \leq -e \end{array} \tag{3.9}$$

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 $\mathbf{R}^m$. Per soddisfare la condizione precedente per i punti in A, l'iperpiano può essere ottenuto risolvendo questo problema di ottimizzazione:

$$\min_{w,\gamma\neq 0} \frac{\|Aw - e\gamma\|^2}{\|Bw - e\gamma\|^2} \tag{3.10}$$

L'iperpiano per B può essere ottenuto minimizzando l'inverso della funzione obiettivo della (3.11). Ora sia

$$G = [A - e]'[A - e], \quad H = [B - e]'[B - e], \quad z = [w' - \gamma]', \quad (3.11)$$

quindi l'equazione (3.10) diventa:

$$\min_{z \in \mathbf{R}^m} \frac{z'Gz}{z'Hz} \quad (3.12)$$

3.4.3 Support Vector Machines

Le tecniche fin qui illustrate hanno alla base una procedura comune:

- si crea un sottospazio a partire da un insieme di immagini di training;
- si proietta nel sottospazio un'immagine campione per ogni classe e l'immagine di test da classificare;
- si misura la distanza nel sottospazio dell'immagine test dalle immagini campione e le si assegna la classe corrispondente al campione più vicino.

Ciò che differenzia le tecniche presentate è unicamente la scelta del sottospazio. Vi sono altri tipi di classificatori che si basano su un approccio differente: le Support Vector Machine [53], le quali sono state ampiamente utilizzate per la rilevazione di volti all'interno di immagini [54] e per il loro riconoscimento [55]. Esse sono classificatori binari, cioè servono per distinguere fra due sole classi di elementi; ipotizzando di avere più di due classi vi sono diverse strategie per usare comunque le SVM come classificatori (si veda ad esempio [56]), ma gli approcci più diffusi sono due: one-vs-all e pairwise. Nell'approccio *one-vs-all* ogni SVM è allenata per separare una particolare classe da tutte le altre [57]. Nell'approccio *pairwise* invece ogni SVM è allenata per distinguere fra due classi. Questo approccio, proposto originariamente in [58] per il riconoscimento di oggetti 3-D, fu applicato al riconoscimento di volti in [59]. Per quanto riguarda l'efficacia nel riconoscimento, ad oggi non esistono analisi teoriche delle diverse tecniche multi-classe per le SVM. Esperimenti sul riconoscimento di persone mostrano risultati simili per le due strategie presentate [60]. Un confronto più recente fra le diverse tecniche multiclasse [56] favorisce l'approccio one-vs-all a causa della sua semplicità e la sua grande efficacia nella classificazione. In [61] tre procedure di pre-elaborazione si applicano su immagini in scala di

grigi per ridurre la variazione fra le immagini della stessa classe; in particolare si elimina il contributo dello sfondo, si attenua l'effetto delle ombre e si equalizzano le immagini per diminuire le variazioni di luminosità e contrasto fra di esse. I livelli di grigio così ottenuti vengono passati al classificatore SVM. Vi sono comunque altri tipi di caratteristiche (feature) che possono essere estratte da un'immagine e passate alle SVM; per alcuni esempi si rimanda a [55]. Per la classificazione di volti umani, l'approccio più semplice è quello di usare immagini dell'intero volto, eventualmente pre-elaborate, come sopra descritto. Questo prende il nome di *global approach*. Questa procedura è però particolarmente sensibile a variazioni di postura, dovute per esempio a rotazioni della testa, e quindi richiede che il soggetto sia posto frontalmente alla telecamera per poter effettuare il riconoscimento. Un approccio diverso, più robusto nei confronti di variazioni di postura, è il cosiddetto *component-based approach*. In questo caso vengono classificate singole componenti del volto (ad esempio occhi, naso e bocca) anziché l'intero volto. Esperimenti effettuati in [54] mettono a confronto i due approcci qui presentati e dimostrano come il component-based approach sia più efficace del global approach per la classificazione di volti con posture variabili.

3.4.4 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 agli algoritmi base per generalizzare il problema di classificazione alle classi multiple. Essenzialmente esistono due strategie: la prima consiste nel suddividere il problema in sotto-problemi, 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. Infine 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.

3.4.5 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 $w_l' - \gamma - l = 0$, con l = 2,3,4. É possibile stimare una media di queste rette, valutando la media $\widetilde{w}$ 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 3.1, 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 iperpani 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 $z_{i,l} = [w_{i,l}'\gamma_{i,l}]'$

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

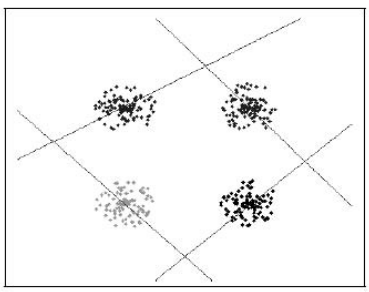

Figura 3.1: Classificazione di 4 classi di uno spazio a due dimensioni

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

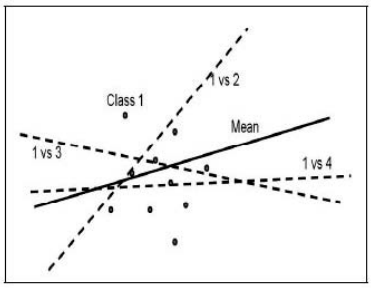

Figura 3.2: Iperpiano medio della classe A1

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, U_1 dove 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 3.2 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 3.3, in cui l'elemento di test viene assegnato alla classe 4.[5]

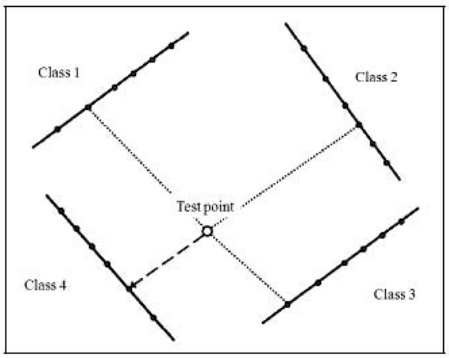

Figura 3.3: Predizione della classe in uno spazio di due dimensioni

3.5 Indipendent Component Analysis

La Indipendent Component Analysis (ICA) è un'altra tecnica usata per la costruzione del sottospazio in cui proiettare le immagini per la loro classificazione. La differenza fondamentale dalla PCA riguarda lo scopo per cui si ricercano i vettori di base su cui proiettare le immagini. Nella PCA vengono ricercati i vettori (le direzioni) nello spazio originale in cui è massima la variazione fra le immagini del training-set; questo ha lo scopo di caratterizzare l'immagine solo per le sue componenti principali. Scopo della ICA è invece trovare vettori di base statisticamente indipendenti fra loro.
Per far ciò non esiste un calcolo fisso come nella PCA (in cui si diagonalizza la matrice covarianza), bensì esistono vari algoritmi iterativi basati su diversi criteri di ricerca [45]. Uno degli algoritmi più conosciuti per la ICA è Infomax, sviluppato da Bell e Sejnowski [46]. La ICA è di gran lunga più dispendiosa della PCA in termini di calcolo e dunque più lenta (con tempi di esecuzione nell'ordine delle ore anziché dei secondi). Esperimenti effettuati in [47] indicano che nel riconoscimento di volti umani la ICA è meno efficace della PCA. Altri esperimenti sul riconoscimento di caratteristiche locali del volto, come le diverse espressioni [48], hanno mostrato che la ICA in questo tipo di applicazioni è più efficace. Ciò è dovuto al fatto che i vettori base calcolati dalla ICA siano statisticamente indipendenti fra loro; questo determina una maggiore localizzazione spaziale dei vettori nello spazio delle immagini. Per questa ragione la ICA è maggiormente adatta a catturare caratteristiche locali all'interno del volto.

3.6 Approcci predominanti al riconoscimento automatico del volto

In letteratura, gli approcci al riconoscimento automatico del volto, possono essere classificati in due principali categorie: quella *geometrica* (basata sull'analisi di *feature o caratteristiche locali*) e quella relativa all'approccio di tipo *olistico* (basato su attività di *template matching*). Il continuo interesse della ricerca scientifica per questo ambito ha portato allo sviluppo di molteplici algoritmi di riconoscimento facciale in entrambe le categorie descritte sopra.
Tra i più noti e ben descritti, vi sono l'*Analisi delle Componenti Principali* (PCA), l'*Analisi Discriminante Lineare* (LDA) e il metodo dell'*Elastic Bunch Graph Matching*.

Il metodo dell'Analisi delle Componenti Principali, noto anche con il nome di "Eigenfaces", costituisce una delle prime strategie sviluppate nell'ambito del riconoscimento dei volti. I primi ad introdurla furono, infatti, M.Kirby e L.Sirovich nel 1988.
Tale metodo fa utilizzo di un "*training set*" costituito da un'insieme di immagine delle stesse dimensioni e "normalizzate" in maniera da evidenziare caratteristiche del volto, quali occhi e bocca. Questa strategia prevede successivamente una riduzione della dimensionalità dei dati a disposizione, tramite una proiezione in un sottospazio, nel quale vengono messe in risalto le caratteristiche salienti di un volto. Tale riduzione dimensionale, infatti, permette di escludere l'informazione che non viene considerata rilevante e precisamente decompone la struttura di un volto in una combinazione di componenti ortogonali, scorrelate tra loro, dette *eigenface*.
Ogni immagine di volto, può successivamente essere rappresentata come una somma pesata (*vettore delle feature*) di queste *eigenfaces*, raccolte in un vettore monodimensionale. Il confronto di un'immagine di volto con le altre presenti nel training set viene effettuato semplicemente valutando la distanza tra questi *vettori di caratteristiche locali*.

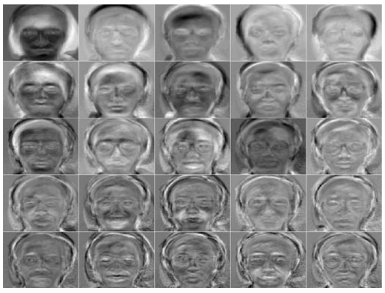

Figura 3.4: Esempi di eigenfaces

L'analisi discriminante lineare (LDA) consiste, invece, in un' approccio statistico volto a classificare immagini di volti in classi non note rispetto a immagini contenute in classi note, contenute in un *training set*. Questa tecnica punta a massimizzare la varianza fra classi di immagini distinte e a minimizzare invece quella che caratterizza immagini di volti appartenenti. [62]

La tecnica dell'*Elastic Bunch Graph Matching* si basa sul fatto che nella realtà le immagini di volti possiedono caratteristiche strutturali simili fra loro. Questa strategia attua la rappresentazione dei volti mediante l'uso di particolari grafi etichettati, i cui nodi rappresentano i punti

ben precisi della faccia, quali naso, bocca, mentre gli archi contengono le distanze fra i nodi stessi. Questi ultimi contengono un set di "*wavelets*" (forme d'onda) di Gabor, scelte per rappresentare un intervallo di frequenza, direzioni ed estensioni, che meglio caratterizzano quella regione. Ogni insieme finito di *wavelets* in un particolare punto del grafo forma un vettore di caratteristiche locali chiamato *jet*. Un'insieme di *jets* può quindi caratterizzare un'immagine, un ridotto insieme di numeri tramite i quali è possibile il confronto fra due immagini.

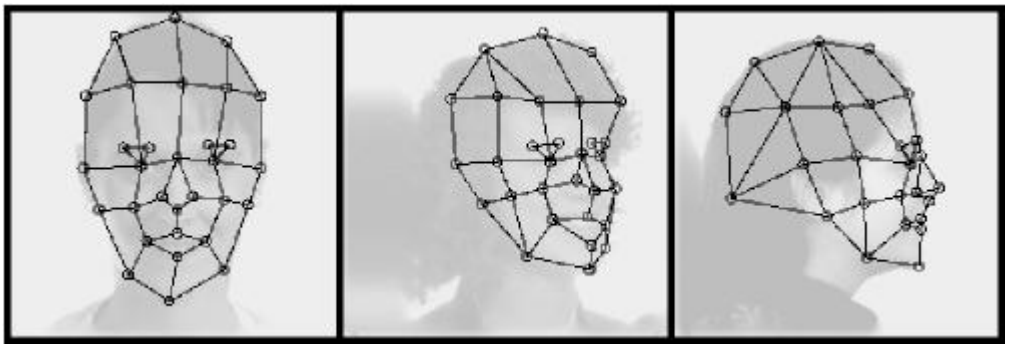

Figura 3.5: Esempio di mappatura del grafo su un volto

3.6.1 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. É 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 training, rispettivamente. La complessità di spazio è $O(n^2m^2)$.Un insieme di eigenfaces può essere generato eseguendo un processo matematico chiamato analisi delle componenti principali (PCA) su un ampio insieme di immagini raffiguranti diversi volti umani. Informalmente, eigenfaces può essere considerato un insieme di "standardized face ingredients", derivata da analisi statistica di molte immagini di volti. Ogni volto umano può essere considerato come una combinazione di questi volti standard. Per esempio, il viso potrebbe essere composto dal viso medio, maggiorato del 10% dal eigenface 1, 55% da eigenface 2, e addirittura -3% da eigenface 3. Sorprendentemente, non servono molte eigenfaces combinate insieme per raggiungere una buona approssimazione della maggior parte dei volti. Inoltre, poiché il volto di una persona non viene registrato da una fotografia digitale, ma piuttosto come un semplice elenco di valori (un valore per ogni eigenface nel database utilizzato), molto meno spazio viene dato per il volto di ogni persona. Le eigenfaces che vengono create appariranno come zone chiare e scure disposte in un pattern specifico. Questo modello è il modo in cui diverse caratteristiche di un viso sono individuate da valutare. Ci sarà un pattern per valutare la simmetria, se non vi è tutto lo stile della barba, dove è l'attaccatura dei capelli, o valutare le dimensioni del naso o la bocca. Altre eigenfaces hanno modelli che sono meno semplici da identificare, e l'immagine dell'eigenface può assomigliare molto poco ad un volto. La tecnica utilizzata nella creazione di eigenfaces e che li utilizza per il riconoscimento viene utilizzato anche al di fuori del riconoscimento facciale. Questa tecnica è utilizzata anche per l'analisi della scrittura, la lettura labiale, riconoscimento vocale, la lingua dei segni interpretazione mano gesti e di analisi di immagini mediche. Di conseguenza, alcuni non utilizzano il termine eigenface, ma preferiscono usare "eigenimage".

Implementazione

Per creare un set di eigenfaces, si deve:

1. Preparare un training set di immagini di volti. Le immagini che costituiscono il training set deve essere preso sotto le stesse condizioni di illuminazione, e deve essere normalizzato per avere gli occhi e bocche allineate su tutte le immagini. Essi devono anche essere tutti ricampionati per avere la stessa risoluzione dei pixel. Ogni immagine è trattata come un vettore, semplicemente concatenando le righe di pixel dell'immagine originale, risultando in una sola riga con elementi $r \times c$. Per questa applicazione, si pre-

sume che tutte le immagini del training set siano memorizzate in una matrice T unica, dove ogni riga della matrice è un'immagine.

2. Sottrarre la media. L'immagine media a deve essere calcolata e poi sottratto da ogni immagine originale in T.

3. Calcolare gli autovettori e autovalori della matrice di covarianza S. Ogni autovettore ha la stessa dimensionalità (numero di componenti), come le immagini originali, e quindi può essere visto come un'immagine. Gli autovettori di questa matrice di covarianza sono quindi chiamate eigenfaces. Sono le direzioni in cui le immagini differiscono dalle immagini media. Di solito questo sarà un passo computazionalmente costoso (se possibile), ma l'applicabilità pratica di eigenfaces deriva dalla possibilità di calcolare gli autovettori di S in modo efficiente, senza mai computare S in modo esplicito, come dettagliato di seguito.

4. Scegliere i componenti principali. La matrice di covarianza $D \times D$ si tradurrà in autovettori D, ognuno dei quali rappresenta una direzione nello spazio immagine $r \times c$-dimensionale. Gli autovettori (eigenfaces) con il più grande autovalore associati sono tenuti.

Queste eigenfaces possono ora essere utilizzate per rappresentare entrambe le facce vecchie e nuove: siamo in grado di proiettare una nuova immagine (mean-subtracted) sulle eigenfaces e, quindi, registrare come nuovo volto che si differenzia dalla faccia media. Gli autovalori associati ad ogni eigenface rappresentano quanto le immagini del training set variano dall'immagine media in quella direzione. Perdiamo informazioni proiettando l'immagine su un sottoinsieme di autovettori, ma noi minimizziamo questa perdita mantenendo le eigenfaces con i più grandi autovalori. Per esempio, se stiamo lavorando con una immagine 100×100, allora otterremo 10.000 autovettori. Nelle applicazioni pratiche, la maggior parte dei volti in genere può essere identificata con una proiezione tra 100 e 150 eigenfaces, in modo che la maggior parte dei 10.000 autovettori possano essere scartati.

Calcolare gli eigenvectors

Eseguire la PCA direttamente sulla matrice di covarianza delle immagini è spesso computazionalmente impossibile. Se piccolo, diciamo 100×100, vengono utilizzate le immagini in scala di grigi, ogni immagine è un punto in uno spazio di 10.000-dimensionale e la matrice di covarianza S è una matrice di $10.000 \times 10.000 = 10^8$ elementi. Tuttavia il rango della matrice di covarianza è limitato dal numero di esempi di

training: se ci sono N esempi di addestramento, ci saranno al più N-1 autovettori con autovalori non-zero. Se il numero di esempi di formazione è inferiore alla tridimensionalità delle immagini, le componenti principali possono essere calcolate più facilmente come segue. Sia T la matrice di esempi di addestramento pre-elaborata, in cui ogni riga contiene un'immagine mean-subtracted. La matrice di covarianza può quindi essere calcolata come

$$Sv_i = T^T T v_i = \lambda_i v_i$$

Tuttavia $T^T T$ è una matrice di grandi dimensioni, e se invece prendiamo la decomposizione di autovalori di

$$TT^T u_i = \lambda_i u_i$$

ci accorgiamo che pre-moltiplicando entrambi i membri dell'equazione con la T^T, otteniamo

$$T^T T^T u_i = \lambda_i T^T u_i$$

Il che significa che, se u_i è un autovettore di TT^T, quindi $v_i = T^T u_i$ è un autovettore di S. Se abbiamo un training set di 300 immagini di 100×100 pixel, la matrice TT^T è una matrice di 300×300, che è molto più gestibile rispetto alla matrice di covarianza 10000×10000 . Si noti tuttavia che i vettori risultanti v_i non vengono normalizzati, se la normalizzazione è richiesta dovrebbe essere applicato come un passo in più.

Uso nel Face Recognition

Il riconoscimento facciale è stato la fonte di motivazione alla base della creazione di eigenfaces. Per questo uso, eigenfaces ha dei vantaggi rispetto ad altre tecniche disponibili, come la velocità del sistema e l'efficienza. Utilizzando eigenfaces è molto veloce, e capace di operare funzionalmente su un sacco di facce in poco tempo. Purtroppo, questo tipo di riconoscimento facciale ha un inconveniente da considerare: problemi nel riconoscere i visi quando sono visualizzati con diversi livelli di luce o angolazioni. Perché il sistema funzioni bene, le facce devono essere viste da una visione frontale con illuminazione simili. Face Recognition utilizzando eigenfaces ha dimostrato di essere abbastanza preciso. Sperimentando il sistema e provandolo sotto le variazioni di determinate condizioni, sono stati trovati i seguenti riconoscimenti

corretti : una media del 96% con la variazione di luce, 85% con variazione di orientamento, e il 64% con la variazione di dimensione. (Turk e Pentland 1991, p. 590)
Per completare eigenfaces, è stato sviluppato un altro approccio chiamato eigenfeatures. Questo combina le metriche del viso (misurando la distanza tra i tratti del viso) con l'approccio eigenface. Un altro metodo, che è in competizione con la tecnica eigenface usa "fisherfaces". Questo metodo di riconoscimento facciale è meno sensibile alle variazioni di illuminazione e posa del volto rispetto al metodo con eigenfaces. Un'alternativa più moderna a eigenfaces e fisherfaces è il *active appearance model*, che disaccoppia la forma del viso dalla sua consistenza: si fa un decomposizione eigenface del volto dopo aver fatto un warping alla forma media. Questo gli permette di lavorare meglio su diverse proiezioni del volto, e quando il volto è inclinato.

3.6.2 Il metodo dell'analisi delle componenti principali

Come visto in precedenza, il riconoscimento del volto umano in immagini 2D può essere fatto tramite diversi tipi di tecniche. Una delle più comuni è quella che utilizza la cosiddetta "*Analisi delle Componenti Principali*" (PCA) o metodo delle autofacce (*Eigenfaces*).
PCA è un utile metodo statistico che permette di trovare *patterns* all'interno di insiemi di dati di grandi dimensioni, classificando tali dati in base al loro grado di similarità. Inoltre estende le sue possibilità di utilizzo non solo all'ambito del riconoscimento, bensì anche a quello della compressione delle immagini digitali. L'implementazione di un sistema completo di riconoscimento facciale, in grado di effettuare anche compiti di autenticazione, richiede che quest'ultimo sia capace di compiere principalmente due tipi di attività: la prima è quella che si riferisce a ciò che in letteratura viene chiamata "face detection" ovvero la capacità di localizzare uno o più volti umani all'interno dell'immagine o di una sequenza video, mentre la seconda è detta "face recognition" ed è strettamente orientata al riconoscimento vero e proprio dei volti, con l'obiettivo di giungere a una classificazione in base a particolari caratteristiche.

3.6.3 Face Detection

Il metodo Viola-Jones

Il rilevamento del volto è stato effettuato attraverso l'uso di un algoritmo di face detection automatico. Tra i numerosi metodi presenti in letteratura, si è scelto di utilizzare il metodo proposto da Viola-Jones,

per le sue caratteristiche di accuratezza ed efficienza computazionale. Una possibile implementazione di tale algoritmo è possibile trovarla all'interno delle librerie *Intel OpenCV* [63], all'interno della sezione dedicata agli esempi d'uso di queste ultime. Il codice sorgente è situato in un file chiamato "facedetect.c", da cui è stato preso spunto per l'implementazione del sistema di localizzazione dei volti, che verrà presentato nella sezione dedicata all'applicativo. L'obiettivo è quello di ottenere il cosiddetto "classificatore a cascata per volti frontali" da poter utilizzare per localizzare volti all'interno di immagini. L'algoritmo è in grado di rilevare volti in posa approssimativamente frontale, ovunque essi siano presenti nell'immagine e l'output è costituito da una serie di regioni rettangolari, ognuna centrata su un volto e i cui limiti racchiudono il volto stesso.

Figura 3.6: Detection di un volto tramite l'algoritmo di Viola-jones

Tale metodo basato su una tecnica di apprendimento statistico supervisionato chiamata *Adaboost*. Il sistema classifica le patch dell'immagine come volto o "non-volto" in base alla risposta a dei semplici template rettangolari; la feature così ottenuta prende il nome di *Haar-like feature*

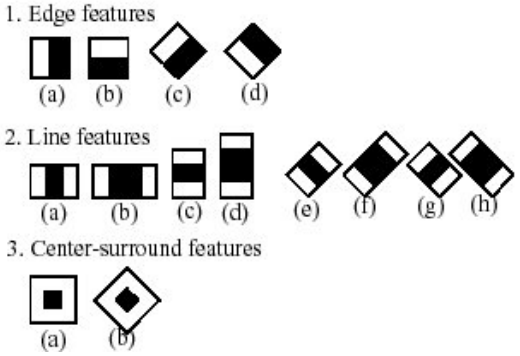

Figura 3.7: Le cosiddette Haar-like features

Per ogni template, il corrispondente valore della feature dato dalla somma dell'intensità dei pixel $p_b(i)$ collocati all'interno dell'area in nero, meno la somma dell'intensità dei pixel $p_w(i)$ nell'area bianca:

$$f(x, y) = \sum_i p_b(i) - \sum_i p_w(i)$$

Le features caratterizzano così il contrasto locale delle zone salienti dell'immagine. L'utilizzo di features dei valori di intensità dei singoli pixel dell'immagine, è giustificato principalmente da due motivi:

1. Le features sono in grado di codificare la conoscenza di un particolare dominio; un vasto e generico insieme di *Haar-like freature*, unitamente ad un processo di selezione delle features, può aumentare la capacità di apprendimento dell'algoritmo.

2. I sistemi di features operano più velocemente; uno dei principali punti di forza del metodo è proprio dato dal fatto che la localizzazione del volto viene eseguita in tempo reale, facendo scorrere sull'immagine una finestra di ricerca, di dimensione fissa a varie scale.

Le feature rettangolari possono essere calcolate molto rapidamente adottando una rappresentazione intermedia dell'immagine detta *integral image*. Il pixel $p(x, y)$ dell' *integral image*, indicata con $ii(x, y)$, è dato dalla somma di tutti i pixel dell'immagine $I(x, y)$ che stanno a sinistra e sopra (x, y):

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

La *integral image* può essere calcolata in un solo passo attraverso l'espressione ricorsiva:

$$\begin{gathered} s(x, y) = s(x, y - 1) + i(x, y) \\ ii(x, y) = ii(x - 1, y) + s(x, y) \\ s(x, -1) = 0, ii(-1, y) = 0 \end{gathered}$$

Il passo successivo consiste nel processo di apprendimento del classificatore. Dato un insieme di feature ed un training set costituito da immagini positive e negative, un qualsiasi metodo di apprendimento può essere usato per addestrare una funzione di classificazione; il sistema adotta una variante di Adaboost. Si noti che per rilevare facce indipendentemente dalla posa, è necessario addestrare due versioni del *feature detector*, una per i volti frontali (o quasi frontali) e una per

quelli di profilo. L'insieme di feature ottenuto col procedimento appena descritto è estremamente ampio; se ad esempio la risoluzione del detector è di 24×24 pixel, l'insieme di *Haar-like feature* è pari ad oltre 180.000 elementi. Il metodo di apprendimento, chiamato *weak-learning*, è in grado di selezionare da tale insieme un set limitato di feature combinandolo con un opportuno *weak-classifier*. Questo avviene selezionando la feature rettangolare che meglio separa gli esempi positivi da quelli negativi; per ogni feature, il *weak-learner* determina la funzione di classificazione a soglia ottimale. Un weak-classifier $h_j(x)$ associato alla feature x, consiste in una feature f_j, una soglia Θ_j e una polarità p_j:

$$h_j(x) = 1 \text{ sse } p_j f_j(x) < p_j \Theta_j$$

L'ultimo passaggio consiste nella ricostruzione di una cascata di weak-classifier, ovvero un classificatore a stadi che sia in grado di respingere molte sotto-finestre negative e, contemporaneamente, di rilevare la quasi totalità delle istanze positive. La cascata viene realizzata posizionando nelle prime posizioni i classificatori più semplici, in modo da eliminare rapidamente buona parte delle finestre negative; in un secondo tempo entrano in gioco i classificatori più complessi in grado di scartare i rimanenti falsi-positivi.[64]

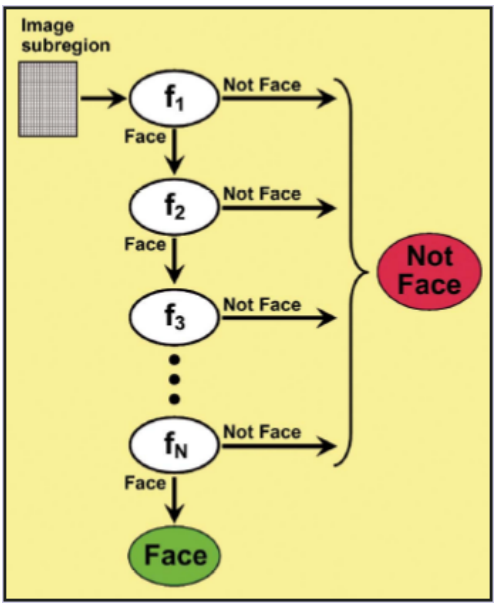

Figura 3.8: Il classificatore a cascata è una catena di filtri. Sottoregioni dell'immagine che la compongono attraverso la cascata sono classificati come "Facce". Tutti gli altri sono classificati come "Non faccia"

3.6.4 Face Recognition

Come primo approccio al metodo dell'analisi delle componenti principali applicato al riconoscimento facciale, è possibile trovare un'ottima descrizione teorica di tale metodo nel lavoro svolto da Romdhani [65] e da M.Turk e A.Pentland[66]. Ciò che segue nelle prossime pagine è infatti tratto da tali documenti a mio avviso molto utili e in grado di facilitare la comprensione dell'utilizzo della PCA.

Il volto visto come un vettore

L'immagine di un volto sostanzialmente può essere trasformata in un vettore. Se chiamiamo w la larghezza dell'immagine e h la sua altezza, entrambe in pixel, il numero delle componenti del vettore che si vuole ottenere è dal prodotto $w * h$, dove ogni pixel dell'immagine iniziale corrisponde ad una componente del vettore. La costruzione di tale vettore quindi può essere effettuata tramite una semplice concatenazione delle righe della matrice dell'immagine iniziale:

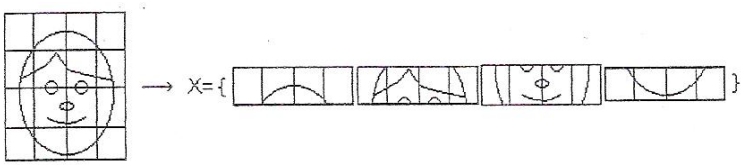

Figura 3.9: La trasformazione di un'immagine in un vettore

Lo "spazio delle immagini"

Il vettore descritto appartiene ad uno spazio vettoriale. Tale spazio è chiamato "spazio delle immagini" ed è lo spazio di tutte le immagini la cui dimensione è di $w * h$ pixels. La base di tale spazio vettoriale si compone con i seguenti vettori:

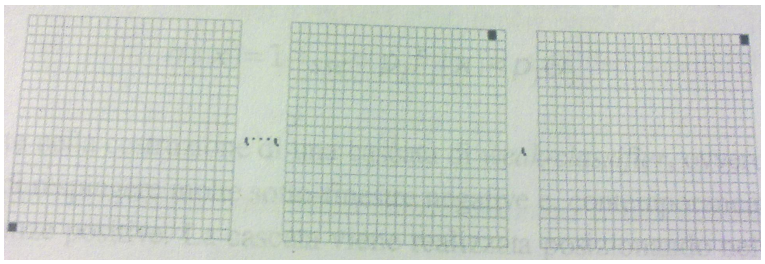

Figura 3.10: La base dello spazio delle immagini

Tutti i volti presenti in ogni immagine si "assomigliano" fra loro; infatti ognuno è caratterizzato dalla presenza di due occhi, una bocca, un naso ecc, ovvero tutte caratteristiche localizzate nella stessa area. Ciò significa che tutti i punti rappresentanti i volti non si spargono in maniera omogenea nello spazio, bensì tendono a localizzarsi in un ristretto "*cluster*" nello spazio immagine, esattamente come mostrato dalla seguente figura.

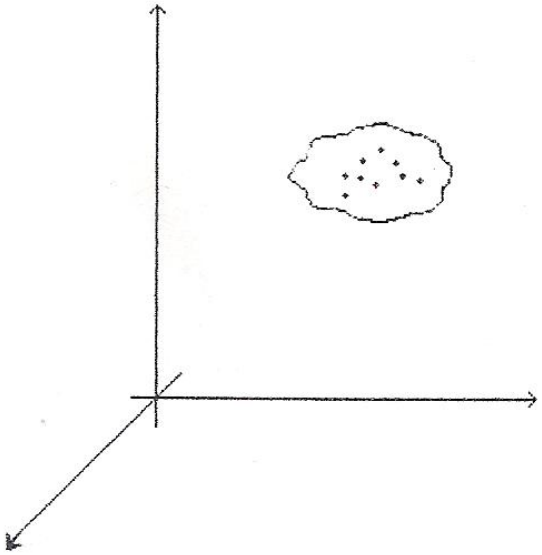

Figura 3.11: Spazio dei volti e spazio delle immagini

Lo spazio delle immagini è uno spazio di dimensione $w * h$. Ovviamente non tutti i pixels che compongono l'immagine di un volto possono essere considerati "rilevanti" ed in particolare ognuno di questi dipende strettamente dal suo immediato vicino. Tale considerazione ci permette di capire che la dimensione dello spazio dei volti sarà sicuramente diversa rispetto allo spazio delle immagini iniziali, ed in particolare la dimensione di tale spazio sarà sicuramente di una dimensione minore. L'obiettivo dell'approccio al riconoscimento facciale tramite PCA è proprio quello di cercare di ridurre la dimensionalità dei dati iniziali, proiettandoli in un sottospazio. Quest'ultimo è definito da una nuova base ed è in grado di descrivere al meglio il "modello" dell'insieme dei dati iniziali, insieme che in questo caso corrisponde a quello dei volti presenti nel *training set* iniziale. I vettori che compongono la base di questo spazio sono detti componenti principali, devono essere correlati fra loro e devono massimizzare la varianza stimata per le variabili originali. [65]

Il calcolo delle componenti principali

Procediamo ora in maniera più approfondita nel calcolo delle componenti principali di un dato insieme di facce appartenenti al training set che si vuole utilizzare per effettuare compiti di riconoscimento facciale. Le prossime righe presenteranno in maniera algoritmica come calcolare tali componenti ed i vari passaggi saranno spiegati tramite l'ausilio di formule matematiche.
Siano,

$$I_1, I_2, I_3, \ldots, I_M$$

immagini del training set iniziale di dimensione $N \times N$, il primo passaggio da effettuare è quello di rappresentare ogni immagine I_i come une vettore Γ_i di dimensione $N^2 * 1$ esattamente come nella figura.
Dopo aver trasformato ogni immagine a disposizione in vettori è necessario calcolare il *vettore medio* Ψ di tutte le facce a disposizione nel training set, definito come

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$

Tale valore medio Ψ dovrà essere sottratto ad ogni singolo vettore Γ_i come segue

$$\Phi_i = \Gamma_i - \Psi_i$$

in modo da "mediare" il valore di ogni singolo pixel contenuto nel vettore stesso.
Chiamiamo la matrice così definita,

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \cdot \Phi_n^T = AA^T$$
$$\text{dove } A = [\Phi_1 \Phi_2 \ldots \Phi_M]$$

Tale matrice C è una matrice di dimensione $N^2 * N^2$ ed è comunemente detta "matrice di covarianza". Dal punto di vista statistico, questa matrice definisce il modo in cui variano coppie di *features* nell'analisi statistica di un particolare fenomeno, e rappresenta quindi il modo in cui ogni *feature* varia rispetto alle altre.
Nel caso dell'analisi di un trainig set di volti permette quindi di rappresentare le variazioni delle caratteristiche principali di ogni volto in ogni singola immagine del training set, permettendo quindi di utilizzare tali variazioni per effettuare ad esempio compiti di riconoscimento facciale.

A questo punto, si rende necessario andare a calcolare tutte quelle caratteristiche che descrivono l'insieme di volti in questione, ed è proprio questa la fase che conferisce a questa metodologia di analisi statistica il nome di *Analisi delle Componenti Principali*, o comunemente detta PCA.
Si procede quindi con il calcolo degli autovettori u_i, e dei relativi autovalori della matrice di covarianza C definita come il prodotto della matrice A per la sua versione trasposta A^T.
Il calcolo degli autovalori e degli autovettori di una matrice di covarianza così definita, è piuttosto complesso e poco pratico, soprattutto dal punto di vista computazionale, in quanto la matrice $C = AA^T$ è una matrice di dimensioni molto grandi: di fatto una matrice di questo tipo possiede N^2 autovalori e autovettori. Utilizzando, per esempio, immagini di volti di dimensione 100×100 si andrebbe a calcolare autovettori e autovalori di una matrice di dimensione 10000×10000, ovvero 10000 autovettori; ciò costituisce sicuramente un'operazione "pesante" e poco pratica, che inevitabilmente andrebbe ad influire sulle prestazioni di un possibile sistema di riconoscimento, specie se si è intenzionati ad eseguire tali riconoscimenti in condizioni *real-time*.
Tuttavia, è possibile ridurre la complessità del calcolo degli autovettori della matrice di covarianza, utilizzando qualche piccola conoscenza nel campo dell'algebra lineare.
Consideriamo, infatti, la matrice di covarianza definita come:

$$C = A^T A$$

Tale matrice è una matrice di dimensione $M \times M$ con autovettori v_i. É facile dimostrare che le matrici AA^T e $A^T A$ possiedono gli stessi autovalori e gli stessi autovettori. La dimostrazione di questo fatto è riportata nelle prossime righe:

$$A^T A v_i = \mu_i v_i$$

Che correlazione esiste tra gli autovettori u_i e v_i delle rispettive matrici di covarianza?

$$A^T A v_i = \mu_i v_i \quad \Rightarrow \quad AA^T A v_i = \mu_i A v_i \quad \Rightarrow$$
$$\Rightarrow \quad C A v_i = \mu_i A v_i \quad \text{oppure} \quad C u_i = \mu_i v_i \quad \text{dove} \quad u_i = A v_i$$

Da questi semplici passaggi algebrici è facile notare che le matrici AA^T e $A^T A$ hanno gli stessi autovalori e i loro autovettori sono legati dalla relazione $u_i = Av_i$. Questa considerazione è fondamentale in quanto riduce notevolmente la dimensione della matrice C e di conseguenza la

complessità del calcolo della matrice di covarianza definita originariamente. É Possibile inoltre affermare che questi ultimi N^2 autovettori delle matrici di covarianza, e i relativi autovalori, corrispondono esattamente agli M autovettori (e corrispondenti autovalori) più grandi della matrice di covarianza definita come prodotto della matrice A per la sua versione trasposta.
L'intero numero degli autovettori non è necessario per descrivere completamente ogni caratteristica dei volti presenti in un possibile training set. Infatti, è possibile utilizzare unicamente i K autovettori che possiedono i relativi autovalori con valore più grande. Tale affermazione è motivata dal fatto che maggiore è l'autovalore associato, più la caratteristica del volto descritta da quest'ultimo è discriminante e quindi utile per una possibile classificazione.[66]
A questo punto della trattazione, si rende necessario effettuare alcune considerazioni sugli autovalori ottenuti tramite le precedenti operazioni. Nell'ambito della PCA applicata al riconoscimento facciale, tali autovettori, o componenti principali del sottospazio ottenuto tramite la trasformazione, prendono il nome di *autofacce* (eigenfaces).

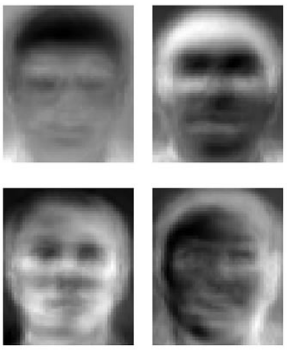

Figura 3.12: Esempio di eigenfaces

Esse non sono altro che delle immagini che possono essere considerate come componenti di base di un volto. Le immagini del training set, alle quali ricordiamo è sottratta l'immagine media proiettandole nel sottospazio dei volti, sono quindi rappresentate come combinazione lineare delle migliori K eigenface ottenute dall'analisi delle componenti principali esattamente come descritto dalla seguente espressione:

$$\widehat{\Phi}_i - media = \sum_{j=1}^{K} w_j u_j, \quad \text{con} \quad w_j = u_j^T \Phi_j \quad \text{e } u \text{ sono le eigenfaces}$$

Figura 3.13: Un volto rappresentato come combinazione lineare do eigenfaces e relativi coefficienti associati

Da ciò segue che, in questa base ogni immagine Φ_i *normalizzata* appartenente al training set, è quindi descritta da un vettore, detto *vettore delle caratteristiche (feature)*, così definito

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \\ \dots \\ w_k^i \end{bmatrix}, i = 1, 2, 3, \dots, M$$

dove $w_1^i, w_2^i, w_3^i, \dots, w_k^i$ sono i coefficienti che si ottengono dalla decomposizione nel sottospazio. Questi ultimi indicano il "peso" associato ad ogni singolo eigenfaces nella combinazione lineare, ovvero quanto la caratteristica (*feature)*, descritta da una particolare autofaccia, sia importante nel volto a cui quest'ultima si riferisce.[66]

Il riconoscimento di un immagine mediante proiezione del sottospazio dei volti

Il riconoscimento di un volto avviene proiettando l'immagine del candidato nel sottospazio dei volti e cercando il punto più vicino. Questo tipo di ricerca viene effettuata valutando la distanza fra la proiezione dell'immagine da riconoscere e quella di ogni singola immagine immagazzinata nel database. L'immagine a cui viene associata la distanza minima, viene considerata come l'immagine di volto che più è simile a quella del candidato che si vuole riconoscere.
Quindi, sia Γ l'immagine da riconoscere; la prima cosa da fare è quella di normalizzarla, sottraendo l'immagine media del training set, ovvero

$$\Phi = \Gamma - \Psi$$

Come descritto nel paragrafo precedente, la sua posizione è data da

$$\widehat{\Phi} = \sum_{j=1}^{K} w_j u_j \quad \text{con} \quad w_j = u_j^T \Phi$$

É ora possibile rappresentare l'immagine Φ in questione secondo il suo *vettore delle caratteristiche*, utilizzando i relativi coefficienti ottenuti dalla decomposizione nel sottospazio dei volti

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ w_3^i \\ \dots \\ w_k^i \end{bmatrix}$$

A questo punto non resta che valutare la distanza geometrica e^r da tutte le altre immagini che compongono il training set e valutarne quella con il valore minimo,

$$e^r = min_I \|\Omega - \Omega^I\|$$

Il passaggio finale per completare la procedura di riconoscimento consiste semplicemente nel sogliare la distanza minima ricavata tramite l'espressione precedente.[66]

$$e^r < T_r$$

La scelta di tale soglia dipende ovviamente dal modo in cui viene calcolata la distanza geometrica fra i punti del sottospazio. Infatti esistono diversi metodi per calcolare questa distanza; comunemente viene valutata la semplice distanza euclidea, ma ciò non toglie il fatto che è possibile utilizzare altri tipo di parametri, come ad esempio la distanza di *Mahalanobis*. In letteratura è inoltre possibile trovare della documentazione che descrive le "performance" di un certo tipo di calcolo di distanze nel sottospazio dei volti rispetto ad altre.
Di fatto, se la distanza geometrica minima che viene calcolata con l'espressione descritta sopra, è al di sotto di questo limite prescelto, il volto del candidato viene riconosciuto come volto presente nel training set, ed associato alla classe di immagini di facce indicata dall'indice "l".

Alcune considerazioni sul metodo PCA/Eigenfaces

Matematicamente, come già argomentato, si utilizzano delle basi ortonormali alternative a quella di partenza per comprimere vettori facciali. La base usata per questo metodo è chiamata Karhonen-Loève (KL) ed essa è ben conosciuta in letteratura come PCA. Una possibile interpretazione è la seguente: gli autovalori della matrice C rappresentano la varianza della popolazione di volti lungo la direzione u_i (si veda la figura) e siccome l'ammontare di informazione (entropia) di popolazione è

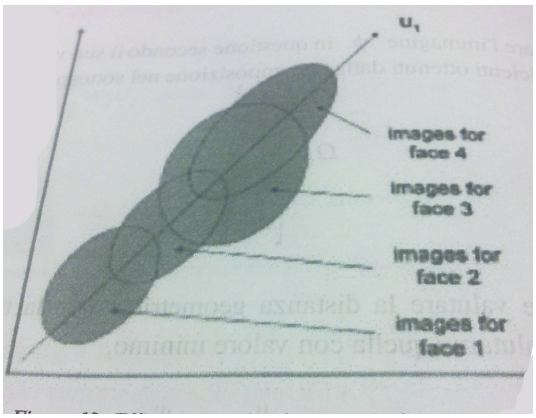

Figura 3.14: Efficacia non implica potere discriminante

proporzionale alla sua ampiezza le eigenfaces manterranno il massimo ammontare di informazione della popolazione.
Tuttavia il potere discriminante offerto dalla rappresentazione ottenuta con KL non è ottimo. Il potere discriminante riguarda la separazione fra le rappresentazioni di individui diversi, piuttosto che l'ampiezza di tutte le facce. Le "performance" a tempo di esecuzione di sistema che usa *eigenfaces* sono molto buone. La costruzione di un insieme di eigenfaces è computazionalmente molto intensa, ma necessita di essere eseguita raramente. Data un'immagine occorre trovare il suo *eigenvector* che, computazionalmente, equivale a risolvere un problema di minimizzazione di minimi quadrati medi. Su un moderno calcolatore si tratta di un calcolo di pochi secondi.[65]
Il processo finale di riconoscimento riguarda il calcolo della distanza fra l'eigenvector del candidato e quelli dei soggetti del training set. Anche senza speciali hardware, questa operazione può ridursi a poche decine di microsecondi per confronto, pochi secondi per database di 100.000 immagini. Pertland [66], afferma che un confronto su database più contenuti (poche centinaia di individui) su hardware standard (Sun Sparc Stations) può raggiungere una velocità pari ai frame rate di una videocamera. La precisione di questo metodo alla distorsione facciale, alla posa e alle condizione di illuminazione è discreta. Sebbene Sirovich e Kirby scoprirono che il loro sistema riconosce candidati che hanno una posa diversa rispetto ai rispettivi campioni, è inevitabile che la qualità del riconoscimento può degradare a seconda del tipo di posa, ma in generale le condizioni di illuminazione restano uno dei problemi principali che affligge questo metodo.[65]

Capitolo 4

Il sistema di riconoscimento

In questo capitolo viene illustrato il funzionamento del sistema di riconoscimento.

4.1 Architettura del sistema

In questo paragrafo vengono presentate le varie componenti hardware e software che costituiscono il nostro sistema di riconoscimento.

Il software è stato sviluppato in ambiente UNIX/Linux, in particolare su una macchina i386 e sistema operativo Debian GNU/Linux, tra ottobre e dicembre 2010.[95]

In dotazione ci è stato dato: un computer-desktop, con installato Ubuntu, in cui noi abbiamo aggiornato la versione alla 10.10 per poter aggiornare poi tutte le librerie di OpenCV e le Qt per l'interfaccia;

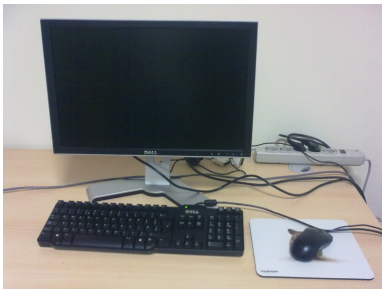

Figura 4.1: Computer in laboratorio VIPS

Hardware

Se si esclude il pc su cui gira il software, le componenti hardware del sistema sono:

- la webcam usata è una Webcam Trust SpaceCm 150 Portable
- la scheda elettronica con relè per far comunicare il pc con la serratura
- la serratura
- il monitor touchscreen CTF400-SL - VGA 7 TFT - Touchscreen USB - PAL/NTSC - IR Remote - Autodimmer - Audio [LED-Backlight] -TRANSFLECTIVE PRO

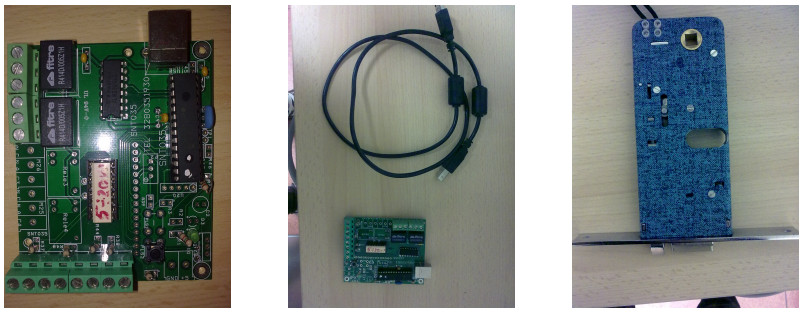

Figura 4.2: Scheda Relè e serratura

Figura 4.3: Webcam

Figura 4.4: Touchscreen

Il funzionamento del software è comunque indipendente dal modello di webcam usato perché, come si vedrà successivamente, le librerie di acquisizione "dialogano" con il driver della periferica di acquisizione, indipendentemente da quest'ultima. La webcam è collegata al PC tramite un normale cavo USB. La scheda è collegata al pc attraverso un normale cavo USB e comunica con la serratura attraverso dei cavi elettrici. Il monitor è collegato con una porta seriale al pc. Dal punto di vista del funzionamento, il software, che permette il riconoscimento del volto di una persona, se l'identificazione ha un esito positivo, attraverso il pc, viene mandato un segnale alla scheda che comunica con la serratura e la sblocca per permettere l'accesso al laboratorio.
Il software da noi sviluppato legge il nome della persona che viene scelto dal soggetto stesso in una lista contenente tutti i soggetti registrati. Come vedremo meglio in seguito, nella fase di training il nome serve per identificare univocamente i campioni salvati, mentre nella fase di riconoscimento indica al sistema quale sia il soggetto rispetto al quale viene avviata la procedura di autenticazione.
La scheda elettronica con relè permette al software di comunicare con la serratura e predisporre l'accesso al laboratorio.

4.2 Configurazione del sistema

La configurazione dell'intero sistema è stato uno dei problemi principali, in quanto il sistema datoci all'inizio, era supportato di una versione non aggiornata di Ubuntu, mentre i programmi, librerie e compilatori erano all'ultima versione. Perciò abbiamo dovuto formattare il computer e installare la nuova versione di Ubuntu 10.10 attraverso una USBkey resa bootable attraverso il programma *Unetbootin* che crea la chiavetta di avvio per il sistema.
Per formattare il PC, appena acceso, si preme F12 per entrare nel *bios* e si seleziona il boot da chiavetta come prima unità da eseguire. Se il boot dà un messaggio di errore selezionare all'interno del bios l'opzione che permette di vedere la chiavetta come hard-disk. Una volta fatto il boot, selezionare installa Ubuntu e seguire le istruzioni di Ubuntu per l'installazione.

4.3 Guida a OpenCV

OpenCV è una libreria open-source per computer-vision che semplifica notevolmente la programmazione C/C++. Include molte capacità avanzate di *face detection, face tracking, face recognition, Kalman filtering, e vari metodi di intelligenza artificiale*, pronti all'uso.

Si può scaricare da http://sourceforge.net/projects/opencvlibrary/
Una buona guida per imparare ad utilizzare questa libreria è
Learning OpenCV: Computer Vision with the OpenCV Library.
Intel rilasciò la prima versione nel 1999. Inizialmente richiedeva Intel Image Processing Library ma ora è possibile usarla come una libreria autonoma. OpenCV è multi-piattaforma. Esso supporta sia Windows che Linux e, più recentemente, MacOSX. Una guida per installarla è
http://opencv.willowgarage.com/wiki/InstallGuide?highlight=((InstallGuide))

Le caratteristiche principali sono :

- General computer-vision and image-processing algorithms (mid- and low-level APIs)
- High-level computer-vision modules
- AI and machine-learning methods
- Image sampling and view transformations
- Methods for creating and analyzing binary (two-valued) images
- Methods for computing 3D information
- Math routines for image processing, computer vision, and image interpretation
- Graphics
- GUI methods
- Datastructures and algorithms
- Data persistence

Si può scaricare la versione più recente da
http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.1/

I prossimi comandi saranno eseguiti da terminale, per vedere la guida completa di esempi si può seguire questo link

Prima di tutto è necessario installare alcuni software per le dipendenze di immagini e video con il comando
`sudo apt-get install build-essential libgtk2.0-dev libavcodec-dev libavformat-dev libjpeg62-dev libtiff4-dev cmake libswscale-dev libjasper-dev`

Poi scarichiamo la OpenCV
```
cd ~
wget http://sourceforge.net/projects/opencvlibrary/files
/opencv-unix/2.1/OpenCV-2.1.0.tar.bz2/download
tar -xvf OpenCV-2.1.0.tar.bz2
cd OpenCV-2.1.0/
```

Passiamo alla compilazione con `cmake .` e poi `make`
e all'installazione con `sudo make install`

Ora dobbiamo configurare la libreria.
Prima, apriamo il file `opencv.conf` con il seguente codice
`sudo gedit /etc/ld.so.conf.d/opencv.conf`

Copiare la seguente riga `/usr/local/lib` e salvare.

Digitiamo `sudo ldconfig` per configurare la libreria.

Ora dobbiamo aprire un altro file:
`sudo gedit /etc/bash.bashrc`
Copiamo queste righe e poi salviamo e chiudiamo il file:
```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

Ora apriamo un altro terminale, riavviamo il computer, perchè le impostazioni siano effettive.

4.4 Guida alle Qt-4.4.3

Questa è la versione Qt 4.4.3.
Qt è un cross-platform C++ application framework Qt 4.4 che introduce nuove caratteristiche e numerosi miglioramenti rispetto alla serie 3.x. Vedere http://doc.trolltech.com/4.4/qt4-intro.html per i

dettagli.

La serie Qt 4.x non è compatibile a livello binario o sorgente compatibile con la serie 3.x. Per ulteriori informazioni sul porting da Qt 3 a Qt 4, vedere `http://doc.trolltech.com/4.4/porting4.html`

Per installare le Qt si procede nel seguente modo:

- Scompattare il tar contenente le Qt nella cartella in cui si vuole installare;
- lanciare il comando `./configure`
- di seguito `make` (potrebbe richiedere diverse ore)
- `sudo make install`
- per la configurazione nel sistema aprire il file `.profile` e inserire alla fine le seguenti righe
  `PATH=/usr/local/Trolltech/Qt-4.4.3/bin:$PATH`
  `export PATH`

4.5 Guida installazione monitor touchscreen

Per l'installazione abbiamo collegato il touchscreen al Pc sia con la porta RS232 che con il cavo USB che permette la configurazione manuale del dispositivo.

Scaricare il xinput_calibrator nella propria home(preferibilmente il pacchetto .deb)

Installare con UbuntuSoftwareCenter il software.

Per migliorare la precisione della calibrazione si deve eseguire per tre volte il programma che comparirà dopo l'installazione nel menù Amministrazione.

Copiare il codice che risulta dalla prima calibrazione nella cartella `xorg.conf.d` che dovrete creare, nel file che indica il programma

Fate la seconda calibrazione.

Copiate il risultato della terza calibrazione in coda al primo risultato della calibrazione sempre nella cartella `xorg.conf.d` nel file indicato dal programma.

4.6 Strumenti utilizzati durante il processo di sviluppo

Il software nasce come applicazione scritta utilizzando il linguaggio di programmazione C. Durante lo sviluppo ci si è resi conto che tale linguaggio era troppo limitato per ciò di cui necessitava una applicazione di riconoscimento del volto, completa di tutte le funzionalità descritte nei paragrafi precedenti. Ciò ha portato l'autore dell'applicazione a utilizzare il linguaggio C++. Tale scelta è motivata dal fatto che l'utilizzo del *paradigma ad oggetti* ha semplificato enormemente il processo di sviluppo permettendo di introdurre soluzioni più rapide ed eleganti. La stessa necessità di un'interfaccia utente usabile ha concorso alla scelta dell'utilizzo di tale linguaggio. Oltre a ciò necessario ricordare l'utilizzo delle librerie OpenCV di Intel, è un'ottimo strumento per la creazione di applicazioni dell'ambito della Visione Artificiale. L'elenco che segue riassume tutti i tool e gli strumenti utilizzati durante il processo.

- ***Qdevelop***- IDE, ambiente integrato per lo sviluppo di applicazioni C/C++, particolarmente orientato all'uso congiunto delle librerie *Trolltech Qt4* per lo sviluppo di interfacce utente;[87]
- ***gcc*** - Compilatore C/C++, versione [88]
- ***Qt Designer*** -Designer di interfacce utente; [89]
- ***Trolltech Qt4*** - Librerie e classi C++ utili per lo sviluppo di interfacce utente multipiattaforma[90]
- ***Intel OpenCV*** - Libreria Open Source di funzioni dedicate alla Computer Vision, rilasciata da Intel [91]
- ***Webcam Trust SpaceCm 150 Portable*** [92]
- ***Xinput_calibrator*** - Software per la calibrazione del touchscreen rilasciato da [93]
- ***Buttons*** - Per la creazione dei pulsanti usati nelle interfacce [94]

Capitolo 5

Funzionamento del programma

L'applicazione creata è il risultato di quanto esposto nei capitoli precedenti per quanto riguarda il riconoscimento del volto effettuato tramite il metodo dell'analisi delle componenti principali (Eigenfaces).
Si tratta sostanzialmente di un'applicazione in grado di effettuare il riconoscimento di volti di un'insieme di persone classificate in un database predisposto precedentemente. L'applicazione si occupa inoltre di effettuare compiti di *face detection* su di una sequenza di video catturata tramite webcam, in modo da poter localizzare il viso della persona che si vuole riconoscere tramite il metodo PCA.
Non è costituita da un singolo applicativo, bensì consta di tre diversi programmi utili per il funzionamento e la manutenzione dell'intero sistema. La prima parte si occupa di instanziare e visualizzare la prima interfaccia, che permette all'utente di fare due diverse scelte: la prima di creare un utente e di inserire le immagine dell'utente stesso nel database; la seconda di effettuare il riconoscimento. La cartella del progetto contiene 4 sottocartelle:

- `img` che contiene tutte le immagini usate per le interfacce;
- `Menuprova` che contiene il menu del programma dove l'utente sceglie tra la creazione di un nuovo utente e il riconoscimento;
- `OffRecognition` contiene la parte per il riconoscimento;
- `RecognitionProject` contiene la parte per creare un nuovo utente o aggiornare le foto.

Nelle prossime pagine segue una descrizione generale dell'intero sistema, con "sguardi" al codice sorgente nelle sue parti maggiormente inte-

ressanti, complesse e con tutte le indicazioni per l'uso e la manutenzione dell'intera applicazione.

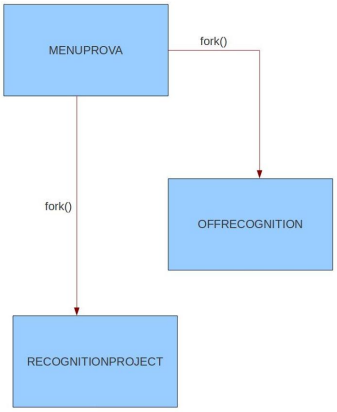

Figura 5.1: Struttura programma

5.1 Menuprova

Menuprova permette di accedere alle altre due parti del programma attraverso i pulsanti **Crea Database** e **Esecuzione Riconoscimento**

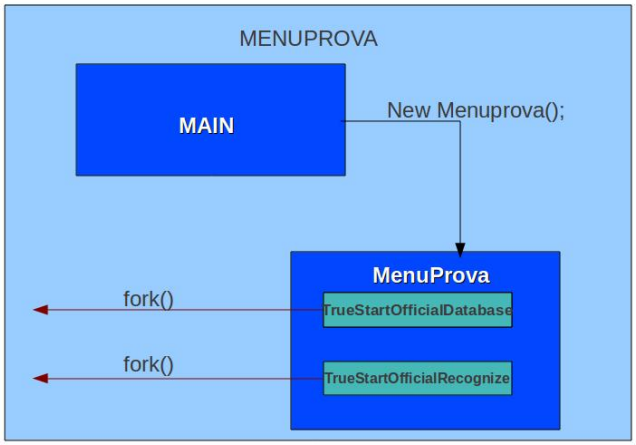

Figura 5.2: Struttura Menuprova

5.2 Creazione Database

É una sezione del programma dedito unicamente a compiere la localizzazione di volti in una sequenza video tramite webcam. Il suo scopo principale è quello di salvare su disco le immagini dei volti in formato *pgm*, in modo da poter comporre successivamente il database dei volti delle persone che possono essere riconosciute. Il database dei volti sarà quindi costituito da una serie di directories contenenti esattamente 10 immagini di volti per ogni persona che viene inserita. Ogni immagine sarà in formato *pgm* ed in particolare di dimensione 92×112 pixels. Nonostante ciò effettua alcune elaborazioni sulle immagini che vengono catturate; tali elaborazioni consistono in una conversione in scala di grigi del modello del colore e in una equalizzazione dell'istogramma dell'immagine che comporta una riduzione della luminosità e un aumento del contrasto. Tutti i file devono essere nominati secondo la seguente sintassi:

```
[num].pgm
```

che non è altro che il numero progressivo da 1 a 10, automaticamente assegnato dall'applicazione.

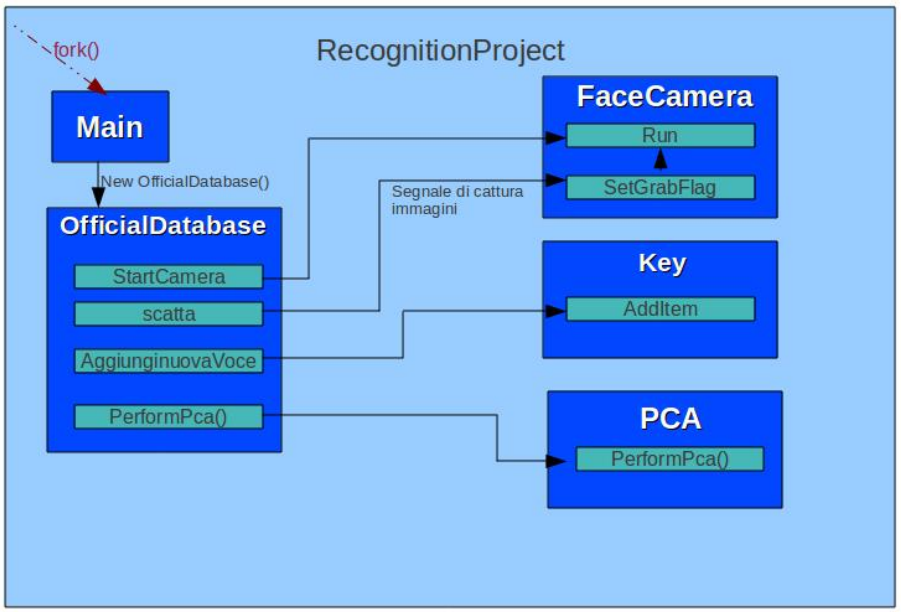

Figura 5.3: Struttura RecognitionProject

Quando clicchiamo su `Crea Database` all'interno del programma avviene una *fork* che crea un processo figlio, il quale va ad aprire il file eseguibile `RecognitionProject` all'interno della medesima cartella.
RecognitionProject ci dà la possibilità di creare la cartella di un utente e di inserirvi al suo interno delle immagini che verranno utilizzate poi dalla parte di riconoscimento. All'interno di questa parte inoltre una volta che le foto vengono salvate, il programma attraverso il file `PCA.cpp` calcola e aggiorna tutte le componenti PCA delle immagini presenti nel database. La cartella relativa a questa funzione è la `RecognitionProject`.
Analizziamola nel dettaglio.

All'interno troviamo la cartella `Database` che contiene tutte le sottocartelle dei vari utenti registrati con le loro immagini.

5.2.1 FaceCamera.cpp

Contiene le istruzione di instanziazione e utilizzo della webcam, la quale è sempre in collegamento con il file OfficialDatabase. Questa istruzione è lanciata quando premo il pulsante avvia camera, che corrisponde in sostanza all'attivazione di un flag *camNeed*, il quale ci permette di ciclare all'interno delle istruzioni di cattura dei frame dalla webcam. All'interno di questo ciclo inoltre avviene la face detection, e su ogni frame la webcam disegna un rettangolo intorno al volto, e tali frame vengono poi spediti al file principale che si occupa di visualizzarli. Per quanto riguarda invece l'acquisizione e il salvataggio delle immagini, quello che avviene all'interno del nostro programma è il semplice cambiamento di un flag *grabFlag* che salva l'immagine. Ovviamente dopo il salvataggio viene subito rimesso a false, fino alla nuova pressione del tasto `AcquisisciImmagine` per evitare che la webcam salvi ogni frame. Quando la webcam non serve più, si disattiva il flag *camNeed* che ci fa uscire dal ciclo di istruzioni di acquisizione.

Proponiamo di seguito il codice della sezione.

```
CvSeq* faces = cvHaarDetectObjects( frame_copy, cascade, storage, 1.1, 2, CV_HAAR_DO_CANNY_PRUNING, cvSize(40, 40) ); //cerco le facce nel frame
for (int counter = 0; counter < (faces ? faces->total : 0); counter++ ) {
    CvRect* r = (CvRect*)cvGetSeqElem( faces, counter ); //creo il rettangolo intorno al volto
        pt1.x = r->x*scale;
        pt2.x = (r->x+r->width)*scale;
        pt1.y = r->y*scale;
        pt2.y = (r->y+r->height)*scale;
        cvRectangle( frame_copy, pt1, pt2, CV_RGB(0,255,0), 3, 8, 0 );
        ROIheight = abs(pt2.y-pt1.y);   //ottengo le dimensioni della ROI
        ROIwidth = abs(pt2.x-pt1.x);    //ottengo le dimensioni della ROI
        ROIpt1 = pt1;

        //se devo catturare il frame e salvarlo su disco
        if (grabFlag == TRUE) {


            CvRect Roi = {ROIpt1.x,ROIpt1.y,ROIwidth,ROIheight};
            frame_copy->roi = NULL;
            cvSetImageROI(frame_copy,Roi); //setto la ROI sul frame

            //definisco una serie di copie del frame catturato su cui fare alcuni
            //passaggi intermedi di eleborazione dell'immagine
            IplImage* frame_grabbato=cvCreateImage(cvSize(ROIwidth,ROIheight),IPL_DEPTH_8U,3);
            IplImage* frame_grabbato_resize=cvCreateImage(cvSize(92,112),IPL_DEPTH_8U,3);
            IplImage* frame_grabbato_gray=cvCreateImage(cvSize(92,112),IPL_DEPTH_8U,1);

            cvCopy (frame_copy, frame_grabbato, 0);
            cvResize(frame_grabbato, frame_grabbato_resize,CV_INTER_LINEAR); //Ridimensiono l'immagine alle dimensioni delle immagini del

            cvCvtColor(frame_grabbato_resize,frame_grabbato_gray,CV_BGR2GRAY); // converto in livelli di grigio
            cvEqualizeHist(frame_grabbato_gray, frame_grabbato_gray);      //effettuo equalizzazione dell'istogramma (diminuisco luminosità e

            sprintf(nomeimg,"%d.pgm", count);

            //strcat(str,dir);
            //chdir("/usr/local/Progetti/OffRecognition/Database");
            //chdir(str);
            // printf("Dir %s",str);

            cvSaveImage( nomeimg, frame_grabbato_gray); // salvo il frame su disco
```

Figura 5.4: Salvataggio eigenface

5.2.2 key.cpp

La classe key è utilizzata per costruire la tastiera che verrà presentata all'utente quanto verrà eseguita la registrazione per inserire i dati (nome e cognome) per poi proseguire con la fase di training delle immagini. La classe contiene quindi tutte le funzioni per abilitare le lettere della tastiera.
La funzione più importante della classe è `exit()` in cui avviene la creazione della cartella dell'utente che conterrà le immagini salvate in questa fase.

```
void Key::exit()
{
        if(count == 3)
        {
        if(labelDisplay->text() == "y")
        {
                QChar sostitute = nome_utente.at(0).toUpper();
                nome_utente[0] = sostitute;
                QChar surnameLetter = user_surname.at(0).toUpper();
                user_surname[0] = surnameLetter;


                directory = nome_utente.append(user_surname);
                QDir *save=new QDir();
                QDir *save2=new QDir();
                save->mkdir("/usr/local/Progetti/OffRecognition/Database/" + directory);
                save2->mkdir("/usr/local/Progetti/OffRecognition/Total/" + directory);
                // Invio la directory appena creata (ovvero il nome e il cognome dell'utente)
                // all'interno della lista degli utenti contenuti nel database!
                // Lista presenta sulla finestra principale del programma!!!
                emit(ne(directory));
                count = 0;
                firstDigitLetter = FALSE;
                hide();
        }
        else
        {
                labelDisplay->clear();
                QFont font("Helvetica", 8, QFont::Bold);
                labelDisplay->setFont(font);
                labelDisplay->setText("[y] per confermare l'operazione, ([n] altrimenti)!");
                firstDigitLetter = FALSE;
                count = 0;
        }
        }

}
```

Figura 5.5: Funzione exit()

5.2.3 OfficialDatabase.cpp

L'interfaccia OfficialDatabase è composta dai pulsanti che guidano l'utente nell'inserimento di nuovi soggetti nel database. In particolare, di rilevante importanza sono i tasti `AggiungiUtente`, `AvviaCamera`, `AcquisisciImmagine`. Il primo consente di visualizzare l'interfaccia della tastiera (la quale è già attiva quando viene istanziato OfficialDatabase ma è nascosta) che dà all'utente la possibilità di inserire prima il nome e poi il cognome del soggetto da registrare e poi un'ulteriore conferma di salvataggio che rende il sistema di registrazione ancora più sicuro. É interessante notare che la creazione vera e propria della cartella del soggetto viene effettuata alla chiusura dell'interfaccia della tastiera.
Parallelamente a questo salvataggio avviene la creazione di un'altra cartella per il soggetto in questione che non verrà utilizzata per il training ma per registrarne data e ora dell'entrata (una volta che è stato riconosciuto dal sistema).
Una volta chiusa l'interfaccia della tastiera, e quindi creata la cartella, si passa all'acquisizione delle immagini. Viene aggiornato la `QCombobox` che ci permette di scegliere la cartella dove andare a salvare le immagini che saranno acquisite. Per acquisire le immagini usiamo il file `faceCamera.cpp` che comunica con il file `OfficialDatabase` il quale lancia il segnale *run* che abilita la webcam. Quando questa è attiva, man mano che le immagini (frame) scorrono, il programma effettua la face detection evidenziano la faccia del soggetto e passa tutti i frame al file `OfficialDatabase` che li visualizza.
Nel momento in cui *facecamera* riceve il segnale di *grabFlag*, facecamera cattura il frame e salva l'immagine contenuta all'interno del rettangolo facendo alcune elaborazioni come la conversione di grigi dell'immagine e una scalatura.

5.2.4 Pca.cpp

La classe permette di applicare il metodo dell'Analisi delle Componenti Principali ad un database di volti creato precedentemente. L'aspetto teorico quindi viene calcolato da questa piccola classe. Di fatto fornisce come output, che andrà poi copiato nella cartella in cui risiede, la seguente serie di file e directories:

- `/avg`: la cartella contiene il file `avg.xml` che non è altro che il vettore medio di tutte le immagini del database in seguito alla loro rappresentazione sotto forma di vettore

- `/eigens`: la cartella contiene le cosiddette "eigenfaces", ovvero tutti gli autovettori della matrice di covarianza del training set in questione. Anche in questo caso sarà salvato in formato XML.
- `Autoval.dat`: il file contiene unicamente la lista di tutti gli autovalori della matrice di covarianza ordinati in maniera decrescente.
- `Decompcoeffs.dat`: il file contiene i cosiddetti coefficienti di decomposizione ottenuti dal calcolo della PCA sul training set dei volti. Questi coefficienti non sono altro che i "pesi" utilizzati per il riconoscimento dei volti. Da un punto di vista matematico corrispondono alla differenza ottenuta da ogni immagine con l'immagine media, moltiplicata per ogni autovettore ricavato dalla matrice di covarianza.

Il metodo `PCA` costituisce il cuore di questa classe. Il suo scopo infatti, è quello di effettuare l'Analisi delle Componenti Principali sul database di volti creato in precedenza. Fornisce in output tutto ciò che è necessario per descrivere un "modello" del database da utilizzare successivamente. Ciò significa che l'applicazione non possiederà internamente una copia del database con tutte le immagini che lo compongono, bensì farà uso di questa particolare descrizione, costituita dall'immagine media, le *eigenfaces* (autovettori) e i relativi autovettori della matrice di covarianza ed infine i coefficienti della decomposizione che verranno salvate su disco in formato XML.

5.3 Esecuzione riconoscimento

É probabilmente la parte più importante di tutte, anche se il suo corretto funzionamento dipende fortemente da ciò che viene fornito in output dalla precedente.
I compiti principali di questa sezione sono quelli di permettere all'utente di catturare un'immagine contenente il suo volto e procedere con il proprio riconoscimento, in modo da compiere ciò che può essere definito un vero e proprio procedimento di autenticazione del soggetto che si trova davanti alla telecamera.

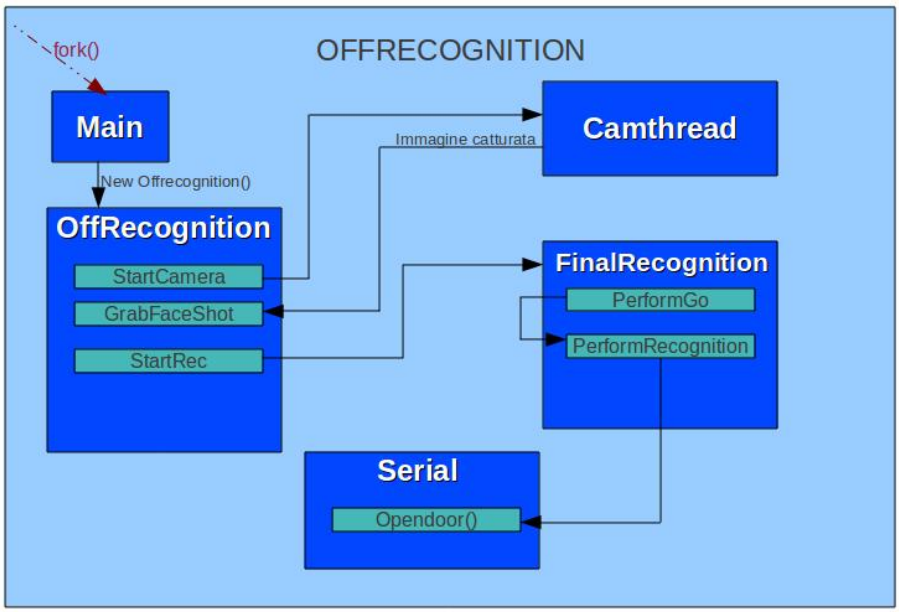

Figura 5.6: Struttura Riconoscimento

5.3.1 CamThread

Analogamente al file `FaceCamera.cpp` si occupa semplicemente della face detection e di salvare l'immagine rilevata.

5.3.2 offrecognition.cpp

Visualizza l'interfaccia che permette l'acquisizione dell'immagine del soggetto e ne permette il riconoscimento effettivo. La funzione principale di questo file è quella di attivare la webcam e comunicare con essa. La parte fondamentale avviene con il metodo `Startrec()` che richiama il file `finalrecognition.cpp`. Inizialmente questo file visualizza una finestra con un pulsante usata per il riconoscimento.

5.3.3 finalrecognition.cpp

Il suo compito è quello di proiettare l'immagine "target" da riconoscere nel sottospazio dei volti e valutarne la distanza euclidea da ogni singola immagine del database, in modo da compierne il riconoscimento. Le istruzioni si occupano principalmente di caricare da disco tutte le informazioni che descrivono il modello del database, ossia autovettori e relativi autovalori della matrice di covarianza, l'immagine media e i

coefficienti della decomposizione. Avviene poi la proiezione vera e propria dell'immagine da riconoscere. Poi Si calcola la distanza euclidea fra l'immagine "target" proiettata nel sottospazio dei volti e la proiezione di ogni singola immagine del database. Queste distanze vengono inserite in un vettore in particolare una sola di queste risulta utile per il processo di riconoscimento, ovvero quella con valore minore. Tale distanza infatti è associata all'immagine del database che è più simile a quella del soggetto che si vuole autenticare, esattamente come prevede il metodo PCA/Eigenfaces. Maggiore è il valore della distanza da un'immagine di un volto, più alta sarà la probabilità che il volto presente in quell'immagine non sia quello del soggetto da autenticare. Viene inoltre valutato l'identificatore univoco dell'utente tramite un'indice che punta alla lista dei soggetti registrati per verificare che coincida veramente con il soggetto da riconoscere. In questo modo oltre al riconoscimento salviamo anche il frame riconosciuto dal sistema per permettere la registrazione dell'accesso del soggetto al laboratorio. Infine avvenuto il riconoscimento la funzione lancia il comando `opendoor()` implementato nel file `serial.cpp` che realizza la reale funzione di apertura della serratura per l'accesso.

```
        printf("\nFinito!!!!!!!!!!!!!");
        printf("\n iNearest = %d\n", iNearest);
        char str[100]="/usr/local/Progetti/OffRecognition/Total/";
        if(((iNearest +10 )/ 10) == comboBoxID)//faccio il confronto per vedere se effettivamente è la persona cercata
        {

            strcat(str,stri);
            chdir(str);
            printf("%s\n",str);
            time_t     now;
            struct tm  *ts;
            char       buf[80];
            /* Get the current time */
            now = time(NULL);
            /* Format and print the time, "dd-mm-yyyy_hh:mm" */
            ts = localtime(&now);
            strftime(buf, sizeof(buf), "%d-%m-%Y_%H:%M", ts);
            //  salvataggio frame
            strcat(buf,".pgm");
            cvSaveImage(buf, tempTest);
            chdir("/usr/local/Progetti/OffRecognition/");

                opendoor();

                timer->start(50);
        }
        else
        {
                timerNegative->start(50);
        }

for(int i=0;i<strlen(str);i++){
                str[i]=NULL;
        }
        for(int i=0;i<strlen(stri);i++){
                stri[i]=NULL;
        }
}
```

Figura 5.7: Riconoscimento avvenuto e apertura porta

5.3.4 serial.cpp

```
void opendoor()
{
        int fd;                                     /* File descriptor per la porta seriale */

           fd = open("/dev/ttyACM0", O_RDWR); /*per conoscere il nome della porta assegnata dmesg*/
           if (fd == -1)
           {                                                  /* Se non riuscisse ad aprire la porta */
             fprintf(stderr, "open_port: Unable to open /dev/ttyACM0");
           }

           int n = write(fd, "A", 1); /*inviando A apre un rele' per un secondo, per aprire la porta*/

           if (n < 0)
              puts("write() di 1 bytes fallito!\n");

           close(fd);
}
```

Figura 5.8: Apertura porta

Come si può notare dal codice questa funzione permette la comunicazione tra il computer e la scheda relè per far scattare la serratura e aprire la porta.

Capitolo 6

Tutorial sul funzionamento

In questo capitolo si illustrano le operazioni necessarie al funzionamento del sistema di riconoscimento.
In generale, per il corretto funzionamento del sistema, è necessario fissare la webcam in modo che rimanga immobile durante tutte le operazioni.
É necessario inoltre che la stanza abbia un'illuminazione costante, dunque è conveniente usare fonti di luce artificiale.

6.1 Panoramica generale

Per avviare il programma cliccare sul link posto nel Desktop denominato `FaceRec`.
Una volta avviato il programma, l'interfaccia si presenta come in figura.

Figura 6.1: Menu

Tutte le funzioni sono accessibili attraverso i pulsanti.
Il programma può funzionare in due modalità: *training* e *riconoscimento*.
La modalità *training* serve per allenare il sistema a riconoscere un particolare soggetto; in questa modalità di funzionamento è possibile eseguire le operazioni seguenti:

- inserire i dati dell'utente di cui si devono raccogliere i campioni
- raccogliere i diversi campioni del volto
- elaborare i campioni raccolti calcolando la matrice di proiezione per la PCA a partire da essi. Questa è una operazione fondamentale per la successiva fase di riconoscimento, la quale utilizza la matrice di proiezione per confrontare le nuove immagini del volto con i campioni.

Il riconoscimento di nuovi volti avviene nella modalità *riconoscimento*; in questa modalità di funzionamento è possibile eseguire le seguenti operazioni:

- selezionare il nome del soggetto che deve essere riconosciuto;
- porre il soggetto da riconoscere di fronte la webcam e posizionarlo alla distanza corretta per il riconoscimento;
- verificare che il soggetto venga correttamente riconosciuto.

6.2 Modalità training

La fase di identificazione di un volto deve essere preceduta da una fase di allenamento (training) del sistema in cui si acquisiscono alcuni campioni del volto da riconoscere e si calcola la matrice di proiezione a partire da essi.

Per iniziare l'acquisizione delle immagini dalla webcam, necessaria per le fasi successive, si deve premere il pulsante `CreazioneDatabase`.

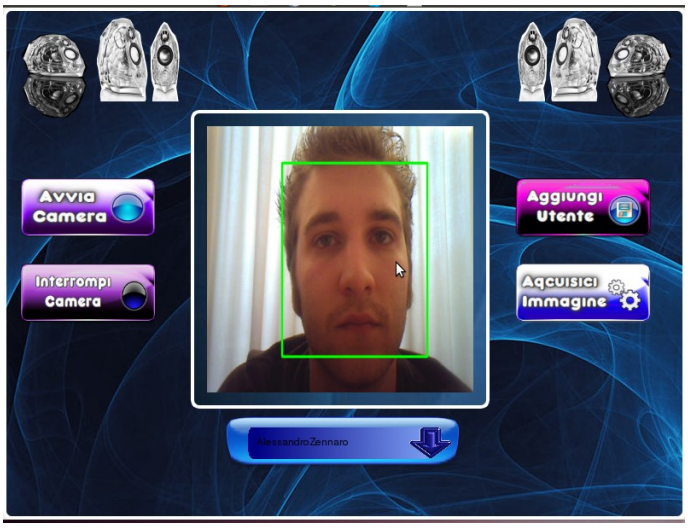

Figura 6.2: Registrazione Utente

É necessario prima creare l'utente premendo il pulsante `AggiungiUtente`. Si aprirà così la tastiera per inserire i dati dell'utente (Nome, Cognome) e successivamente la conferma della correttezza di tali dati.

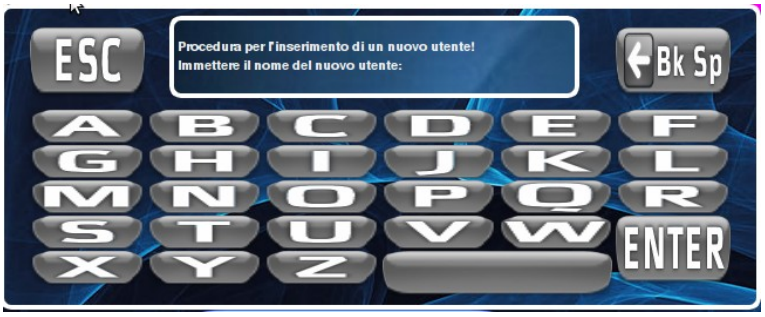

Figura 6.3: Inserimento dati utente

Il soggetto ora deve premere il pulsante `AvviaCamera` e posizionarsi in modo da far corrispondere i lati del rettangolo verde con il contorno del volto. Raggiunto l'allineamento corretto si può procedere al salvataggio di campioni del volto premendo il pulsante `AcquisisciImmagine`. É necessario mantenere la posizione per circa 5 secondi mentre i campioni vengono salvati; può essere opportuno variare leggermente l'espressione degli occhi in modo da rendere il training-set più ricco. Premere ancora `AcquisisciImmagine` fino ad ottenere un totale di 10 immagini per ogni persona. Infine per permettere il salvataggio premere `InterrompiCamera`. Il sistema è ora allenato per riconoscere il soggetto!

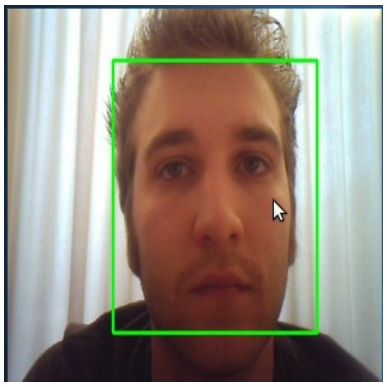

Figura 6.4: Allineamento corretto

6.3 Modalità Riconoscimento

Per entrare in modalità *Riconoscimento* si deve usare il pulsante `EsecuzioneRiconoscimento` del menù principale.

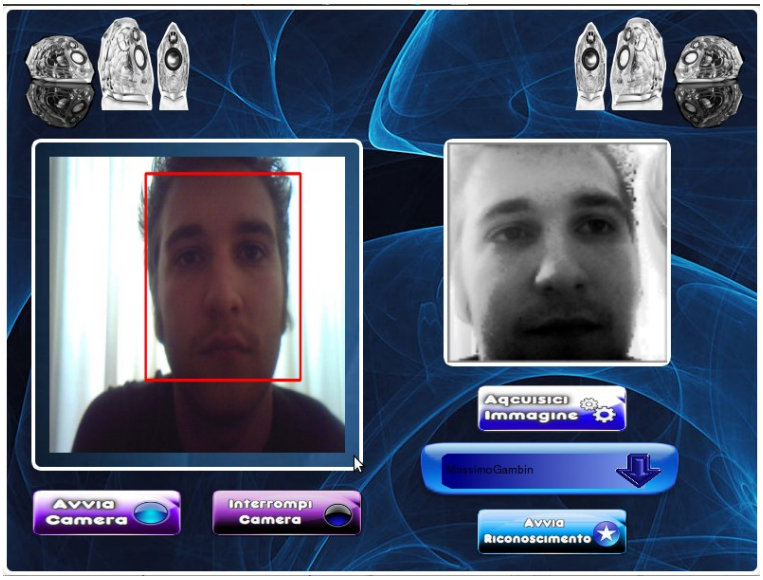

Figura 6.5: Fase di riconoscimento

É necessario selezionare il nome dell'utente dalla lista presente nell'interfaccia per scegliere il soggetto. Posizionarsi correttamente di fronte alla camera premere di nuovo `AvviaCamera` per dialogare con la webcam. Scattare quindi una foto con il pulsante `AcquisisciImmagine` e premere `AvviaRiconoscimento` e dare poi conferma alla finestra successiva premendo `Riconosci`.

Figura 6.6: Riconoscimento avvenuto

Il tentativo di riconoscimento avviene solo se il soggetto si trova alla distanza corretta della webcam, cioè se il volto è alla stessa scala dei campioni usati per il riconoscimento. Una volta effettuato il riconoscimento, benvenuti nel laboratorio VIPS!!!

Capitolo 7

Risultati

In questo capitolo analizziamo i principali risultati raggiunti con il lavoro di tesi e descriviamo il presente stato del lavoro.
Nel primo paragrafo si presentano i risultati ottenuti con il sistema.
Nel secondo si spiega come il sistema sia stato messo alla prova per il riconoscimento di alcuni volti umani. Nel terzo paragrafo si espongono i limiti evidenziati dal sistema e le condizioni necessarie per il suo miglior funzionamento, portando alcuni esempi in cui il sistema può fallire e nell'ultimo i problemi riscontrati durante lo sviluppo.

7.1 Risultati ottenuti

Il sistema che abbiamo sviluppato è in grado di acquisire immagini di training di una o più persone e validare l'identità delle stesse su immagini nuove (che non appartengano al training). Gli aspetti dell'architettura, dell'interfaccia, dell'elaborazione della sequenza e del fotogramma sono soddisfacenti e garantiscono un'elaborazione dei frame in tempo reale.
Con accorgimenti minimi essi producono risultati gradevoli anche con immagini acquisite da webcam di qualità medio-bassa. La fase di riconoscimento di persone è invece prototipale; per il momento fornisce risultati soddisfacenti solo nel caso in cui le differenze tra training e test siano piccole.
In un caso del genere, come vedremo nel prossimo paragrafo, il sistema è riuscito a distinguere sei soggetti diversi, senza alcun falso positivo o negativo.

7.2 Un test effettuato sul sistema

Il sistema di riconoscimento è stato testato con un gruppo di sei soggetti.
Dopo aver creato un training-set per ciascun soggetto, si è passati alla fase di riconoscimento usando la seguente procedura:

- inizializzazione del sistema per il riconoscimento di uno dei soggetti;
- prova di riconoscimento per tutti i sei soggetti.

Ovviamente ci si aspettava che solo uno dei sei soggetti, quello per cui il sistema era stato inizializzato, fosse riconosciuto, mentre gli altri cinque venissero rifiutati. La procedura è stata ripetuta inizializzando il sistema via via con tutti i sei soggetti.
Nella tabella della pagina seguente è possibile vedere i risultati delle 36 prove di riconoscimento effettuate in tal modo.

	 Cristina	 Massimo	 Omar	 Chiara	 Mauro	 Jhon
Cristina	4.17	3.46	0.61	2.37	3.07	1.83
Massimo	3.46	0.71	2.85	1.09	0.39	1.63
Omar	0.61	2.85	3.56	1.76	2.46	1.16
Chiara	2.37	1.09	1.76	1.8	0.7	0.54
Mauro	3.07	0.39	2.46	0.71	1.1	1.24
Jhon	1.83	1.63	1.22	0.54	1.24	2.34

Tabella 7.1: Testi di riconoscimento basato su sei soggetti

I numeri indicano la misura di distanza calcolata fra i modelli e i test. Sulla diagonale, evidenziate in blu, le misure di distanza calcolate nei casi in cui come test si usava lo stesso soggetto del modello.

7.3 Limiti del sistema

Parlando di limiti del sistema di riconoscimento, la prima considerazione da fare è che il sistema necessita della collaborazione dell'utente. Come si è visto precedentemente, sia nella fase di training che in quella

di riconoscimento una corretta interazione dell'utente con il sistema è fondamentale per il suo funzionamento. Nella fase di training il soggetto si deve allineare in maniera molto precisa per acquisire i campioni e questo implica non solo che egli voglia collaborare, ma anche che riesca a farlo nella maniera giusta.
Anche in fase di riconoscimento, seppure in misura minore, è importante la collaborazione dell'utente, poiché come detto il riconoscimento viene tentato solo se il soggetto è posto alla distanza giusta dalla webcam. Esistono poi dei limiti operativi del sistema, che illustreremo in questo paragrafo. Come abbiamo visto precedentemente, il sistema è il risultato di una serie di operazioni compiute sui fotogrammi acquisiti dalla webcam; ognuna di queste operazioni, se effettuata in maniera errata, può pregiudicarne il funzionamento. Distinguiamo i limiti del sistema fra quelli dovuti alle operazioni di pre-elaborazione compiute sulle immagini e quelli dovuti all'algoritmo di riconoscimento.

7.3.1 Limiti dell'algoritmo di riconoscimento

Anche l'algoritmo di riconoscimento necessita di particolari condizioni per funzionare correttamente. Come si è visto nei capitoli precedenti, il riconoscimento di un volto si basa fondamentalmente su una misura di distanza fra questo e un insieme di campioni.
A parte l'equalizzazione e la conversione in scala di grigi , utili per la localizzazione del volto all'interno dell'immagine, nessun' altra operazione viene compiuta sull'immagine del volto, la quale viene confrontata con i campioni così com'è; questo procedimento ha il vantaggio di essere molto semplice e di permettere una rapida elaborazione di ogni fotogramma, fondamentale per il funzionamento in tempo reale del sistema. D'altra parte è ovvio che le condizioni generali in cui il volto si mostra al sistema per essere riconosciuto devono essere circa uguali a quelle presenti al momento del training.
Caratteristiche come l'espressione del volto, la rotazione orizzontale e verticale dello stesso, la presenza o meno di occhiali e l'illuminazione devono restare costanti tra la fase di acquisizione dei campioni e la successiva fase di riconoscimento. In particolare il sistema si è mostrato molto sensibile alle variazioni di illuminazione, siano esse globali o legate alle diverse angolazioni di incidenza della luce sul soggetto. È intuitivo infatti come una variazione dell'illuminazione determini un cambiamento nel livello di grigio dei pixel dell'immagine. Pensando l'immagine come vettore, variazioni di questo tipo hanno l'effetto di cambiare in modo cospicuo le componenti di quest'ultimo, falsando completamente la misura di distanza effettuata e dunque impedendo il corretto funzionamento del sistema di riconoscimento. Esistono poi

limiti di tipo statistico al metodo di classificazione usato. Senza entrare nel dettaglio possiamo dire che esso usi solo esempi positivi per ogni classe e per questo motivo potrebbe perdere efficacia nel caso in cui le classi fra cui distinguere diventino molte, oppure due persone diverse si somiglino in maniera particolare.

7.4 Problemi riscontrati durante lo sviluppo

Quanto scritto precedentemente, permette di avere un'idea di quanto sia complessa l'applicazione creata. Inevitabile dire che il processo di sviluppo ha richiesto molto tempo e lavoro. Tuttavia sono state riscontrare molte difficoltà risoltesi comunque in maniera positiva, escogitando spesso soluzioni particolari e interessanti. Elencarla tutte richiederebbe una trattazione a parte, quindi ci si limiterà a riportare esclusivamente quelle maggiormente degne di nota.
Questi applicativi sono orientati all'ambito della visione artificiale, e come tali si pongono l'obiettivo di compiere elaborazioni su immagini digitali. Di fatto, l'utilizzo della libreria *Intel OpenCV*, ha semplificato enormemente lo sforzo del programmatore. Nonostante ciò sono state riscontrate, ad esempio difficoltà nell'installare il monitor touchscreen per la difficile funzione di calibrazione, il salvataggio delle immagini di una particolare classe di immagini su disco; gli *eigenface*. Questi autovettori sono immagini a tutti gli effetti e rappresentano lo strumento fondamentale per il riconoscimento del volto. La libreria *OpenCV* fa uso di un proprio formato di immagini, denominato `IplImage`, descritto da una struttura contenente una serie di campi. Uno dei più importanti è il campo `depth` che contiene informazioni riguardo al tipo di valore dei pixel dell'immagine. Questo campo può assumere i seguenti valori:

- `IPL_DEPTH_8U`: intero senza segno a 8-bit;
- `IPL_DEPTH_8S`: intero con segno a 8-bit;
- `IPL_DEPTH_16S`: intero con segno a 16-bit;
- `IPL_DEPTH_32S`: intero con segno a 32-bit;
- `IPL_DEPTH_32F`: virgola mobile in singola precisione 32-bit;

Il salvataggio delle immagini su disco viene sempre effettuato mediante il metodo `cvSaveImage()`, il quale permette unicamente di salvare su disco immagini con il valore del campo sopracitato, pari a `IPL_DEPTH_8U`. Le istruzioni che compongono l'implementazione della

PCA, invece forniscono necessariamente in output immagini con profondità del tipo `IPL_DEPTH_32F`.
Ci si è accorti immediatamente che questo fatto comportava delle difficoltà in quanto il salvataggio delle immagini (per esempio eigenfaces) con profondità a 32-bit richiede necessariamente una conversione del formato del campo `depth`, con conseguente perdita di informazione. Per ovviare a questo inconveniente si è deciso di utilizzare l'istruzione `cvSave()`, salvando direttamente su disco le matrici numeriche delle immagini con profondità a 32-bit all'interno del formato XML. In questo modo, al momento del loro utilizzo, basta solamente invocare il metodo `cvLoad()` per caricare le matrici numeriche all'interno di immagini tipo `IplImage`. Ciò ha di fatto evitato qualsiasi perdita di informazione e permesso inoltre un notevole risparmio di memoria.
Un'altra difficoltà incontrata, consisteva che le librerie Qt di Trolltech [87], tramite le quali sono state implementate le interfacce utente, possiedono a loro volta, un proprio formato immagine. Tale formato si chiama `QImage` e di conseguenza è nata la necessità di creare una classe chiamata *qtipl*, la cui implementazione è riscontrabile all'interno di ogni applicazione che compone il software. Tale classe si occupa sostanzialmente di convertire ogni immagine (ad esempio i frame catturati dalla webcam) dal formato `IplImage` di *OpenCV* al formato `QImage`, in modo da permettere la visualizzazione nei *widgets* che compongono le interfacce utente.
Si è inoltre reso necessario porre attenzione all'implementazione stessa di queste interfacce, cercando, per quanto possibile, di facilitarne l'interazione con gli utenti. L'obiettivo è stato, infatti, quello di renderle più possibili usabili, intuitive e gradevoli.

L'altro problema è stato l'installazione del monitor touchscreen in quanto, una volta formattato il Pc per aggiornarlo, non riconosceva più il monitor. Abbiamo dovuto quindi scaricare un driver diverso da quello fornitoci e installare il pacchetto `.deb` in quanto tutte le guide trovate non risolvevano il problema che riscontravamo, ossia la calibrazione non veniva eseguita correttamente. Con il pacchetto invece, una volta installato, la calibrazione è stata eseguita facilmente. Essendo però una versione beta abbiamo dovuto copiare nel file di configurazione i dati delle diverse calibrazioni per renderle effettiva anche una volta che il Pc veniva spento o semplicemente veniva ripresa la sessione.

Capitolo 8

Conclusioni e sviluppi futuri

Nei prossimi paragrafi si traggono le conclusioni del lavoro svolto. Si analizza il punto di partenza e i risultati a cui si puntava, si descrive il punto a cui si è arrivati e le possibili strade da intraprendere con sviluppi futuri.

8.1 Conclusioni

Lo scopo del lavoro era la realizzazione di un sistema completo e funzionante per il riconoscimento di volti umani in tempo reale. Per raggiungere questo obiettivo si è scelto di utilizzare semplici tecniche di visione computazionale e apprendimento statistico che garantissero un buon compromesso fra rapidità di esecuzione ed efficacia. Possiamo dire che l'obiettivo è stato raggiunto in maniera piuttosto soddisfacente. Il sistema sviluppato è in grado di imparare a riconoscere il volto di un soggetto da un insieme di campioni dello stesso acquisiti precedentemente.
Se le variazioni fra i campioni acquisiti e il nuovo volto da riconoscere sono poche, cioè le condizioni di illuminazione restano costanti e non intervengono cambiamenti nell'aspetto del volto, il sistema è efficace nel riconoscimento. Si è arrivati ad un sistema funzionante in tempo reale con concrete possibilità a livello applicativo, seppure piuttosto limitate allo stato attuale del lavoro.
Questo rappresenta una ottima base di partenza per lo sviluppo e la ricerca di soluzioni alternative e maggiormente efficaci per i vari moduli componenti il sistema. Nel prossimo paragrafo indagheremo proprio gli aspetti legati a ciò che si può concretamente pensare di ottenere dal sistema a cui siamo giunti con il presente lavoro.

8.2 Sviluppi futuri

Il sistema mostra alcuni limiti nel caso in cui le variazioni fra il training-set e le nuove immagini del volto siano cospicue. Allo stato attuale il sistema può essere impiegato per applicazioni che non prevedano una grande variazione nei soggetti e nella scena (tipicamente quindi sessioni di lavoro limitate nel tempo) e non richiedano la gestione di molte persone diverse. Un esempio: ausilio visivo alla verifica della password per l'accesso ad una sessione di lavoro su PC. È noto che gli accessi tramite password siano tuttora il miglior compromesso tra sicurezza e semplicità e che i sistemi biometrici non siano ancora riusciti a proporre soluzioni semplici e convincenti. Il nostro sistema, poiché basato su una fase di apprendimento, cioè sulla raccolta di un insieme di campioni del volto da riconoscere successivamente, potrebbe rappresentare una soluzione efficace per un contesto del genere. Sarebbe possibile infatti raccogliere campioni del volto dell'utente mentre questo lavora al PC, in modo del tutto trasparente. Il sistema in tal modo verrebbe allenato al riconoscimento del volto dell'utente, nelle condizioni di illuminazione e di aspetto della sessione corrente. Mentre l'applicazione appena descritta può già essere alla portata del sistema nello stato attuale, esistono diverse altre applicazioni interessanti, che però richiedono ulteriori ricerche e sviluppi per il sistema. Un esempio potrebbe essere un dispositivo per il controllo degli ingressi in edifici o laboratori. La riuscita di tale applicazione è basata sulla capacità di validare un alto numero di persone distinte (per applicazioni realistiche almeno centinaia) ed essere in grado di gestire i cambiamenti dell'aspetto di una persona o dell'ambiente circostante. Per quest'ultimo problema occorre studiare compromessi tra la gestione di un'ampia variabilità di soggetti tramite training-set più ricchi e una fase di training più impegnativa ed invasiva. Sarebbe infatti teoricamente possibile richiedere a un centinaio di persone di farsi fotografare per l'addestramento del sistema in giorni e condizioni differenti, ma praticamente irrealizzabile. I fronti di ricerca su cui concentrarsi riguardano metodi diversi per il confronto fra immagini, ad esempio lo studio dell'estrazione di caratteristiche (feature) del volto che siano invarianti per cambiamenti di condizioni (luce, aspetto del soggetto), oppure metodi di arricchimento dei training-set tramite contributi sintetici, cioè modelli di volti costruiti artificialmente a partire da immagini di volti reali, con variazioni simulate di posa, aspetto e illuminazione.

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, *Twodimensional 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. Ersboll, 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.

[12] P.J. Phillips, H. Moon, P. Rauss, S.A. Rizvi. The FERET Evaluation Methodology for Face-Recognition Algorithms. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 833-838, 1998.

[13] Biometrika http://www.biometrika.it/wp_biointro.html *Introduzione ai sistemi biometrici*

[14] Face Recognition

[15] C. Wang, M. Brandstein. Multi-Source Face Tracking with Audio and Visual Data. IEEE MMSP, Pag. 168, 1999.

[16] E. Saber, A. Tekalp. Frontal-View Face Detection and Facial Feature Extraction using Colour, Shape and Symmetry based Cost Functions. Pattern Recognition Letters, Pag. 669-680, 1998.

[17] J. Gang, E. Sung. Frontal-View Face Detection and Facial Feature Extraction using Colour and Morphological Operations. Pattern Recognition Letters, Pag. 1053-1068, 1999.

[18] J. Brand, J. S. Mason. A Comparative Assessment of Three Approaches to Pixel-level Human Skin-Detection. Proc. Intl. Conf. Pattern Recognition, Pag. 1056-1059, 2000.

[19] A. Ogihara, A. Shintani, S. Takamatsu, S. Igawa. Speech Recognition based on the Fusion of Visual and Auditory Information using Full-Frame Colour Image. IEICE Trans. Fundamentals, Pag. 1836-1840, 1996.

[20] T. Wark, S. Sridharan. A Syntactic Approach to Automatic Lip Feature Extraction for Speaker Identification. ICASSP, Pag. 3693, 1998.

[21] M. Jones, J. Rehg. Compaq skin database.http://www.crl.research.digital.com/pubblications/techreports/abstracts/98_11.htm

[22] V.Bruce, P.J.B.Hancock, A.M.Burton. Human Face Perception and Identification. Face Recognition: From Theory to Applications, Pag.5172,1998.

[23] L.D. Harmon. The Recognition of Faces. Scientific American, vol.229, N. 5, Pag. 7182, 1973.

[24] A.P. Ginsburg. Visual Information Processing Based on Spatial Filters Constrained by Biological Data. AMRL Technical Report,Pag. 78-129, 1978.

[25] J. Sergent. Microgenesis of Face Perception. Aspects of Face Processing, 1986.

[26] H. Hill, P.G. Schyns, S. Akamatsu. Information and Viewpoint Dependence in Face Recognition. Cognition, vol. 62, Pag. 201-222,1997.

[27] H. Hill, V. Bruce. Effects of Lighting on Matching Facial Surfaces. Journal of Experimental Psychology: Human Perception and Performance, vol. 22, Pag. 986-1004, 1996.

[28] B. Knight, A. Johnston. The Role of Movement in Face Recognition. Visual Cognition, vol. 4, Pag. 265-274, 1997.

[29] V. Bruce. Recognizing Faces. London: Lawrence Erlbaum Associates. 1988.

[30] K. Sung, T. Poggio. Example-based Learning for View-based Human Face Detection. A.I. Memo 1521. A.I. Laboratory. MIT. 1994.

[31] H.A. Rowley, S. Baluja, T. Kanade. Neural Network Based Face Detection. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 20, 1998.

[32] H.A. Rowley, S. Baluja, T. Kanade. Rotational Invariant Neural Network-based Face Detection. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 38-44, 1998.

[33] M. Oren, C. Papageorgiu, P. Sinha, E. Osuna, T. Poggio. Pedestrian Detection Using Wavelet Templates. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 193- 199, 1997.

[34] T.K. Leung, M.C. Burl, P. Perona. Finding Faces in CLuttered Scene using Random Labeled Graph Matching. Proceedings, International Conference on Computer Vision, Pag. 637-644, 1995.

[35] K.C. Yow, R. Cipolla. Detection of Human Faces Under Scale, Orientation and Viewpoint Variation. Proceedings, International

Conference on Automatic Face and Gesture Recognition, Pag. 295-300, 1996.

[36] B. Moghaddam, A. Pentland. Probabilistic visual learning for object representation. IEEE Trans. PAMI, 19:696-710, 1997.

[37] A.J. Colmenarez, T.S. Huang. Face Detection with Information-Based Maximum Discriminant. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 782-787, 1997.

[38] N. Kruger, M. Potzsch, C.v.d. Malsburg. Determination of Face Position and Pose with a Learned Representation Based on Labelled Graphs. Image and Vision Computing, vol. 15, Pag. 665-673, 1997.

[39] E. Osuna, R. Freund, F. Girosi. Training Support Vector Machines: An Application to Face Detection. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 130-136, 1997.

[40] B.K. Low, E. Hjelmas. Face Detection: A Survey. Pattern Recognition, 1999.

[41] R. Chellappa, C.L. Wilson, S. Sirohey. Human and Machine Recognition of Faces, A survey. Proc. IEEE, vol. 83, Pag. 705-740, 1995.

[42] D. Reisfeld, Y. Yeshurun. Robust Detection of Facial Features by Generalized Simmetry. Proceedings, International Conference on Pattern Recognition, Pag. 117-120, 1992.

[43] P.W. Hallinan. Recognizing Human Eyes. SPIE Proceedings, vol. 1570: Geometric Methods in Computer Vision, Pag. 214-226, 1991.

[44] H. Moon, R. Chellappa, A. Rosenfeld. Optimal Edge-based Shape Detection. IEEE International Conference on Image Processing. Vancouver, BC, Canada, 2000

[45] A. Hyvarinen, E, Oja. Indipendent Component Analysis:Algorithms and Applications. Neural Networks, 13:411-430, 2000.

[46] A. J. Bell, T. J. Sejnowski. An information-maximization Approach to Blind Separation and Blind Deconvolution. Neural Computation, 7:1129-1159, 1995.

[47] K. Baek, B. A. Draper, J. R. Beveridge, K. She. PCA vs. ICA: A comparison on the FERET data set.

[48] G. Donato, et al. Classifyng Facial Actions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21:974-989, 1999.

[49] R. A. Fisher. The use of Multiple Measures in Taxonomic Problems. Ann. Eugenics, 7:179-188, 1936.

[50] R. Duda, P. Hart. Pattern Classification and Scene Analysis. New York: John Wiley & Sons. 1973.

[51] D. Swets, J. Weng. Using Discriminant Eigenfeatures for Image Retrieval. IEEE Trans. PAMI, 18:831-836, 1996.

[52] P. Belhumeur, J. Hespanha, D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Trans. PAMI, 19:711-720, 1997.

[53] V. Vapnik. Statistical Learning Theory. New York: John Wiley and Sons. 1998.

[54] B. Heisele, P. Ho, T. Poggio. Face recognition with support vector machines: global versus component- based approach. Proc. 8th International Conference on Computer Vision, volume 2, Pag. 688-

[55] B. Heisele, T. Poggio, M. Pontil. Face detection in still gray images. A.I. Memo 1687. Center for Biological and Computational Learning. MIT, Cambridge, MA. 2000.

[56] R. Rifkin. Everything old is new again: a fresh look at historical approaches in machine learning. MIT, 2002.

[57] C. Cortes, V. Vapnik. Support Vector Networks. Mach. Learning20, Pag. 1-25, 1995.

[58] M. Pontil, A. Verri. Support Vector Machines for 3-d object recognition. IEEE Trans. Pattern Anal. Mach. Intell., Pag. 637-646,1998.

[59] G. Guodong, S. Li, C. Kapluk. Face recognition by Support Vector Machines. Proc. IEEE Int. Conf. on Automatic Face and Gesture Recognition, Pag. 196-201, 2000.

[60] C. Nakajima, M. Pontil, T. Poggio. People recognition and pose estimation in image sequence. Proc. IEEE-INNS-ENNS International Joint Conf. on Neural Networks, vol. 4, Pag. 4189-4195, 2000.

[61] K.-K. Sung. Learning and Example Selection for Object and Pattern Recognition. Artificial Intelligence Laboratory and Center for Biological and Computational Learning, MIT, Cambridge, MA, 1996.

[62] Furio Cascetta, Marco De Luccia, *Sistemi di identificazione personale*, Mondo Digitale, n.1, Marzo 2004

[63] Intel Open Source Computer Vision Library http://www.intel.com/technology/computing/opencv/

[64] Lamberto Ballan, *Riconosciemnto automatico di volti utilizzando descrittori locali di caratteristiche locali* , Tesi di laurea, Università degli studi di Firenze, Facoltà di Ingegneria- Dipartimento di Sistemi e Informatica http://www.mic.unifi.it/ballan/research/ballan-thesis.pdf

[65] Sami Romdhani, *Face Recognition using PCA* Msc Thesis, University of Glasgow

[66] M.Turk, A.Pertland, *Eigenfaces for Recognition, Journal of Cognitive Neuroscience*, Vo.3, No. 1, 1991, pp.71-86 http://www.face-rec.org/algorithms/PCA/jcn.pdfP.J. Phillips, H. Moon, P. Rauss, S.A. Rizvi. The FERET

[67] Evaluation Methodology for Face-Recognition Algorithms. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 833-838, 1998.

[68] Y. Adini, Y. Moses, S. Ullman. Face Recognition: The Problem of Compensating for Changes in Illumination Direction. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 19, Pag. 721-732, 1997.

[69] W. Zhao, R. Chellappa. Robust Face Recognition using Symmetric Shape-from-Shading. Technical Report CAR-TR-919. Center for Automation Research. University of Maryland. 1999.

[70] W. Zhao. Robust Image Based 3D Face Recognition. University of Maryland, 1999.

[71] W. Zhao. Improving the Robustness of Face Recognition. Proceedings of International Conference on Audio- and Video-Based Person Authentication, Pag. 60-65, 1999.

[72] D.W. Jacobs, P.N. Bellhumeur, R. Basri. Comparing Images under Variable Illumination. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Pag. 610-617, 1998.

[73] A. Shashua. Geometry and Photometry in 3D Visual Recognition. MIT, 1994.

[74] P. Hallinan. A Low-Dimensional Representation of Human Faces for Arbitrary Lighting Conditions. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 995-999, 1994.

[75] H. Murase, S. Nayar. Visual Learning and Recognition of 3D Objects from Appearances. International Journal of Computer Vision, vol. 14, Pag. 5-25, 1995.

[76] P.N. Belhumeur, D.J. Kriegman. What is the Set of Images of an Object Under All Possible Lighting Conditions?. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 52-58, 1997.

[77] J. Atick, P. Griffin, N. Redlich. Statistical Approach to Shape form Shading: Reconstruction of Three-Dimensional Face Surfaces from Single Two-Dimensional Images. Neural Computation, vol. 8, Pag. 1321-1340, 1996.

[78] S. Ullman, R. Basri. Recognition by Linear Compinations of Models. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13, Pag. 992-1006, 1991.

[79] S. Akamatsu, T. Sasaki, H. Fukamaci, N. Masui, Y. Suenaga. An Accurate and Robust Face Identification Scheme. Proceedings International Conference on Pattern Recognition, Pag. 217-220, 1992.

[80] D.J. Beymer. Face Recognition Under Varying Pose. Technical Report 1461, MIT AI Laboratory, 1993.

[81] A.S. Georghiades, P.N. Belhumeur, D.J. Kriegman. Illumination-Based Image Synthesis: Creating Novel Images of Human Faces Under Differing Pose and Lighting. Proceedings, Workshop on Multiview Modeling and Analysis of Visual Scenes, Pag. 47-54, 1999.

[82] T. Maurer, C.v.d. Malsburg. Single-View Based Recognition of Faces Rotated in Depth. Proceedings, International Workshop on Automatic Face and Gesture Recognition, Pag. 176-181, 1996.

[83] D.J. Beymer, T. Poggio. Face Recognition from One Example View. Proceedings, International Conference on Computer Vision, Pag. 500-507, 1995.

[84] T. Vetter, T. Poggio. Linear Object Classes and Image Synthesis from a Single Example Image. IEEE Trans. on Pattern Analysis and Machine Intelligence, Pag. 733-742, 1997.

[85] E. Sali, S. Ullman. Recognizing Novel 3-D Objects Under New Illumination and Viewing Position Using a Small Number of Example Views or Even a Single View. Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Pag. 153-161, 1998.

[86] H.D. Ellis. Introduction to Aspects of Face Processing: Ten Questions in Need of Answers. Aspects of Face Processing, Pag. 3-13, 1986.

[87] Trolltech-Qt4, http://trolltech.com/products/qt

[88] Qdevelop - A Development Enviroment for Qt4, http://qdevelop.org

[89] GCC - The GNU C Compiler, http://gcc.gnu.org/gcc-4.5

[90] Qt Designer, http://trolltech.com/products/qt/features/designer

[91] Intel Open Source Computer Vision Library http://www.intel.com/technology/computing/opencv

[92] Trust Computer Products, http://www.trust.com/

[93] Xinput calibrator
http://wiki.ubuntu-it.org/TouchScreenKit
http://www.freedesktop.org/wiki/Software/xinput_calibrator

[94] Buttons http://vistabuttons.com/vista-buttons-setup.exe?affid=88105

[95] Debian GNU/Linux Operating System, http://www.debian.org