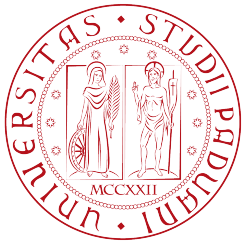

Università degli Studi di Padova
Facoltà di Ingegneria

DEI - Departement of Information Engineering
Corso di Laurea Magistrale in Ingegneria Informatica

Sistemi di Autenticazione Multibiometrica: Strumenti per il Deployment Automatico in Ambito Distribuito

Tesi di
Nicolò Paganin

Relatore:
Chiar.mo Prof. Carlo Ferrari .....

Candidato:
Nicolò Paganin Matr. 607267 .....

Sessione Laurea Dicembre 2010
Anno Accademico 2010/2011
Consultazione Consentita

A Giulia...

Alla mia Famiglia...

Sommario

Negli ultimi anni, l'autenticazione biometrica ha goduto di considerevoli miglioramenti, specialmente in materia di accuratezza ed affidabilità, con alcuni tratti biometrici che ottengono una buona performance complessiva.

Tuttavia, persino i migliori tratti biometrici disponibili incontrano ancora numerosi problemi, alcuni dei quali inerenti alla tecnologia stessa. In particolare, i sistemi di autenticazione biometrica soffrono generalmente di problemi di enrollment dovuti alla non-universalità dei tratti, di suscettibilità allo spoofing biometrico oppure presentano un alto livello di inaccuratezza nelle misurazioni attribuibile per lo più ad acquisizioni rumorose dovute, nella maggior parte dei casi, all'ambiente.

La multibiometria rappresenta un approccio che cerca di superare queste limitazioni, attraverso la realizzazione di un sistema che considera molteplici sorgenti di informazione biometrica.

L'obiettivo di questa tesi è la progettazione di un insieme di strumenti software che, prendendo in considerazione le molteplici scelte realizzative, supportano il deployment e l'esecuzione di un sistema di autenticazione multibiometrico.

Abstract

Multi biometric systems exploit different biometric traits, multiple samples and multiple algorithms to establish the identity of an individual. Over any single biometric system, they have the advantage of increasing the population coverage, offering user choice, making biometric authentication systems more reliable and resilient to spoofing, and most importantly, improving the authentication performance.

However, both the design and deployment of multi biometric systems raise many issues. These include system architecture, fusion methodology, selection of component biometric experts based on their accuracy and diversity, measurement of their quality, reliability and competence, as well as overall system usability, and economic viability.

The aim of this thesis is the realization of some instruments to support deployment and esecution phases of a multi-modal biometric authentication system, that considers a variety of possible opportunities, due to (obvious) the increasing complexity of the problem, when more distinct information sources are joined in order to obtain an unique final result.

Indice

Elenco delle figure

Capitolo 1

Biometria e sistemi biometrici

Contents

1.1 Introduzione

Sicurezza è una parola che, specialmente nell'ultimo decennio e per cause purtroppo note, viene adoperata sempre più spesso per indicare un bisogno crescente e trasversale in ambito sociale, economico, industriale e persino privato. Sovente tale necessità pone il suo essere a causa di una semplice sensazione condivisa su vasta scala, ma al migliorare delle conoscenze tecnologico/informatiche, crescono l'oggettiva incombenza ed urgenza di porre rimedio, o quanto meno ostacolare in maniera decisa, alle minacce che si presentano con frequenza sempre maggiore. Si pensi che i furti di identità negli esborsi per la previdenza sociale, nelle transazioni con carta di credito, nelle chiamate con telefoni cellulari provocano un costo totale annuo superiore ai sei milioni di dollari. Inoltre, dal momento in cui cresce il numero delle persone che si connettono elettronicamente, la capacità di raggiungere un livello di identificazione personale automatica molto accurato diviene un elemento sempre più critico. L'identificazione personale è il processo che consiste nell'associare un particolare individuo con una identità e rappresenta il centro attorno al quale ruoterà tutta la trattazione. Tradizionalmente, password (sicurezza basata sulla conoscenza) e ID-card (sicurezza basata su possesso di una chiave) sono state impiegate per restringere gli accessi ai sistemi di sicurezza. Tuttavia, è possibile fare breccia in questa fragile barriera allorchè una password venga divulgata ad utenti non autorizzati oppure nei casi in cui una carta venga rubata da un impostore. In aggiunta a questi semplici casi, può accadere che una parola d'accesso venga indovinata con relativa facilità da un utente non desiderato, dato che parole molto complicate possono risultare difficili da memorizzare per l'utente legittimo.

La nascita della biometria ha risolto molti dei problemi che minavano l'affidabilità dei metodi di verifica tradizionali. Questa disciplina fa riferimento all'identificazione (o alla verifica) automatica di un individuo (o dichiarato tale), adoperando determinati tratti di tipo fisiologico o comportamentale associati alla persona. Utilizzando la biometria è possibile stabilire un'identità basata su "chi sei" piuttosto che sul "cosa possiedi" (ID cards) o su "cosa ricordi" (password). (Sorprendentemente il 25% della popolazione scrive il proprio codice PIN sulla carta di credito, annullando così ogni garanzia di protezione!). Tuttavia anche questi sistemi (basati su di un singolo tratto) si sono dimostrati non esenti da lacune. Il passo successivo nella direzione di un ulteriore incremento nella sicurezza è rappresentato dall'impiego di più tratti biometrici all'interno dello stesso sistema, la cosiddetta multibiometria. Come si può intuire ci si aspetta un risultato migliore in termini di affidabilità rispetto ad un sistema unimodale; vi sono però per contro alcuni aspetti critici di cui tener conto in fase di progettazione, aumentando gioco-forza la complessità.

1.2 Obiettivo della tesi

L'obiettivo di questa tesi è la progettazione di un insieme di strumenti software a supporto del deployment e dell'esecuzioni di una architettura di autenticazione multibiometrica.

Il progetto prevede la realizzazione di interfacce user-friendly per la registrazione degli utenti nel sistema (*enrollment*), un tool grafico per il deployment e l'esecuzione dell'architettura del sistema e l'adattamento al sistema di un algoritmo per il riconoscimento di immagini facciali.

Particolare attenzione deve essere posta alla modalità di inserimento dei dati, ai vincoli progettuali sulla definizione dell'architettura di sistema ed alla combinazione delle informazioni acquisite ed elaborate dai veri sottoprocessi, al fine di non compromettere l'accuratezza complessiva del sistema.

1.3 Prerequisiti per la comprensione

Non sono richieste particolari conoscenze per un lettore che si appresti ad affrontare una lettura seria ed approfondita della tesi, nè per quanto concerne la biometria, nè per il campo informatico. Tuttavia una conoscenza delle informazioni basilari in ambito informatico agevoleranno notevolmente il compito, soprattutto nei seguenti ambiti:

- la programmazione, in quanto il software prodotto in questa tesi è stato sviluppato usando il linguaggio c/c++;
- alcune nozioni caratterizzanti i *sistemi informativi*, come il paradigma client-server, architetture e protocolli di comunicazione.

Una breve quanto organica visione d'insieme riguardante il campo della biometria e relativo modus operandi è necessaria per la comprensione di un sistema multimodale.

1.4 Introduzione alla biometria

Gli uomini hanno utilizzato le caratteristiche del proprio corpo come il volto, la voce, ecc. per centinaia di anni allo scopo di riconoscersi tra loro. Alphonse Bertillon, capo della divisione di identificazione criminale della polizia di Parigi, sviluppò e mise in pratica l'idea di impiegare un certo numero di misurazioni del corpo per identificare i criminali. Siamo a metà del diciannovesimo secolo. Nel momento in cui la sua idea iniziava a prendere piede, fu oscurata da una scoperta più significativa e soprattutto pratica: al volgere del diciannovesimo secolo venne scoperta l'unicità delle impronte digitali. Non trascorse molto tempo da questa scoperta, allorchè i diversi dipartimenti in materia di legge abbracciassero l'idea di creare il primo

booking delle impronte digitali dei criminali (una sorta di archivio) e memorizzarlo in quello che poteva essere definito concettualmente un database. In un secondo momento, tipicamente le impronte della mano destra, potevano essere rilevate direttamente nella scena del crimine e confrontate con quelle presenti nell'archivio per determinare l'identità nella scena del crimine e confrontate con quelle presenti nell'archivio per determinare l'identità del malfattore. Sebbene la nascita della biometria sia dovuta ad esigenze prettamente di legge nell'identificazione dei criminali, viene oggi impiegata in maniera sempre più frequente per stabilire il riconoscimento di una persona in uno svariato numero di applicazioni civili.

Quale misura biologica può essere qualificata a *biometrica*? [5] Qualsiasi caratteristica umana fisiologica e/o comportamentale ha diritto ad essere considerata come biometrica nel momento in cui soddisfa i seguenti requisiti:

- *Universalità*: ogni persona dovrebbe possedere tale caratteristica;
- *Distintività*: due persone qualsiasi dovrebbero essere sufficientemente differenti per quanto concerne la caratteristica in esame;
- *Permanenza*: la caratteristica dovrebbe essere sufficientemente invariante (relativamente al criterio di match) lungo un determinato periodo di tempo;
- *Collezionabilità*: la caratteristica può essere misurata quantitativamente.

Tuttavia, un sistema biometrico reale (un sistema che possa essere effettivamente impiegato per il riconoscimento tramite biometria), ci sono altre specifiche che debbono essere considerate, come ad esempio:

- *Performance*: fa riferimento all'accuracy ed alla velocità raggiungibili, le risorse necessarie per raggiungere i livelli desiderati, oltre ai fattori operazionali e ambientali che influenzano l'accuratezza e la velocità;
- *Accettabilità*: indica il grado con il quale la gente tende ad accettare l'utilizzo di un particolare identificatore biometrico nel proprio vivere quotidiano;
- *Permanenza*: la caratteristica dovrebbe essere sufficientemente invariante (relativamente al criterio di match) lungo un determinato periodo di tempo;
- *Circumvention*: riflette la facilità con la quale il sistema può essere aggirato utilizzando metodi fraudolenti;

Un sistema biometrico di utilizzo pratico dovrebbe possedere la specificata accuratezza [6] nel riconoscimento, velocità, requisiti di risorse, essere semplice da utilizzare per l'utente, accettata di buon grado dalla popolazione

ed essere sufficientemente robusto ai vari metodi di attacco fraudolento al sistema.

1.5 Sistema Biometrico

Un *sistema biometrico* [21] è essenzialmente un sistema di pattern recognition che opera acquisendo dati biometrici da un individuo, estraendo un feature set dalle informazioni ottenute e confrontando tale insieme di features con un template memorizzato tipicamente in un database. A seconda del contesto dell'applicazione, un sistema biometrico può operare, per quanto concerne l'identificazione, nelle modalità di *verifica* e *identificazione*:

- Nella modalità di "verifica", il sistema valida l'identità di una persona confrontando il dato biometrico acquisito con il relativo template (anche più d'uno) memorizzato nel database di sistema. All'interno di un sistema siffatto, un individuo che voglia essere identificato dichiara un'identità, usualmente attraverso un PIN (Personal Identification Number), uno username, una smart card, ..., ed il sistema realizza un confronto uno-a-uno al fine di determinare se l'identità acclarata sia vera o meno. La verifica dell'identità viene spesso impiegata per il *riconoscimento positivo*, dove lo scopo consiste nel prevenire l'utilizzo della stessa identità da parte di più persone.

- Attraverso la modalità di "identificazione", il sistema riconosce un individuo ricercando tra i templates di tutti gli utenti memorizzati nel database al fine di trovare riscontro. Quindi, il sistema effettua un confronto di tipo uno-a-molti per stabilire l'identità del soggetto (oppure fallimento qualora l'utente non risulti registrato nel database) senza che questi abbia dichiarato una qualche identità specifica. L'identificazione rappresenta un componente critico per quanto concerne le applicazioni di *negative recognition* nelle quali il sistema appura se la persona sia chi (implicitamente o esplicitamente) nega di essere. Il proposito del riconoscimento negativo è di impedire che un singolo individuo utilizzi una pluralità di identità. L'identificazione può altresì essere impiegata nel riconoscimento di tipo positivo. Mentre i metodi tradizionali di riconoscimento dell'individuo come password, PIN, parole chiave e token possono essere impiegate per il riconoscimento di tipo positivo, il riconoscimento negativo può essere stabilito unicamente attraverso la biometria.

In futuro, il termine *riconoscimento* verrà utilizzato in maniera generica allorchè non si riesca, o non si voglia distinguere tra la modalità di verifica o di identificazione. I diagrammi a blocchi di un sistema di verifica e di identificazione sono rappresentati in Figura 1.1. L'architettura di un sistema di autenticazione di identità automatico consiste di quattro componenti:

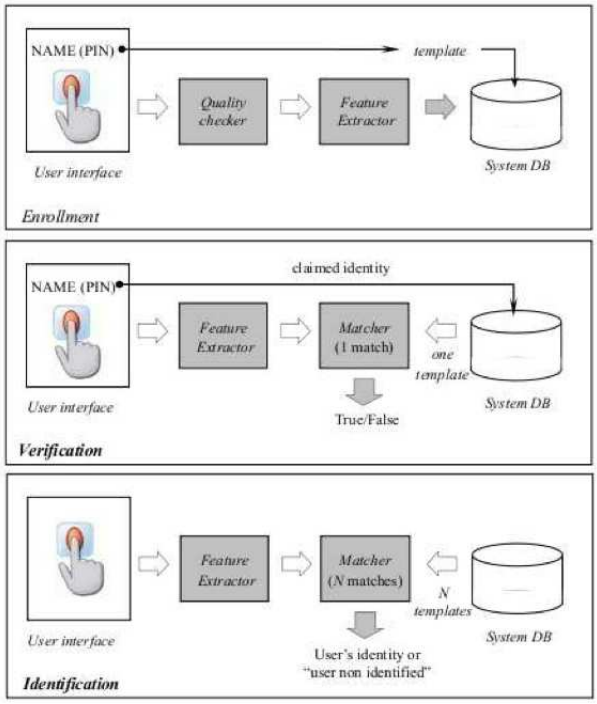

Figura 1.1: Procedure di enrollment, verifica e identificazione

1. interfaccia utente;
2. database di sistema;
3. modulo di enrollment;
4. modulo di autenticazione o verifica.

L'interfaccia utente fornisce quei meccanismi necessari all'utente per fornire le informazioni riguardanti la propria identità e per inserire il relativo tratto biometrico nel sistema. Il database consiste in una collezione di record contenenti alcuni campi utilizzati per scopi di identificazione:

- username della persona;
- template dei dati caratteristici del tratto relativo al soggetto;
- altre informazioni utili.

I rimanenti due elementi sono quelli che caratterizzano il modo di operare del sistema:

- *Modalità Enrollment*: in questa modalità viene acquisito il dato biometrico dell'utente utilizzando un lettore e successivamente memorizzato nel database. Il dato catturato può anche essere supervisionato

da parte di un umano a seconda del tipo di applicazione e del grado di accuratezza richiesto. Per facilitare le operazioni di matching, la rappresentazione digitale in input viene poi processata da un estrattore di features al fine di generare una rappresentazione compatta, ma dall'alto contenuto informativo, il *template*. Tale template viene etichettato con l'identità dell'utente (come visto in precedenza) per facilitarne l'autenticazione e, a seconda della situazione, può essere posto su un database centralizzato oppure su una smart card appartenente al soggetto registrato. Può risultare buona norma memorizzare più di un template, i quali possono essere aggiornati nel tempo per evitare l'insorgenza di problemi legati alla non permanenza.

- *Modalità Autenticazione*: per mezzo di questa modalità, il tratto biometrico viene nuovamente acquisito e il sistema lo utilizza per identificare quale utente sia, oppure allo scopo di accertare l'identità dichiarata dallo stesso. Si ricorda che, mentre l'identificazione consiste nel confronto del template acquisito con tutti quelli presenti nel database, la verifica comprende il solo confronto con il template relativo all'identità dichiarata. In questo modo, identificazione e verifica sono due problemi diversi aventi le proprie complessità.

Per quanto concerne i componenti, invece, in un sistema biometrico si distinguono quattro parti principali:

1. *sensore* che consente l'acquisizione del dato biometrico di un individuo. Un esempio è il sensore per le impronte digitali;

2. *Feature Extraction* nel quale il dato acquisito viene processato per estrarre i valori di feature. Per esempio, la posizione e l'orientamento dei punti delle minuzie in un'immagine di impronta digitale verranno prelevati ed estratti nel modulo di estrazione delle features del sistema;

3. *modulo di matching*, dove i valori presenti nelle features vengono confrontati con quelli presenti nel template al fine di generare un punteggio finale;

4. *modulo di decisione* nel quale viene stabilità l'identità dell'utente, oppure l'identità acclarata viene accettata o rifiutata, basandosi sul punteggio generato nel modulo di matching.

1.6 Metriche e Prestazioni

Valutare in maniera opportuna un sistema di identificazione biometrico è un campo di ricerca ancora aperto. Le prestazioni complessive di un

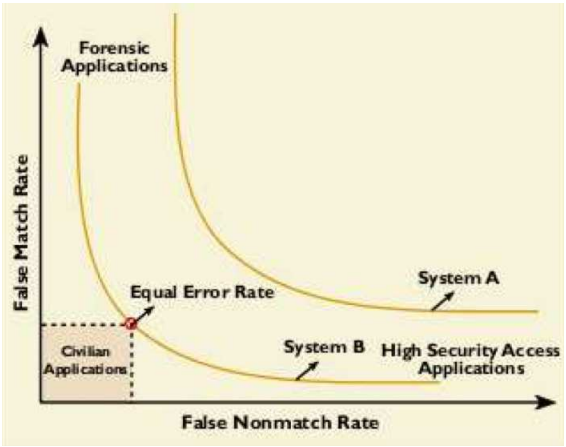

Figura 1.2: Receiver Operating Characteristics (ROC) di un sistema.

sistema biometrico sono stimate in termini di accuratezza, velocità e memorizzazione. Vengono altresì impiegati altri fattori che rivestono una qual certa importanza come il costo, la facilità di impiego e l'efficacia [22].

I sistemi biometrici non sono ovviamente perfetti, ed alle volte può accadere che venga accettato erroneamente un impostore come un individuo valido (quello che viene detto *false match* o, al contrario, respingere un soggetto valido (*false nonmatch*. La probabilità di commettere uno di questi due tipi di errore viene definita come false nonmatch rate (FNR) e false match rate (FMR); l'ampiezza di questi due errori dipende da quanto liberamente o conservativamente il sistema biometrico opera. La figura 1.2 evidenzia il trade-off tra il FMR e il FNR di un sistema generico in vari punti operativi; questo parametro ha un nome tecnico, viene denominato infatti "Receiver Operating Characteristics" (ROC) e rappresenta una misura comprensiva dell'accuratezza del sistema in un dato ambiente di test [23].

Applicazioni di accesso ad alta sicurezza, nei quali la preoccupazione di un break-in è notevole, operano ad un basso FMR. In applicazioni di medicina legale, dove il desiderio di catturare un criminale supera di gran lunga l'inconveniente dovuto al dover misurare un gran numero di individui falsamente accusati, opera i suoi matcher ad un alto livello di FMR. L'error rate del sistema nel punto operativo in cui la FMR eguaglia la FNR è chiamato Equal Error Rate (EER), il quale può spesso essere utilizzato come un descrittore conciso dell'accuratezza del sistema. Il grado di accuratezza di un sistema biometrico è considerato accettabile se i rischi (benefici) associati agli errori nel prendere le decisioni ad un dato punto nella curva ROC per un fissato ambiente di test sono ritenuti accettabili. In maniera del tutto similare, l'accuratezza di una identificazione basata sulla biometria è povera/inaccettabile qualora i rischi (benefici) associati agli errori relativi a qualsiasi punto sula ROC per uno specifico ambiente di test siano considerati

inaccettabili (insufficienti). La taglia e il numero dei template memorizzati per ogni individuo, l'opportunità di possedere meccanismi di compressione, determina lo spazio di memoria richiesto per ogni utente. Qualora si verifichi che la taglia del template non sia trascurabile e che il template stesso venga memorizzato in un database centralizzato, l'ampiezza di banda può diventare un collo di bottiglia per l'identificazione. Una tipica smart card è in grado di contenere solo pochi chilobyte di informazione (ad esempio, 8KB) quindi in un sistema che faccia utilizzo di tali dispositivi per distribuire il template, la taglia di quest'ultimo diviene un argomento di rilevante importanza.

La fase di decisione per un sistema biometrico risulta critica in molte applicazioni. Per una tipica applicazione di controllo accessi, il sistema necessita di una decisione pressochè in real-time. In applicativi ATM, ad esempio, è desiderabile ottenere una risposta entro il secondo, al più due. Per quanto concerne l'applicazione in medicina legale, tuttavia, il requisito temporale non rappresenta un parametro particolarmente restrittivo.

Il singolo fattore maggiormente rilevante nell'influenzare la realizzazione di un sistema è, come intuibile, il costo complessivo del sistema biometrico stesso, il quale include il sensore e le relative infrastrutture. Alcuni sensori, come i microfoni, sono già molto economici, mentre altri, come le telecamere CCD, stanno per raggiungere lo stadio di periferiche standard in ambiente di personal computing. Con i recenti progressi della tecnologia a stato solido, i sensori di impronte digitali diventeranno alla portata di tutti in brevissimo tempo. I requisiti di memoria dei template biometrici ed i requisiti di processing per il matching rappresentano due tra i più rilevanti aspetti inerenti i costi di infrastruttura.

Non deve essere sottovalutato nemmeno il *fattore umano* per quanto riguarda il successo di un identificatore basato sulla biometria. Quanto semplice e agevole è acquisire il tratto? Ad esempio, le misurazioni biometriche che non coinvolgono l'individuo, come il volto, la voce, l'iride, possono essere percepiti dall'utente come maggiormente user-friendly. In aggiunta a quanto detto, quelle tecnologie biometriche che richiedono un basso grado di coinvolgimento/cooperazione da parte dell'utente (come il volto o la termografia) possono essere percepite come le maggiormente convenienti. Un argomento strettamente collegato a questo è rappresentato dall'accettazione pubblica. Può esserci la percezione prevalente che la biometria sia un pericolo per la privacy di un individuo. A questo proposito, la popolazione avrebbe bisogno di essere informata che la biometria può essere uno dei più efficaci, e nel lungo periodo, più proficui metodi per proteggere la privacy. Ad esempio, un sistema di informazione sui pazienti basato sulla biometria può efficacemente assicurare che le cartelle mediche siano accessibili solo dal personale medico e da utenti autorizzati. Un buon approccio per guidare e aumentare gradualmente l'accettazione di soluzioni biometriche potrebbe essere la loro introduzione su base volontaria, unitamente ad incentivi impliciti od espliciti allo scopo di optare per queste soluzioni.

1.7 Tratti biometrici e loro utilizzi

La biometria misura caratteristiche fisiche o comportamentali uniche negli individui al fine di riconoscere o autenticare la loro identità. I tratti fisici più comuni comprendono le impronte digitali, la geometria del palmo, la retina, l'iride o le caratteristiche del volto.

I tratti comportamentali includono la firma, la voce (che ha anche una componente fisica), il processo di pressione tasti (keystroke) e l'andatura. All'interno di quest'ultima classe la firma e la voce rappresentano senza dubbio quelli maggiormente impiegati [24].

1.7.1 Impronte digitali

Un'impronta digitale fa riferimento ai pattern trovati sui polpastrelli. Esistono numerosi approcci per quanto concerne l'autenticazione mediante impronta digitale. Alcune emulano il tradizionale metodo usato in polizia del confronto tra le minuzie; altri adoperano dei dispositivi di pattern matching; ulteriori e più complessi prevedono addirittura l'impiego di ultrasuoni. Alcuni sistemi riescono anche a capire quando si stanno utilizzando le dita di un essere vivo; altri invece no.

Questo tipo di tratto è quello che può vantare il maggior numero di dispositivi disponibili in commercio. Dato che il prezzo di questi sensori ed i costi di processing sono in continua diminuzione, l'impiego delle impronte digitali per l'identificazione degli utenti sta prendendo sempre più piede.

La verifica tramite impronte può rappresentare una scelta valida per i sistemi in-house, nei quali è possibile fornire agli utilizzatori adeguate spiegazioni ed effettuare un buon training, dove il sistema opera in un ambiente controllato. Non sorprende infatti che le applicazioni di accesso a workstation siano basate quasi esclusivamente su impronte digitali, dovuto in gran parte come detto al costo relativamente basso, la piccola taglia e la facilità di integrazione con dispositivi di autenticazione di impronte.

1.7.2 Retina

Un sistema biometrico basato su retina coinvolge l'analisi dello strato di vasi sanguigni situato posteriormente l'occhio. Una tecnologia affermata, questa tecnica impiega sorgenti luminose a bassa intensità per mezzo di un accoppiatore ottico allo scopo di effettuare una scansione dei pattern unici della retina. Tali scansioni possono essere abbastanza accurate, ma richiedono che l'utente guardi all'interno di un ricettacolo focalizzandosi su di un dato punto. Questo non risulta essere particolarmente conveniente se ad esempio si indossano gli occhiali o se si è preoccupati dall'avere uno stretto contatto con il dispositivo di lettura. Per queste ragioni, la scansione

della retina non viene accettata di buon grado da tutti gli utenti, anche se la tecnologia stessa funziona in effetti bene.

1.7.3 Iride

Questo particolare tratto richiede l'analisi delle caratteristiche riscontrate nell'anello di tessuto colorato che circonda la pupilla. La scansione dell'iride, senza alcun dubbio il meno intrusivo tra tutti i tratti che riguardano l'occhio, impiega una telecamera di uso abbastanza convenzionale e non richiede contatto tra il dispositivo di lettura e l'utente. in aggiunta, possiede il potenziale per ottenere performance di template-matching sopra la media. La biometria dell'iride lavora anche con occhiali ed è una delle poche che possono ben figurare allorchè si attua la modalità di identificazione. La facilità di impiego e l'integrazione nel sistema non sono certamente e tradizionalmente punti di forza dei dispositivi di lettura della retina, ma ci si aspetta miglioramenti significativi entro breve tempo.

1.7.4 Volto

L'analisi del volto analizza le caratteristiche del viso. Richiede una telecamera digitale per sviluppare un'immagine del volto dell'utente per l'identificazione. Questa tecnica ha attirato notevole interesse, sebbene molte persone non ne comprendano a pieno le capacità. Alcuni commercianti hanno annunciato possibilità stravaganti - molto difficili, se non impossibili da realizzare in pratica - per dispositivi di riconoscimento del volto. Siccome lo scanning del volto richiede una periferica extra non inclusa con i PC tradizionali che si trovano in commercio, si è ancora in un mercato di nicchia per quanto riguarda l'autenticazione di rete. Tuttavia l'industria dei casinò ha investito in questa tecnologia allo scopo di creare un database di volti di giocatori-truffatori per il loro riconoscimento immediato da parte del personale di sicurezza.

1.7.5 Firma

La verifica della firma analizza il modo mediante il quale l'utente scrive il proprio nome. Caratteristiche inerenti la scrittura quali la velocità e la pressione sono importanti, come la forma statica della fine della firma. Questo tipo di tratto realizza una sinergia con i processi esistenti che altri sistemi non riescono ad ottenere. La gente è solita intendere la firma come un mezzo di verifica dell'identità per quanto concerne le transazioni, quindi estendere il passo concettuale alla biometria non richiede un grande sforzo. I dispositivi atti a questo tipo di rilevazioni sono ragionevolmente accurati nelle operazioni e ovviamente di prestano nelle applicazioni dove la firma è un identificatore accettato. In maniera piuttosto sorprendente, sono emerse relativamente poche applicazioni significative sulla firma, se confrontate con

altre metodologie biometriche; ma se l'applicativo da realizzare lo richiede, rappresenta una tecnologia da sfruttare senza remore.

1.7.6 Voce

L'autenticazione basata su voce non si basa sul riconoscimento della voce in senso stretto, ma su di un meccanismo di autenticazione voice-to-print, nel quale una complessa tecnologia trasforma la voce in testo. Questo tipo di tratto biometrico è quello con il potenziale di crescita più elevato perchè non richiede nuovo hardware (la maggior parte dei computer contengono già un microfono). Tuttavia, la scarsa qualità e la rumorosità degli ambienti possono compromettere l'autenticazione. In aggiunta, la procedura di enrollment risulta spesso molto più complicata che con altri tratti biometrici, giungendo alla percezione che la voce non sia affatto user-friendly. Quindi, il software per l'autenticazione della voce necessita di miglioramenti. Si prospetta un futuro in coppia con le impronte digitali per quanto riguarda la voce; dal momento in cui molta gente vede la rilevazione delle impronte digitali come la forma più alta di autenticazione, la biometria della voce rimpiazzerà o migliorerà l'utilizzo di PIN, password, o nomi di account.

1.8 Scegliere una tecnologia biometrica

La tecnologia biometrica rappresenta un'area che nessun segmento dell'industria IT può permettersi di ignorare. La biometria fornisce benefici in termini di sicurezza su vasta gamma, dai commercianti appunto in IT agli utenti finali, dagli sviluppatori di sistemi di sicurezza fino a giungere ai loro fruitori. Tutti questi settori dell'industria devono valutare i costi e i benefici derivanti dall'implementare questo tipo di misure di sicurezza.

Tecnologie differenti possono essere appropriate per diverse tipologie di applicazioni, a seconda della percezione che si ha dei profili dell'utente, della necessità di interfacciarsi con altri sistemi o database, condizioni ambientali, ed una serie di altri parametri dipendenti dalla specifica applicazione (cfr. figura 1.3).

1.8.1 Facilità di utilizzo

Alcuni dispositivi biometrici non sono user-friendly. Ad esempio, utenti senza un training adeguato possono incontrare delle difficoltà nell'allineare la propria testa ad un dispositivo per l'enrollment ed il successivo matching dei template del volto.

Table 1. Comparison of biometrics

Characteristic	Fingerprints	Hand geometry	Retina	Iris	Face	Signature	Voice
Ease of Use	High	High	Low	Medium	Medium	High	High
Error incidence	Dryness, dirt, age	Hand injury, age	Glasses	Poor lighting	Lighting, age, glasses, hair	Changing signatures	Noise, colds, weather
Accuracy	High	High	Very high	Very high	High	High	High
Cost	*	*	*	*	*	*	*
User acceptance	Medium	Medium	Medium	Medium	Medium	Very high	High
Required security level	High	Medium	High	Very high	Medium	Medium	Medium
Long-term stability	High	Medium	High	High	Medium	Medium	Medium

* The large number of factors involved makes a simple cost comparison impractical.

Figura 1.3: Confronto tra parametri biometrici

1.8.2 Incidenza d'errore

Due sono le cause primarie che affliggono il dato biometrico: il tempo e le condizioni ambientali. I tratti biometrici possono variare con il trascorrere dell'età. Le condizioni ambientali possono altresì alterare il tratto in maniera diretta (ad esempio, se un dito è tagliato o presenta cicatrici), oppure interferire con la raccolta del dato stesso (per esempio il rumore di sottofondo quando si sta acquisendo una voce).

1.8.3 Accuratezza

I commercianti adottano spesso due differenti metodi per giudicare l'accuratezza di un sistema biometrico: il *False-Acceptance Rate* ed il *False-Rejection Rate*. Entrambi i metodi si caratterizzano sull'abilità del sistema nel permettere accessi limitati ad utenti non autorizzati. Tuttavia, queste misure possono variare in maniera significativa, a seconda di come si aggiusta la sensibilità del meccanismo che confronta i dati biometrici. Ad esempio, è possibile richiedere un matching restrittivo tra le misurazioni della geometria della mano ed il template dell'utente (incremento della sensibilità). Questo probabilmente comporterà una diminuzione del False-Acceptance Rate, ma allo stesso tempo può aumentare il False-Rejection Rate. Quindi è doveroso porre particolare attenzione nel comprendere come i commercianti giungono ai valori espressi di FAR e FRR.

Essendo FAR e FRR interdipendenti, è più significativo rappresentarli assieme uno contro l'altro. In linea di principio i tratti biometrici di tipo fisico risultano più accurati di quelli comportamentali.

1.8.4 Costo

La componente costo comprende:

- l'hardware per la cattura del tratto biometrico;

- carico di lavoro di back-end processing per il mantenimento del database;
- ricerca e testing sul sistema biometrico;
- installazione, comprendente il compenso del team operante nell'implementazione;
- montaggio, installazione, connessione e costi di integrazione nel sistema utente;
- formazione degli utenti, spesso condotte per mezzo di campagne di marketing;
- eccezioni al processing, tutti quegli utenti che non possono per vari motivi, evidentemente biologici, essere inseriti nel sistema (mancanza di impronte, ecc.);
- mantenimento del sistema.

1.8.5 Accettazione da parte dell'utente

Parlando in maniera meramente generica, quanto meno intrusivo è il tratto biometrico da acquisire, tanto più rapidamente verrà accettato. Tuttavia, alcuni gruppi di utenti, come ad esempio quelli religiosi o ferventi sostenitori della libertà civile, rifiutano le tecnologie a carattere biometrico per motivi di etica o privacy.

1.8.6 Livello di sicurezza richiesto

Le organizzazioni dovrebbero determinare il livello di sicurezza richiesto per una specifica applicazione: basso, moderato, oppure elevato. Tale decisione avrà una notevole incidenza su quale tratto biometrico sia più appropriato. Generalmente, le caratteristiche di tipo comportamentale sono sufficienti per un sistema con livello di sicurezza basso-moderato, tratti fisici sono invece preferiti per applicazioni ad elevato grado di sicurezza.

1.8.7 Stabilità nel lungo periodo

Le imprese dovrebbero altresì considerare la stabilità di un tratto biometrico, concetto comprendente la maturità della tecnologia, il grado di standardizzazione, il livello del venditore ed il supporto governativo, la condivisione del mercato, ed ulteriori fattori di supporto. Tecnologie mature e standardizzate godono ovviamente di una maggiore stabilità.

Capitolo 2

Multibiometria: l'unione fa la forza

Contents

2.1 Limitazioni dei sistemi biometrici unimodali

L'installazione con successo di sistemi biometrici in varie applicazioni civili non implica affatto che la biometria sia un problema completamente risolto. Risulta del tutto evidente che esistono parecchie opportunità di miglioramento. La ricerca non persegue solamente la direzione che conduce

ad una riduzione degli error rate, ma anche quella che porta al miglioramento dell'usabilità del sistema stesso.

I sistemi biometrici che operano utilizzando una singola caratteristica biometrica presentano le seguenti limitazioni:

1. *Rumore del dato acquisito*: il dato acquisito potrebbe essere afferro da rumore oppure distorto. Un'impronta con una cicatrice, oppure una voce alterata dal freddo sono tipici esempi di dato rumoroso. Un dato rumoroso potrebbe anche essere il risultato di un mantenimento difettoso o improprio dei sensori (come l'accumulo di polvere in un sensore per impronte digitali) o di avverse condizioni ambientali (scarsa illuminazione del volto di un utente in un sistema di riconoscimento del volto). Dati biometrici afflitti da rumore possono essere comparati in maniera errata con i template nel database, con il risultato di ottenere un utente respinto incorrettamente.

2. *Variazioni intra-classe*: il dato biometrico acquisito da un individuo durante il processo di autenticazione può essere anche di molto difforme rispetto al dato che è stato impiegato per generare il template durante la procedura di enrollment, condizionando pertanto il processo di matching. Tale variazione è causata tipicamente da un utente che sta interagendo in maniera non corretta con il sensore, oppure quando le caratteristiche del sensore stesso vengono modificate (ad esempio cambiando il sensore, problema di interoperabilità del sensore) durante la fase di identificazione.

3. *Distintività*: sebbene ci si aspetti che un tratto biometrico possa cambiare significativamente tra la popolazione, si possono presentare un vasto numero di similitudini intra-classe all'interno dei feature set utilizzati per la rappresentazione di queste caratteristiche. Tali limitazioni riducono la discriminabilità fornita dal tratto biometrico. è stato dimostrato che il contenuto informativo (inteso come numero di pattern distinguibili) in due delle più comuni rappresentazioni impiegate per il volto e l'iride siano dell'ordine di 10^3 e 10^5 rispettivamente. In tal modo, ogni tratto biometrico ha un qualche teorico limite superiore in termini di capacità discriminatoria.

4. *Non universalità*: ogni utente possiede il tratto biometrico da acquisire. Questa affermazione è al contempo falsa e pericolosa. E' in realtà possibile che per un sottoinsieme di utenti non possedere una particolare caratteristica biometrica. Un sistema biometrico basato su impronta digitale, ad esempio, può non essere in grado di estrarre feature dalle impronte di un determinato individuo in conseguenza alla scarsa qualità delle creste. In questo modo si ha un *Failure To Enroll rate* (FTE) associato all'impiego di un singolo tratto. E' stato stimato

empiricamente che all'incirca il 4% della popolazione può presentare scarsa qualità nelle creste delle impronte digitali, un dato difficile da immaginare pensando ai sensori di impronta digitale disponibili oggi nel mercato, con l'ovvio risultato di un errore FTE.

5. *Spoof attacks*: un impostore può provare ad appropriarsi di un tratto biometrico di un utente legittimamente registrato per ingannare il sistema. Questo tipo di attacco assume particolare importanza quando vengono utilizzati tratti biometrici come la firma e la voce. Tuttavia caratteristiche di tipo fisico non sono immuni da questo fenomeno. Ad esempio, è stato dimostrato come sia possibile (sebbene difficile, ingombrante e richieda l'aiuto da parte dell'utente legittimo) costruire dita/impronte artificiali in un ragionevole arco temporale sufficiente ad ingannare il sistema di verifica.

2.2 Vantaggi dei sistemi multibiometrici

Alcune delle limitazioni derivanti dall'impiego di sistemi biometrici unimodali (sistemi biometrici che si basano sulla testimonianza fornita da un solo tratto biometrico) possono essere superate impiegando una pluralità di tratti biometrici. Tali sistemi, conosciuti come multibiometrici, sono ritenuti maggiormente affidabili per la presenza contemporanea di più caratteristiche, ritenute fortemente indipendenti tra loro. Questi sistemi sono anche in grado di soddisfare gli stringenti requisiti di performance imposti dalle varie applicazioni nei quali di solito sono inseriti. L'incremento dell'accuratezza del matching non è auspicabilmente l'unico vantaggio introdotto dai sistemi multibiometrici rispetto ai tradizionali sistemi unimodali; alcuni dei più significativi miglioramenti che si verificano sono riportati in questo paragrafo [25].

1. I sistemi multibiometrici affrontano il problema della non-universalità (copertura della popolazione limitata) incontrato nei sistemi unimodali. Se un dito non particolarmente secco di un soggetto gli impedisce di eseguire in maniera corretta la fase di enrollment all'interno di un sistema di riconoscimento di impronta digitale, la disponibilità di un tratto biometrico (o più d'uno), può essere rilevante aiuto allo scopo di includere l'individuo stesso nel sistema biometrico. Viene raggiunto un certo grado di flessibilità allorchè un utente si registra nel sistema impiegando una molteplicità di tratti (come volto, iride, voce, impronte digitali e mano), mentre solamente un sottoinsieme di queste caratteristiche (voce e impronte) viene richiesto durante l'autenticazione, che può essere basata sulla natura dell'applicazione che si sta considerando e la comodità dell'utente.

2. I sistemi multibiometrici possono facilitare il filtraggio o l'indicizzazione dei database biometrici su larga scala. Ad esempio, in un sistema bimodale composto da volto ed impronta digitale, il feature set del volto può essere adoperato per calcolare un valore di indice allo scopo di estrarre una lista di candidati di potenziali identità da un vasto database di soggetti. La modalità impronte digitali può successivamente determinare l'identità finale da questa ristretta cerchia di candidati.

3. Diventa sempre più complicato (anche se non impossibile) per un impostore tentare di eseguire operazioni di spoofing su una pluralità di tratti biometrici appartenenti ad un utente legittimamente registrato. Se ogni sottosistema indica la probabilità che un particolare tratto sia uno "spoof", viene successivamente impiegato uno schema di fusione appropriato allo scopo di determinare se l'utente, di fatto, sia un impostore. In aggiunta a quanto riportato, chiedendo ad un soggetto di presentare un ristretto numero random di tratti al momento dell'acquisizione, un sistema multibiometrico facilita una tipologia di meccanismo challenge-response, assicurando quindi che il sistema stia interagendo con un utente effettivo. Da notare che un meccanismo di questo tipo può essere inizializzato anche nei sistemi di tipo unibiometrico (ad esempio, il sistema può asserire "Per favore dire 1-2-5-7", "chiudi gli occhi due volte e muovi gli occhi verso destra", "cambia la tua espressione sorridendo", ecc.).

4. Un ulteriore problema che viene superato in modo efficace per mezzo dei sistemi multibiometrici è quello relativo alla rumorosità del dato in acquisizione. Nel momento in cui un segnale biometrico catturato da un singolo tratto risulta alterato da rumore, la disponibilità di ulteriori caratteristiche (auspicabilmente meno rumorose) possono aiutare nel determinare in maniera affidabile l'identità. Alcuni sistemi prendono in considerazione la qualità dei segnali biometrici dell'individuo durante il processo di fusione. Questo diventa importante specialmente allorchè il riconoscimento debba avvenire in condizioni avverse, dove alcuni tratti biometrici non possano essere estratti in maniera consona. Per fare un esempio, in presenza di rumore ambientale di tipo acustico, che impedisce la misura accurata della caratteristica della voce di un individuo, può essere adoperata dal sistema multibiometrico la caratteristica del volto per effettuare la procedura di autenticazione. Stimare la qualità di un dato acquisito rappresenta un ulteriore problema da affrontare ed impone determinate scelte, ma quando viene realizzato in maniera opportuna, rappresenta un valore aggiunto in termini di benefici per un sistema multibiometrico.

5. Questa tipologia di sistemi aiuta anche nel continuo monitoraggio od il tracking di un individuo in situazioni nelle quali un singolo tratto non è sufficiente. Considerando un sistema biometrico che impiega una telecamera 2D per catturare il volto e le informazioni sull'andatura di una persona che cammina lungo un corridoio particolarmente affondato, a seconda della posizione e della distanza del soggetto rispetto al punto di ripresa, le caratteristiche possono essere entrambe contemporaneamente presenti oppure no. Quindi, almeno una (ma anche entrambe) di queste caratteristiche possono essere sfruttate in relazione al posizionamento del soggetto rispetto al sistema di acquisizione, permettendo in tal modo il continuo monitoraggio dello stesso.

6. Un sistema multibiometrico può anche essere visto come un sistema fault tolerant, il quale continua ad operare anche quando certe sorgenti biometriche diventano non più affidabili, in seguito a malfunzionamenti nel sensore o nel software, oppure a deliberate manomissioni da parte di utenti. La nozione di fault tolerance diviene utile specialmente per quanto concerne i sistemi di autenticazione su vasta scala che coinvolgono un grande numero di soggetti (come un'applicazione di controllo imbarchi).

2.3 Tassonomia dei sistemi multibiometrici

Fino a questo punto della trattazione si è parlato di multibiometria e di sistema multibiometrico in termini del tutto generici, indicando come caratteristica fondamentale e distintiva il basarsi su evidenze fornite da diversi tratti biologici appartenenti ad un determinato individuo. Ora che il discorso è sufficientemente introdotto, è possibile e doveroso approfondire il concetto di multibiometria. Basandosi sulla natura delle sorgenti di informazione biometrica, un sistema può essere classificato in una delle seguenti categorie: multi-sensore, multi-algoritmo, multi-istanza, multi-campione, multi-modale e ibrido [26].

1. *Sistemi multi-sensore*: come ovviamente suggerito dal nome, tali sistemi impiegano una molteplicità di sensori per catturare un singolo tratto biometrico di un individuo. Ad esempio, il sistema di riconoscimento del volto può contare su alcune telecamere 2D per acquisire l'immagine del volto di un soggetto; un sensore all'infrarosso può essere adoperato assieme ad un sensore con una diversa sensibilità alla luce per acquisire l'informazione sulla superficie del volto di una persona; una telecamera multispettro ha invece la sua utilità nell'ottenimento di immagini relative all'iride o alle dita; nel caso di impronte digitali, possono trovare impiego un sensore capacitivo ed uno ottico. L'impiego di più sensori, in alcuni casi, può dare luogo all'acquisizione

di informazioni tra loro complementari, con l'effetto di migliorare le capacità riconoscitive complessive del sistema. A titolo esemplificativo, basandosi sulla natura dell'illuminazione dovuta alla luce naturale, le immagini ad infrarosso e visible-light del volto di una persona possono presentare diversi gradi di informazione portando ad ottenere una migliore accuratezza del sistema. In maniera del tutto similare le performance di un sistema di rilevazione attraverso una telecamera 2D può essere incrementato con il coinvolgimento degli strumenti 3D.

2. *Sistemi multi-algoritmo*: in alcune configurazioni, invocare più di un algoritmo per l'estrazione di feature e/o per il matching, può condurre ad una più elevata efficacia nel matching. I sistemi multi-algoritmo consolidano l'output fornito da diversi algoritmi per l'estrazione delle feature, o quello di vari matcher che operano sullo stesso feature set. Un suddetto sistema non necessita dell'impiego di nuovi sensori, quindi risulta essere più economico se paragonato ad altre tipologie di sistema multibiometrico. D'altro canto, però, l'introduzione di nuovi moduli per quanto concerne il processo di feature extraction e matching possono gravare sulla complessità computazionale del sistema.

3. *Sistemi multi-istanza*: tali sistemi impiegano più istanze differenti dello stesso tratto del corpo e vengono anche riconosciuti in letteratura con il nome di sistemi multi-unità. Ad esempio, gli indici della mano destra e sinistra, oppure gli iridi sinistro e destro di un individuo, possono essere adoperati per verificare l'identità di una persona. Il programma di sicurezza di bordo US-VISIT impiega attualmente l'indice della mano destra e sinistra dei viaggiatori per validare i documenti di viaggio alla porta d'imbarco. Il IAFIS[1] della FBI combina dati di tutte e dieci le dita delle mani per determinare il matching di identità nel database. Questi sistemi possono essere convenienti dal punto di vista economico qualora venga impiegato un solo sensore per acquisire i dati multi-unità in modo sequenziale (come US-VISIT[2]). Tuttavia, in alcune circostanze, può essere desiderabile ottenere questi dati in maniera simultanea (IAFIS), esigendo quindi la strutturazione di un dispositivo di acquisizione efficace (e probabilmente più costoso).

4. *Sistemi multi-campione*: un singolo sensore può essere adoperato per acquisire molteplici campioni dello stesso tratto biometrico allo scopo di considerare le variazioni che possono verificarsi nello stesso, oppure per ottenerne una descrizione più completa. Un sistema di riconoscimento del volto, ad esempio, può essere in grado di catturare (e memorizzare) il profilo frontale relativo al volto di un individuo assieme ai

[1] Integrated Automated Fingerprint Identification System, `http://www.fbi.gov/hq/cjisd`

[2] per info, `http://www.dhs.gov/files/programs/usv.shtm`

profili destro e sinistro allo scopo di considerare le possibili variazioni nella posa. In maniera del tutto simile, un sistema di riconoscimento basato sulle impronte digitali che fa uso di un piccolo sensore è in grado di acquisire più porzioni di impronta dello stesso individuo al fine di ottenere immagini di varie regioni. Sarà poi possibile comporre l'immagine completa utilizzando uno schema a mosaico. Uno degli elementi chiave in un sistema multi-campione consiste nel determinare il *numero* di campioni che devono essere acquisiti da un individuo. È di estrema importanza che i campioni forniti rappresentino sia la *variabilità* che la *tipicità* del dato biometrico relativo al soggetto. A questo scopo, la relazione desiderata tra i campioni deve essere stabilita in principio, al fine di poter ottimizzare i benefici derivanti dalla strategia di integrazione. Per esempio, un sistema di riconoscimento del volto che utilizzi sia le immagini frontali che laterali del profilo di un individuo può decidere che le immagini del profilo siano tutto ottenute mediante una cattura di tre quarti del volto. Alternativamente, dato un set di campioni biometrici, il sistema dovrebbe essere automaticamente in grado di determinare il sottoinsieme "ottimale" in grado di rappresentare al meglio la variabilità di un individuo.

5. *Sistemi multi-modali*: stabiliscono l'identità basandosi sulla prova fornita da molteplici tratti biometrici. A titolo di esempio, alcuni dei più recenti sistemi biometrici multimodali impiegano le caratteristiche della voce e del volto per stabilire l'identità di un soggetto. Tratti fisicamente non correlati (come impronte e iride) sono indicati come candidati a fornire maggiori miglioramenti nella performance rispetto a caratteristiche correlate tra loro (vedasi voce e movimento labbra). Il costo di sviluppo per questo tipo di sistemi è sostanzialmente dovuto più all'esigenza di avere nuovi sensori e, di conseguenza, lo sviluppo di appropriate interfacce utente. L'accuratezza nell'identificazione può essere significativamente migliorata utilizzando un numero crescente di tratti sebbene il fenomeno del *curse-of-dimensionality* imponga un limite a tale numero. Il numero di tratti impiegato per una specifica applicazione verrà ristretto nella pratica anche per considerazioni basate sul costo di sviluppo, il tempo di registrazione, il tempo di throughput, l'error rate atteso, ecc.

6. *Sistemi ibridi*: il termine "ibrido" viene adoperato per descrivere sistemi che integrano un sottoinsieme dei cinque scenari descritti in precedenza.

2.4 Livelli di fusione e architetture

Nel paragrafo precedente si è cominciato a vedere come, nel design di un sistema multibiometrico, debbano essere considerati un numero considerevole e vario di fattori. Questi includono, ad esempio, la scelta del numero di tratti; il livello nel sistema nel quale le informazioni fornite della varie caratteristiche dovrebbero essere integrate, la metodologia adottata per attuare tale combinazione; anche la valutazione del compromesso tra costo e performance del matching rappresenta un parametro di valutazione importante. La scelta del numero di tratti è quasi sempre guidata dalla natura dell'applicazione, dall'overhead introdotto da tratti multipli (risorse di calcolo e costo, ad esempio) e la correlazione tra i tratti considerati. Considerando un telefono cellulare dotato di telecamera, può sembrare semplice combinare le caratteristiche di un volto e impronte digitali di un utente. Una volta ottenuta l'informazione dei vari tratti componenti il sistema attraverso i sensori dedicati, sorge la necessità di unire in qualche modo quanto acquisito in una determinata forma allo scopo di ricavare una risposta quanto più chiara possibile al problema dell'identificazione. Tale processo di unione prende il nome di *fusione*; a seconda di dove si verifica, il sistema assume una connotazione architetturale specifica e differente dalle altre. Le descrizioni dei livelli di fusione e relative architetture saranno l'oggetto di questo paragrafo [27].

Come suggerito dalla letteratura (riferimenti in [28] e [29]), i sistemi multibiometrici vengono classificati in tre architetture di sistema in accordo con le strategie utilizzate per fondere le informazioni pervenute:

- Fusione a livello di *estrazione delle feature*;
- Fusione a livello di *matching score*;
- Fusione a livello di *decisione*.

Riepilogando, i sistemi vengono categorizzati a seconda di quanto presto, all'interno del processo di autenticazione, l'informazione viene combinata.

In realtà vi sono due ulteriori livelli, peraltro scarsamente utilizzati, riscontrabili in letteratura e che verranno brevemente accennati per completezza informativa al termine della descrizione delle architetture principali: la *fusione a livello di sensore* e quella a *livello di rank*.

è già stato illustrato nel capitolo relativo alla biometria unimodale come l'autenticazione sia un processo a catena, che segue le fasi riassunte nella figura 2.1.

La fusione a livello di estrazione delle feature rappresenta l'integrazione immediata dei dati all'inizio della catena del processo, mentre la fusione a livello di decisione rappresenta l'ultimo punto di integrazione al termine del percorso.

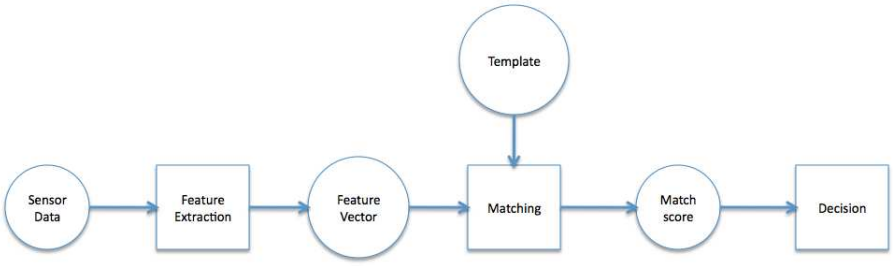

Figura 2.1: Flusso del processo di autenticazione

2.4.1 Fusione a livello di estrazione delle feature

In questa architettura [30], l'informazione estratta dai diversi sensori viene codificata in un feature vector congiunto, che viene poi confrontato con un template registrato nella fase di enrollment (che è ovviamente anch'esso un feature vector congiunto memorizzato nel database) e al quale viene assegnato un punteggio di similarità (matching score) come in un comune sistema biometrico unimodale (vedasi figura 2.2).

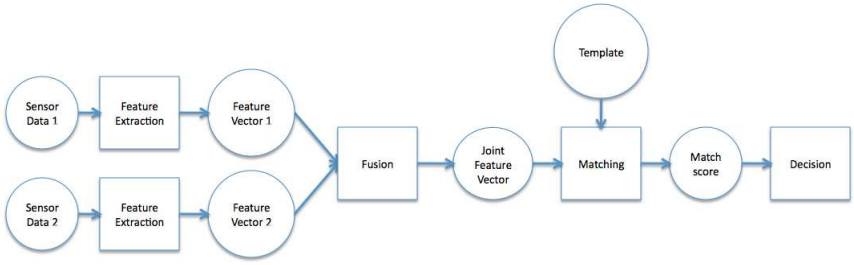

Figura 2.2: Fusione a livello di estrazione delle feature

Da un'analisi approfondita della letteratura sull'argomento non emerge una qualche significativa, quanto recente, ricerca in questa strategia di integrazione. Questo suggerisce che la fusione a livello di estrazione delle feature sia meno preferibile rispetto alle altre due. Si identificano due principali problemi relativamente a questo approccio:

- I feature vector da unire possono essere incompatibili (dovuto banalmente ad incongruenze di tipo numerico), oppure alcuni di essi possono non essere disponibili (ad esempio, nei casi in cui l'utente non possegga tutti gli identificatori biometrici). Mentre il primo problema potrebbe essere risolto per mezzo di un'accorta progettazione del sistema, portando al raggiungimento di un sistema fortemente accoppiato, il secondo probabilmente causerà quei problemi nella fase di enrollment già elencati nella relativa sezione.

- La generazione del punteggio rappresenta un problema: anche in un sistema con un singolo tratto biometrico, non è cosa da poco trovare un buon classificatore, nel senso di creare un punteggio rappresentativo basato sul confronto tra il feature vector ed il template memorizzato. Quindi, nel caso di feature vector congiunti di grandi dimensioni in un sistema multibiometrico, diviene ancora più complicato. La relazione tra le diverse componenti del vettore potrebbe inoltre non essere lineare [31].

2.4.2 Fusione a livello di matching score

In un sistema multibiometrico costruito con questa architettura, i feature vector vengono creati indipendentemente da ogni sensore e successivamente confrontati con i template generati nella fase di enrollment, i quali sono memorizzati separatamente per ciascun tratto biometrico. Basandosi sulla prossimità tra il feature vector ed il template, ogni sottosistema calcola a questo punto il proprio punteggio di similarità. I punteggi individuali così ottenuti vengono combinati in un punteggio totale (tipicamente uno scalare), che viene infine consegnato al modulo di decisione. L'intero processo è raffigurato in figura 2.3

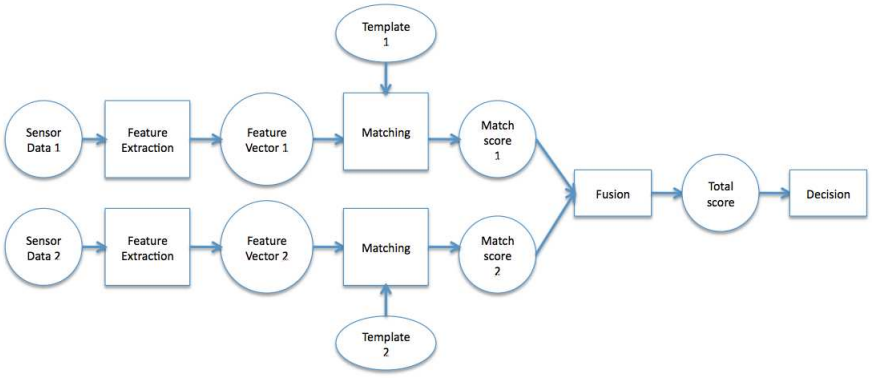

Figura 2.3: Fusione a livello di matching score

Il flusso di processo all'interno di un sottosistema è lo stesso che in un singolo sistema biometrico, permettendo in tal modo l'impiego di algoritmi consolidati sia per l'estrazione delle feature che per il matching.

I punteggi di similarità relativi a ciascun sistema unimodale considerato vengono quindi normalizzati e combinati impiegando una delle seguenti strategie (il processo di normalizzazione sarà oggetto di approfondimento in un paragrafo apposito):

- La *regola della somma*: rappresenta la più semplice forma di combinazione e consiste nel considerare la media pesata dei punteggi ottenuti

dal riscontro sui singoli sottosistemi. Questa strategia può essere applicata adottando pesi uguali per ogni tratto oppure computando pesi specifici per ogni utente;

- L'*albero di decisione* (decision tree): tale strategia impiega una sequenza di confronti di soglia sui vari punteggi per assumere una decisione di autenticazione. Un albero di decisione deriva una sequenza di regole di tipo if-then-else utilizzando un particolare training set allo scopo di assegnare una etichetta di classe (categorizzare) al dato input. Questo risultato viene ottenuto individuando un attributo (feature) che massimizza il guadagno informativo ad un particolare nodo. Le soglie possono essere computate utilizzando un software di machine learning basato sugli alberi come il C5.0[3] per massimizzare il guadagno di informazione ottenibile in ciascun confronto. La figura 2.4 ne rappresenta la realizzazione grafica. Ovviamente altri meccanismi si calcolo sono evidentemente possibili;

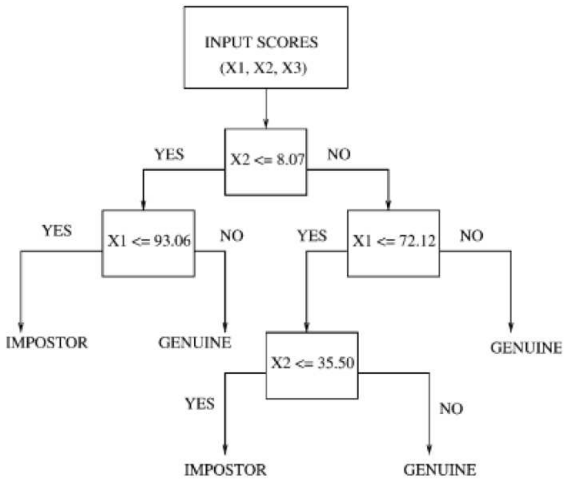

Figura 2.4: Esempio di decision tree operante su algoritmo C5

- L'*analisi discriminante lineare* trasforma i vettori di punteggio n-dimensionali (dove n corrisponde al numero di tratti distinti impiegati) in un nuovo sottospazio, nel quale la separazione tra le classi di punteggio degli utenti genuini e degli impostori è massimizzata. I parametri ottimali per questa trasformazione vengono calcolati anticipatamente basandosi su un determinato training set. Il punteggio di output viene definito come la minima distanza dai centroidi delle due classi, impiegando una particolare metrica: la distanza Mahalanobis.

[3]Induction of Decision Trees, `http://www.cse.unsw.edu.au/`

Basandosi su dati sperimentali, finora si è osservato come la regola della somma raggiunga le migliori performance. Una nuova e più importante direzione è rappresentata da una particolare estensione alla regola della somma, un approccio che suggerisce di applicare pesi specifici per ogni utente ai tratti individuali in modo da essere combinati impiegando delle soglie anch'esse user-specific, allo scopo di rendere ulteriormente accurata la decisione finale.

L'ultimo interrogativo al quale rispondere è quando questo approccio porta realmente ad un incremento significativo dell'accuratezza. I risultati sperimentali, al momento, indicano un aumento della performance combinando gli identificatori biometrici, ma non così significativo rispetto all'utilizzo ad esempio del singolo sistema composto dall'impronta digitale. Questo è dovuto probabilmente al fatto che si impiegano singoli sistemi che sono ancora piuttosto deboli rispetto ad una tecnologia ormai matura come quella per il riconoscimento basato su impronta digitale. Al crescere della validità dei verificatori si avrà gioco-forza il raggiungimento delle performance attese.

2.4.3 Fusione a livello di decisione

Adottando questa strategia di fusione [32], viene realizzata una decisione di autenticazione separata per ciascun tratto biometrico. Queste decisioni sono poi combinate in un voto finale, come evidenziato in figura 2.5

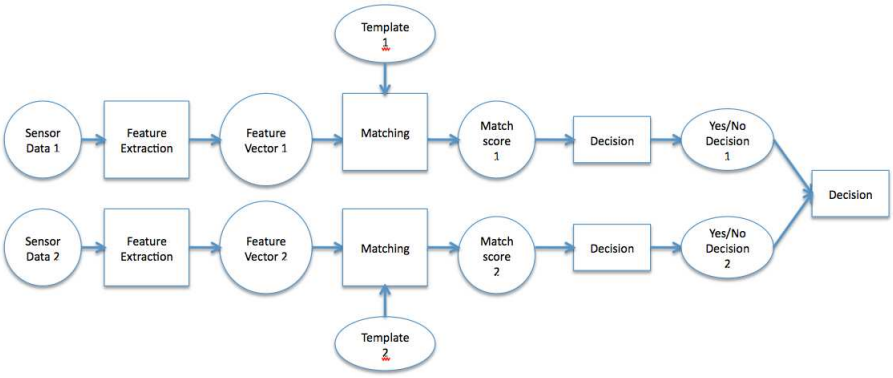

Figura 2.5: Fusione a livello di decisione

La fusione a livello di decisione rappresenta un'architettura di sistema piuttosto disaccoppiata, con ogni sottosistema che si comporta esattamente come un singolo sistema biometrico. Tale tipologia di architettura è quindi diventata sempre più popolare tra i venditori del ramo della biometria, spesso pubblicizzata con il nome di "biometria layered". L'avvento di standard biometrici come BioAPI ha ulteriormente sostenuto questo concetto.

Sono disponibili varie strategie per combinare le diverse decisioni nella decisione di autenticazione finale. Si spazia dal voto a maggioranza, a sofisticati metodi basati su base statistica (come la fusione Bayesiana). Nella pratica, tuttavia, gli sviluppatori sembrano preferire il metodo più semplice: *l'unione booleana*. Le strategie maggiormente utilizzate per quanto concerne quest'ultimo tipo di unione, sono le seguenti:

- la **AND rule** richiede una decisione affermativa da parte di tutti i moduli di verifica (unanimità). Un approccio di questo tipo porterà ad ottenere un basso livello di false autenticazioni, ma aumenterà in maniera cospicua il FRR;

- la **OR rule** prova ad autenticare l'utente impiegando un singolo tratto. Nel caso in cui questo fallisca, si tenta l'autenticazione con la caratteristica successiva (è sufficiente una sola risposta affermativa). Una politica di questo tipo si traduce in un basso FRR, in cambio di un alto FAR;

- una regola molto più interessante è la cosiddetta **random rule**, secondo la quale viene scelto a caso un tratto biometrico dal pool di tratti disponibili. Sebbene questa idea sia piuttosto semplicistica, può certamente rappresentare un buon meccanismo di prevenzione dallo spoofing; per contro, però, si presenta l'inconveniente dovuto al fatto di avere una acquisizione multilivello ad ogni tentativo di autenticazione.

La fusione a livello di decisione avviene ad uno stadio molto avanzato (temporalmente parlando) nel processo di autenticazione. Si può quindi ipotizzare che non faccia intravedere il medesimo potenziale di miglioramento nella performance complessiva del sistema, che si può scorgere invece nella fusione a livello di matching score. Solamente sotto alcune condizioni specifiche, i miglioramenti nell'accuratezza possono essere garantiti. Se queste condizioni vengono violate utilizzando test biometrici che differiscono significativamente tra loro in termini di performance, la loro combinazione a livello di decisione può portare ad avere una notevole degradazione delle performance.

2.4.4 Ulteriori livelli di fusione

Da quanto visto finora, si può riassumere che una delle difficoltà maggiori nel progettare un sistema multibiometrico è rappresentato dalle varie opportunità di scelta che si hanno a disposizione man-mano che ci si addentra nelle specifiche realizzative. Nello specifico, relativamente a quanto visto per la fusione delle informazioni, esistono diverse modalità di intervento, che si possono schematizzare in *pre-classificazione*, o fusione prima del matching, e *post-classificazione* o classificazione dopo il matching. La figura

2.6 illustra in via schematica come sono collocate le alternative viste finora rispetto a questo tipo di classificazione.

Dalla figura si evince altresì che esistono due alternative di fusione: la fusione a *livello di sensore* e quella a *livello di rank* [33]. Le due modalità appena citate sono riportate in questa sezione per completezza di informazione; è bene sapere che non vengono mai considerate nella pratica per fini realizzativi.

- *Fusione a livello di sensore*: il dato grezzo acquisito da più sensori viene processato ed integrato per generare un nuovo dato dal quale poter estrarre le feature. Ad esempio, nel caso della biometria del volto, sia l'informazione della texture 2D che l'informazione 3D (profondità, ottenuta utilizzando due diversi sensori) possono essere fuse per generare un'immagine della texture 3D del volto, la quale sarà successivamente soggetta sia al processo di feature extraction che al matching. Come si può immaginare, il dato acquisito a questo livello rappresenta la sorgente più ricca di informazioni, sebbene ci si aspetti che sia contaminata da rumore (illuminazione non uniforme, sfondi ingombranti, ecc.). In definitiva, la fusione a livello di sensore fa riferimento al consolidamento del dato grezzo ottenuto impiegando più sensori, oppure diverse istantanee di un tratto adoperando un singolo sensore;

- *Fusione a livello di rank*: questo tipo di combinazione diviene importante nei sistemi di identificazione (uno-a-molti), nei quali ciascun classificatore associa un rank ad ogni identità registrata (più alto il rank, migliore il match). In questo modo, la fusione comporta il consolidamento di rank differenti associati ad una specifica identità e la determinazione di un nuovo rank che possa aiutare nello stabilire la decisione finale. Quindi il meccanismo di ranking fornisce una visione più ampia del processo di decisione rispetto ad un matcher che fornisce solo l'identità del candidato migliore; al contempo, però, rivela meno informazione rispetto al punteggio di similarità. Tuttavia, a differenza di quest'ultimo, gli output dei processi di rank dei vari sistemi biometrici utilizzati sono confrontabili. Come prima evidente conseguenza, non viene richiesto alcun tipo di normalizzazione; questo rende gli schemi di fusione basati sul rank più semplici da implementare, se confrontati con quelli basati sul livello di fusione matching score.

2.5 Normalizzazione del matching score

I sistemi biometrici multimodali consolidano le prove fornite da diverse sorgenti biometriche e tipicamente di ottengono migliori performance nel

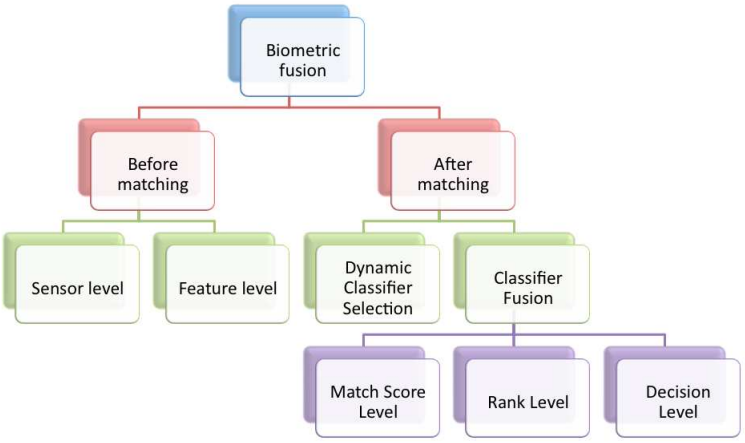

Figura 2.6: Tipologie di fusione in un sistema multibiometrico

riconoscimento rispetto a quelli basati su di una singola caratteristica biometrica. Sebbene la fusione delle informazioni in un sistema multimodale, come analizzato nel paragrafo precedente, possa essere realizzata a vari livelli, l'integrazione al livello di matching score rappresenta l'approccio più utilizzato comunemente, dovuto soprattutto alla facilità di accesso e combinazione dei punteggi generati dai differenti matcher. Siccome l'output dei punteggi di similarità forniti dai diversi tratti analizzati è eterogeneo, la normalizzazione dei punteggi diventa un fattore necessario al fine di portare questi punteggi in un dominio comune, prima di combinarli. Nelle prossime righe verranno esposte brevemente le tecniche di normalizzazione più diffuse in letteratura ed i risultati che derivano dal loro impiego in ambito sperimentale per quanto concerne i sistemi biometrici multimodali [34].

Con riferimento ad un sistema multimodale che prevede l'impiego dell'impronta digitale, della rilevazione del volto e dell'iride, per fare un esempio, si ha che al termine del processo di matching sui singoli tratti, i punteggi ottenuti sono di natura diversa: mentre per quanto riguarda il primo tratto riportato si ricava un'informazione di similarità, per gli altri due il punteggio rappresenta un risultato di distanza tra il template corrente e quello memorizzato. A questo punto risulta del tutto evidente la non-omogeneità tra i vari matching score, cosa che non permette la loro combinazione diretta. Si rende necessaria quindi un'operazione di normalizzazione prima di procedere.

La normalizzazione del punteggio fa riferimento al cambiamento dei parametri di posizione e scalatura delle distribuzioni dei punteggi di matching che si ottengono come output nel processo di confronto dei singoli parametri, in modo tale da trasformarli tutti in punteggi a dominio comune. Quando i parametri utilizzati per la normalizzazione vengono determinati utilizzando

un training set fissato, si fa riferimento alla normalizzazione a punteggio fisso. In questo caso, viene esaminata la distribuzione del punteggio di score del training set e successivamente viene scelto un opportuno modello per l'adattamento. I parametri di normalizzazione vengono stimati di volta in volta in base al feature vector corrente. Questo tipo di approccio, ovviamente, ha la caratteristica di adattarsi alle variazioni dei dati in input, come ad esempio il cambiamento della lunghezza del segnale parlato in un sistema di riconoscimento vocale.

Il problema della normalizzazione nei sistemi biometrici multimodali corrisponde al problema della normalizzazione del punteggio nella metaricerca. La metericerca è un tecnica utilizzata per combinare i punteggi di rilevanza dei documenti forniti da diversi motori di ricerca, allo scopo di migliorare il sistema di reperimento dei documenti stessi. Le tecniche di *min-max* e *z-score* rappresentano due tra le più popolari impiegate per la normalizzazione nell'ambito della metaricerca. Nella letteratura, la distribuzione dei punteggi dei documenti rilevanti viene generalmente approssimata come una distribuzione Gaussiana con ampia deviazione standard, mentre i documenti giudicati come non rilevanti vengono approssimati con una distribuzione esponenziale. Negli esperimenti condotti in multibiometria relativamente ai tre tratti sopra citati, le distribuzioni dei punteggi di utenti genuini ed impostori, per quanto concerne le impronte digitali, seguono in maniera evidente la distribuzione dei documenti rilevanti/non rilevanti della metaricerca. Tuttavia, i punteggi determinati su volto e iride non esibiscono questo tipo di comportamento.

Ottenere un buon schema di normalizzazione si traduce nello stimare in maniera *robusta* ed *efficiente* i parametri di posizione e scalatura della distribuzione del punteggio di similarità. La robustezza si riferisce all'insensibilità, alla presenza di possibili deviazioni (outliers). L'efficienza invece rappresenta il grado di prossimità della stima ottenuta dalla stima ottima quando la distribuzione è un parametro conosciuto.

2.5.1 Min-Max

La tecnica di normalizzazione più semplice è la cosiddetta normalizzazione *Min-Max*. Tale approccio è caldamente consigliato e trova le migliori condizioni di funzionamento nei casi in cui siano noti i limiti dei punteggi prodotti da un matcher (valori massimo e minimo). In questo caso, si possono spostare con relativa semplicità gli score minimo e massino rispettivamente a 0 e 1. Tuttavia, anche qualora i punteggi di similarità non siano limitati, è possibile stimare i valori minimo e massimo per un set di punteggi (una sorta di campione) e successivamente applicare la normalizzazione min-max. Dato un insieme di matching score s_k, $k = 1, 2, \ldots, n$, i punteggi

normalizzati sono dati da:

$$s'_k = \frac{s_k - min}{max - min} \tag{2.1}$$

Nei casi in cui i valori minimo e massimo siano stimati da un insieme di punteggi di similarità, il metodo risulta non robusto (altamente sensibile ad outlier nei dati impiegati per la stima). La normalizzazione min-max conserva la distribuzione originale dei punteggi ad eccezione di un fattore di scala e riporta tutti gli score in un intervallo comune $[0, 1]$. I punteggi di distanza possono essere trasformati in punteggi di similarità sottraendo il valore normalizzato min-max da 1.

Il *decimal scaling* rappresenta un particolare approccio derivato da min-max e può essere applicato allorchè i punteggi dei diversi matcher siano su scala algoritmica. Ad esempio, se un matcher fornisce un punteggio nell'intervallo $[0, 1]$, mentre gli altri esibiscono un punteggio nel range $[0, 1000]$, è possibile applicare la seguente normalizzazione:

$$s'_k = \frac{s_k}{10^n} \tag{2.2}$$

dove $n = \log_{10} max(s_i)$. I problemi che emergono con questo approccio sono la mancanza di robustezza e l'assunzione che i punteggi ottenuti varino di un fattore logaritmico.

2.5.2 Z-score

La tecnica di normalizzazione dei punteggi più comune è la z-score, calcolata utilizzando la media aritmetica e la deviazione standard dei dati ottenuti. Ci si attende che questo tipo di schema fornisca i risultati migliori se si ha una conoscenza a priori (quindi la disponibilità) dei valori relativi al punteggio medio e alle variazioni standard dei punteggi relativi ai vari matcher impiegati. Anche in questo caso, se non si ha alcuna conoscenza circa la natura degli algoritmi di matching, si rende necessaria una stima di media e deviazione standard dei punteggi a partire da un insieme di risultati noti. Per quanto concerne la normalizzazione z-score, la formula è la seguente:

$$s'_k = \frac{s_k - \mu}{\sigma}, \tag{2.3}$$

dove μ rappresenta la media aritmetica e σ la deviazione standard. Tuttavia, sia la media che la deviazione standard. Tuttavia, sia la media che la deviazione standard sono sensibili alle variazioni e, quindi, ne deriva che questo metodo non è robusto. La normalizzazione z-score non garantisce un intervallo numerico comune per i punteggi normalizzati dei differenti matcher. Se i punteggi di input non presentano distribuzione Gaussiana, la normalizzazione z-score non ne mantiene la distribuzione in output. Questo

si riconduce al fatto che la media e la deviazione standard sono parametri ottimali di posizione e scalatura solamente per distribuzioni di tipo Gaussiano. Per una distribuzione arbitraria, media e deviazione standard rappresentano stime anche ragionevoli, ma non valori ottimali.

2.5.3 Altre tecniche di normalizzazione meno comuni

Tra le tecniche meno applicate, ma non per questo meno valide, è doveroso citarne almeno tre: *median* e *median absolute deviation* (MAD), la *funzione a doppia sigmoide* e la *stima della tangente iperbolica tanh*. Per una descrizione approfondita si rimanda a [35], qui tralasciata perchè non utilizzata ai fini della tesi. Basti sapere che vi sono approcci di varia natura che sono efficienti e/o insensibili agli outlier; non si pensi però che rappresentino la panacea di tutti i mali: ogni metodo presenta dei punti di malfunzionamento in determinate condizioni, per determinate caratteristiche. La tabella sottostante riassume le tecniche ad oggi presenti in letteratura con relativi punti di forza e debolezza.

2.5.4 Considerazioni

Le tecniche di normalizzazione citate in questo paragrafo sono state testate sul sistema multibiometrico caratterizzato dai tratti sopracitati (impronte digitali, volto, iride). Una volta attuata la normalizzazione, i punteggi sono stati combinati utilizzando varie strategie, tra cui la somma (descritta nella sezione relativa alla fusione a livello di matching score). E' stato osservato che un sistema multimodale che impiega il metodo della somma fornisce prestazioni migliori rispetto alla resa del miglior sistema unimodale, questo per *ciascuna* tecnica di normalizzazione eccetto la strategia MAD. A titolo di esempio, per un FAR dello 0.1%, il GAR del singolo modulo per l'impronta digitale risulta essere attorno allo 83.6%, mentre quello del sistema multimodale arriva al 98.6% con la tecnica di normalizzazione z-score. Questo miglioramento nella performance è significativo e sottolinea il beneficio derivante dalla multimodalità.

Tra le diverse tecniche di normalizzazione, si osserva che tanh e min-max danno risultati che superano qualsiasi altra tecnica quando si hanno bassi livelli di FAR. Ad alti livelli di FAR, z-score lavora meglio rispetto a tanh e min-max. La differenza complessiva nelle performance tra gli approcci min-max, z-score e tanh è minima (chiaramente, per quanto concerne a combinazione di questi tratti).

In conclusione, le tecniche di normalizzazione min-max, z-score e tanh, seguite da una semplice somma lineare nella combinazione dei punteggi, risultano in un GAR superiore rispetto a tutte le altre strategie. Sia min-max che z-score sono sensibili alle variazioni. D'altro canto, il metodo tanh è sia robusto che efficiente. Se i valori chiave (valori minimo e massimo

per min-max, oppure media e deviazione standard per z-score) delle singole modalità rilevate sono note in anticipo, le tecniche di normalizzazione più semplici come min-max e z-score risultano essere più che sufficienti. Nell'eventualità che questo non accada, bisogna procedere con una stima dei parametri attraverso uno specifico training set.

2.6 Pesi user-specific nei sistemi multibiometrici

Giunti a questo punto della trattazione si ha un'idea precisa di come funzioni il meccanismo della multibiometria (a livello generico di blocchi funzionali); sono stati altresì approfondite questioni più avanzate come il tipo di fusione dei dati acquisiti e la normalizzazione degli stessi nel caso in cui si adotti la fusione a livello di matching score. Sono state illustrate anche le possibilità di combinazione dei punteggi per ottenere un risultato finale. Un ulteriore passo verso un miglioramento delle performance di un sistema multibiometrico può essere compiuto introducendo il concetto di parametri user-specific. In questo paragrafo verrà esposto con dovizia di particolari il concetto [36].

La parola chiave che accompagna la descrizione di questo argomento è *soglia* (threshold). Gli esperimenti condotti finora nell'ambito dei sistemi multibiometrici indicano come la regola della somma, utilizzata con punteggi normalizzati, fornisca i migliori risultati operando con fusione a livello di matching score (che rappresenta il livello di fusione largamente più utilizzato). Un sistema di questo tipo può essere ulteriormente migliorato in termini di performance mediante l'impiego di soglie e pesi specifici per utente relativamente al singolo tratto biometrico componente il sistema. Le soglie vengono utilizzate per decidere se un matching score corrisponde ad un impostore oppure ad un utente genuino. Punteggi maggiori alla soglia di matching indicano un utente genuino; punteggio inferiori alla soglia suggeriscono che si tratti di un impostore. I sistemi biometrici, invece, adoperano tipicamente una soglia comune a tutti gli utenti. La pesatura rappresenta invece un secondo meccanismo utilizzato al fine di variare l'importanza dei matching score di ciascun tratto biometrico. Risulta evidente che anche questo accorgimento conduce ad un miglioramento ulteriore delle performance complessive del sistema.

L'apprendimento automatico e l'aggiornamento dei parametri del sistema aiutano a ridurre gli error rate associati ad un individuo, contribuendo quindi al miglioramento dell'accuratezza complessiva del sistema. In un sistema di tipo multibiometrico, è essenziale che ai differenti tratti biometrici siano assegnati vari gradi di importanza per utenti diversi. Questo diventa rilevante specialmente quando il tratto biometrico di qualche utente non possa essere acquisito in maniera affidabile. Ad esempio, utenti che hanno le dita persistentemente secche potrebbero non essere in grado di fornire

impronte digitali di buona qualità. Questo tipo di soggetti potrebbe andare incontro ad un alto numero di false reject interagendo con il sistema di riconoscimento. Riducendo il peso del tratto relativo all'impronta digitale, e aumentando i pesi associati agli altri tratti, il false reject error rate di questi utenti può essere ridotto. In aggiunta, diversi utenti sono inclini a diversi tipi di errore. Il FRR degli individui con un'ampia variabilità intra-classe può essere elevato. In maniera del tutto simile, il FAR associato ad utenti con bassa variazione intra-classe può risultare elevato. L'apprendimento di parametri specifici per utente si ottiene mediante osservazione delle performance del sistema lungo un determinato periodo di tempo. Questo attrarrà quel segmento della popolazione avverso all'interazione con un sistema che richiede costantemente ad un utente di fornire molteplici letture dello stesso tratto biometrico. L'enfasi consiste nel regolare i parametri di sistema automaticamente, ed in maniera appropriata, per giungere ad un guadagno di performance. Quindi, l'adattamento in un sistema multibiometrico implica i seguenti punti:

1. sviluppo di *soglie di matching user-specific*;
2. assegnazione di *pesi ai tratti biometrici individuali.*

2.6.1 Soglie user-specific

La soglia di matching per ciascun utente viene calcolata utilizzando l'istogramma cumulativo dei punteggi degli impostori per ciascun tratto considerato, come segue:

1. Per l'i-esimo utente nel database, sia $t_i(\gamma)$ la soglia nell'istogramma cumulativo che conserva γ frazione dei punteggi, $0 \leqslant \gamma \leqslant 1$.
2. Utilizzando $t_i(\gamma)$ come soglia di matching, calcolare $FAR_i(\gamma)$, $GAR_i(\gamma)$; dove GAR rappresenta Genuine Accept Rate.
3. Calcolare il FAR e il GAR complessivo in questo modo: $FAR(\gamma) = \sum_i FAR_i(\gamma)$, $GAR(\gamma) = \sum_i GAR_i(\gamma)$.
4. Utilizzare $FAR(\gamma)$ e $GAR(\gamma)$ per generare la curva ROC.

Quando il sistema biometrico è sviluppato, il γ corrispondente ad uno specifico FAR viene usato per invocare l'insieme di soglie specifiche per utente, $t_i(\gamma)$.

2.6.2 Pesatura individuale dei tratti biometrici

Ogni tratto biometrico fornisce un punteggio di matching basato sul feature set in input ed il template con il quale viene confrontato. I punteggi ottenuti vengono successivamente pesati in accordo con il tratto biometrico

utilizzato (W_1 per il tratto 1,W_2 per il tratto 2, W_3 per il tratto 3), allo scopo di ridurre l'importanza delle caratteristiche meno attendibili (ed aumentare la rilevanza di quelle più attendibili). L'operazione di pesatura dei punteggi derivanti dal matching può essere realizzata nei seguenti modi:

- Pesando in maniera uguale tutti i tratti ed impiegando una soglia di matching user-specific: vengono assegnati pesi uguali ai punteggi singoli ottenuti dai tratti W_1, W_2 e W_3, ed il nuovo punteggio complessivo è ottenuto come $S_{fus} = \sum_{k=1}^{3} \frac{1}{3} S_k$. La soglia user specific viene calcolata utilizzando la distribuzione degli impostori di S_{fus} (per ciascun utente) impiegando il procedimento illustrato nella sezione precedente.

- Stimando i pesi user-specific per mezzo di ricerche esaustive impiegando una soglia di matching comune: i pesi specifici per utente vengono stimati da un training data set come segue:

  1. Per l'i-esimo utente nel database, cambiare i pesi $W_{1,i}$, $W_{2,i}$ e $W_{3,i}$ nell'intervallo $[0, 1]$, con il vincolo $W_{1,i} + W_{2,i} + W_{3,i} = 1$. Calcolare $S_{fus} = W_{1,i}S_1 + W_{2,i}S_2 + W_{3,i}S_3$.
  2. Scegliere l'insieme di pesi che minimizza l'error rate totale associato ai vari punteggi. L'error rate totale è rappresentato dalla somma del FAR e del FRR.

2.7 Stato dell'arte

La letteratura recente sulla multibiometria + piuttosto ampia e in questo paragrafo cercheremo di riassumere le ricerche piè rilevanti.

In generale, la combinazione di diversi sistemi basati su pattern recognition è stata studiata in [41]; in applicazioni correlate ad audio-video speech processing [42], [43], [44]; in speech recognition - alcuni esempi di approcci sono: multi-band [45], multi-stream [46], [47], front-end multi-feature [48] e union model [49]; in maniera assemblata [50]; in audio-video person authentication [51]; e in multibiometria [52], [53], [54], [55]. Uno dei primi lavori indirizzati alla fusione multibiometrica risale al 1978 [56], quindi la sua storia risale ad oltre trent'anni fa. Recenti studi hanno focalizzato l'attenzione sulla qualità della fusione dei risultati [57], [58], [59], [60], [61], dove la qualità associata al modello ed i risultati delle valutazioni biometriche vengono presi in considerazione a livello decisionale. A questo proposito, una pletora di quality measure sono state proposte in letteratura per varie modalità biometriche come l'impronta digitale [62], [63], l'iride [64], il volto [65], la voce [66], la firma [67] e misure classifier-dipendent (confidenza) [68], [69]. Le misure di qualità proposte, in generale, mirano a quantificare il grado di eccellenza o di conformità di campioni biometrici ad alcuni criteri predefiniti che inlfuenzano le prestazioni del sistema. Per esempio, per l'identificazione

del volto vengono considerate la messa a fuoco dell'immagine, il contrasto e la face detection reliability.

Nella fusione basata sulla qualità, i match score di campioni biometrici di qualità superiore hanno un peso maggiore, nel calcolo del punteggio finale combinato. Ci sono due modi per incorporare misure di qualità in un classificatore di fusione, a seconda del loro ruolo, cioè, se si tratta di un parametro di controllo o un parametro di prova. Il ruolo principale delle misure di qualità è quello di essere utilizzate per modificare il modo in cui un classificatore viene impostato o testato, come suggerito nel classificatore Bayesian-based chiamato "esperto di conciliazione" [57], nel classificatore polinomiale ridotto [70], nel quality-controlled support vector [58] e nella regola fissa di fusione quality-based [71]. In ruolo secondario, le misure di qualità sono spesso correlate con degli output prima di essere fornite al classificatore, come accade nella regressione logistica [59] e nei classificatori Gaussiani e Bayesiani [60].

Altri lavori di notevole importanza riguardano l'utilizzo delle reti Bayesiane per valutare il complesso rapporto tra gli output e le misure di qualità, come ad esempio la rete Bayesiana Maurer e Baker [72] e la ricerca di Poh sulla qualità state-dependent [73]. Il lavoro in [70] prende in considerazione un array di misure di qualità piuttosto che considerarne una sola; in questo modo, raggruppandole la strategia di fusione può essere effettuata separatamente per ogni cluster.

Altre ricerche suggeriscono l'uso di misure di qualità per migliorare l'interoperabilità tra dispositivi biometrici [74], [75]. Tale approccio è comunemente utilizzato nella verifica del parlante [76] in cui vengono utilizzate strategie diverse per diversi tipi di microfoni.

Ultima ma non meno importante, un'altra promettente direzione di fusione è quella di considerare la stima di affidabilità di ogni modalità biometrica. In [77], l'affidabilità stimata per ogni modalità biometrica è stata usata per combinare decisioni symbolic-level ([78], [79], [80] e [69]), dove erano state considerate fusioni score-level. Tuttavia, in [78], [79], [80], il termine failure prediction è stato usato. Tale informazione, proveniente esclusivamente da output noti, è risultata essere efficace per ogni singola modalità biometrica [78], attraverso la fusione di sensori per una singola modalità biometrica [79], e attraverso differenti tecniche di apprendimento automatico [80]. In [69], la nozione di affidabilità è catturata marginalmente: sono ancora aperte questioni di ricerca su come si possa definire esattamente l'affidabilità, come debba essere stimata e come potrà essere utilizzata efficacemente nella fusione.

2.7.1 Analisi di alcuni casi sperimentali

La letteratura sulla biometria abbonda di esempi sulla fusione multibiometrica. In questo paragrafo, metteremo in luce tre esempi illustrando i

benefici in tre contesti differenti.

Il primo caso illustra il potenziale della biometria multimodale nel miglioramento delle performance, rispetto al caso unimodale. Il secondo caso illustra i benefici dell'uso di misure della qualità nella fusione; infine, il terzo esempio dimostra la possibilità di ottimizzare il costo dell'autenticazione per un dato target, gestendo opportunamente la scelta dei parametri biometrici coinvolti. Tale analisi cost-based permette, per esempio, di decidere se combinare più elementi biometrici, piuttosto che unire campioni multipli dello stesso parametro biometrico (riutilizzando lo stesso dispositivo), in quanto quest'ultimo scenario è sicuramente meno costoso.

Il primo esempio, che illustra i vantaggi dell'acquisizione multimodale, descritto in [81], è caratterizzato dalla fusione dei seguenti parametri biometrici: volto, voce e movimento delle labbra. Il sistema che ha usato tecnologie convenzionali off-the-shelf, sfrutta il database XM2VTS [82], producendo risultati ottenuti in accordo con il Protocollo Sperimentale Lausanne nella Configurazione 1, come mostrato in tabella 2.1. Sebbene le performance individuali delle modalità siano mediocri o basse, ad eccezione di una modalità vocale, la fusione di questi parametri ha comportato performance migliorate, come mostrato in tabella 2.2. I risultati mostrano che la fusione multimodale ha potenzialmente migliorato le performance dell'unico miglior parametro biometrico, anche se alcuni dei componenti del sistema raggiungono tassi di errore di un ordine di grandezza peggiore del miglior parametro biometrico. E' interessante notare che la combinazione dei tre migliori parametri è solo migliore marginalmente rispetto alla migliore modalità voce. Analizzando nel dettaglio le modalità face e voice, nella seconda riga della tabella 2.2 possiamo notare che nelle due modalità l'esperimento di fusione coinvolge più algoritmi per la stessa modalità, e i pesi assegnati al peggiore algoritmo sono maggiori di quelli associati ai migliori algoritmi per ogni modalità. A quanto pare, la varietà offerta da questi algoritmi più deboli ha portato a risultati molto migliori della fusione di due algoritmi di elevata precisione. Questo, in combinazione con altri algoritmi di verifica del volto e della voce, la fusione di tre modalità ottiene un fattore di incremento delle prestazioni pari a cinque, rispetto al migliore singolo parametro biometrico.

Il secondo esempio, dimostra i benefici derivanti dall'uso di misure di qualità nella fusione [59]. Nello studio referenziato, la regressione logistica era utilizzata come fusion classifier e viene prodotto un output che approssima la probabilità a posteriori di osservare un vettore di match score (denotato con x) degli elementi che corrispondono alle uscite dei sistemi di base. Le misure di qualità (denotate con q) sono quindi considerate come input addizionali al fusion classifier. L'interazione di misure di qualità e punteggi match è esplicitamente modellata attraverso l'alimentazione del fusion classifier con le seguenti tre varianti di input: $[x, q]$ (cioè, aumentando l'osservazione di x da q tramite concatenazione), $[x, x \otimes q]$ (dove $\otimes$

Algorithm	threshold	FRR	FAR
Lips	0.50	14.00 %	12.67%
Face 1	0.21	5.00 %	4.45%
Face 2	0.50	6.00 %	8.12%
Voice 1	0.50	7.00 %	1.42%
Voice 2	0.50	0.00 %	1.48%

Tabella 2.1: Performance delle modalità sul test set (Configurazione 1)

Modalities	weights	threshold	FRR	FAR
Lips, face and voice	0.27, 0.23, 0.50	0.51 %	0.00%	1.31%
4 experts (no lips)	0.02, 0.06, 0.87, 0.05	0.50 %	0.00%	0.52%
all 5 experts	0.03, 0.01, 0.04, 0.89, 0.03	0.50 %	0.00%	0.29%

Tabella 2.2: Risultati della fusione (Configurazione 1)

denota il prodotto tensoriale) e $[x, q, x \otimes q]$. Se ci sono N_q termini in q e N_x termini in x, il prodotto tensoriale fra x e q produce $N_q \times N_x$ elementi, di conseguenza, fornendo il classificatore di fusione con un ulteriore grado di libertà al modello gli elementi del prodotto generati a coppie dai due vettori.

Il risultato di questa architettura, applicato al database XM2VTS con data set standard, è mostrato in figura 2.8. Ci sono sei sistemi di riconoscimento del volto e un sistema di riconoscimento della voce. Come risultato, il numero totale di possibili combinazioni sono $2^6 - 1 = 63$. Ogni barra in figura 2.8 contiene una statistica misurante la differenza relativa tra un sistema di fusione senza alcuna misura di qualità con una delle tre varianti di cui sopra. Come è possibile osservare, in tutti i casi, l'uso di misure di qualità riduce di circa il 40% l'errore di verifica in termini di Equal Error Rate. Tale miglioramento è possibile, soprattutto quando le modalità biometriche hanno tipi di qualità significativamente differenti. Nelle impostazioni sperimentali, la voce è stata corrotta con rumore additivo uniformemente distribuito di ampiezza variabile (da 0 dB a 20 dB), mentre il volto presentava zone illuminate diversamente (figura 2.7). Gran parte della ricerca è necessaria quando si presentano fonti di rumore sconosciute o non ben definite.

Il terzo esempio pone il problema della fusione in ottica di ottimizzazioni. Dato che l'utilizzo di più modalità biometriche implica l'uso di più di hardware e software di calcolo, eventualmente richiedendo più tempo di autenticazione e disagi ulteriori da parte dell'utente, è ragionevole attribuire un costo astratto per ogni aggiunta di un dispositivo biometrico. L'obiettivo di ottimizzazione in questo contesto può essere la ricerca dell'insieme di sistemi biometrici candidati che minimizza il costo totale delle operazioni.

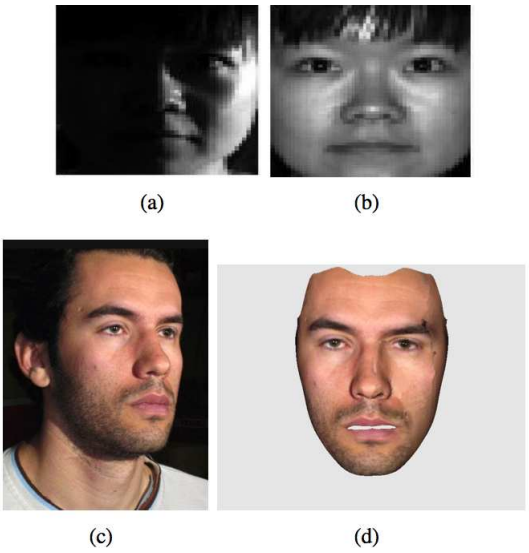

Figura 2.7: (a) esempio di illuminazione laterale e frontale (b); (c) posa fuori piano e (d) posa corretta usando un modello 3D.

Purtroppo, la strategia maggiormente utilizzata sfrutta la valutazione empirica che, come la cross-validation, ha dimostrato di essere particolarmente sensibile alla mancata corrispondenza di popolazione [83]. Come possibile soluzione, il Chernoff bound ed il suo caso speciale, il Bhattarchayya bound era stato proposto come alternativa. Bisogna prestare attenzione al fatto che il bound assume che i punteggi di matching siano uniformemente distribuiti. Tale debolezza può essere superata, trasformando i punteggi match in modo tale da meglio adattarsi alle distribuzioni normali. Questo può essere ottenuto, per esempio, usando la trasformazione Box-Com [84]. In uno studio separato [85], considerando un punteggio match unidimensionale (che coinvolge solo una uscita singola del sistema), è stato dimostrato che, anche se i punteggi di un sistema biometrico non sono conformi a una distribuzione normale, le prestazioni previste in termini di Equal Error correlano molto bene con l'errore misurato empiricamente (con correlazione pari a 0,96). L'esempio usato qui, cioè [83], è una generalizzazione di [85] alla fusione multimodale. Un esempio di curva cost-sensitive è mostrata in figura 2.9.

Questo esperimento è stato condotto su Biosecure DS2 quality-based fusion[4]. In ordine di esecuzione dell'esperimento, il database di punteggi di matching è stato diviso in due partizioni di utenti enrolled, costituendo rispettivamente gli insiemi di sviluppo e valutazione. Per ogni partizione di utenti, diversi insiemi di individui sono stati usati come impostori.

Sono stati utilizzati due criteri errore teorico, cioè il Chernoff e i limiti

[4]il data set è disponibile su `http://face.ee.surrey.ac.uk/qfusion`

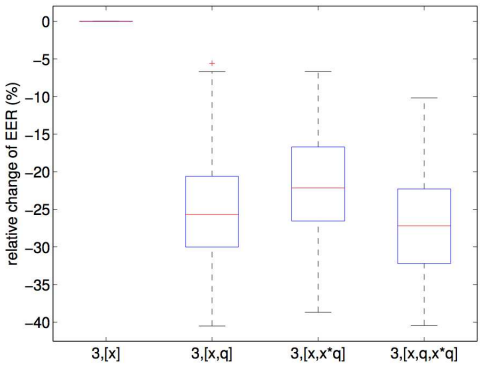

Figura 2.8: Variazioni relative di *a posteriori EER* (%) di tutte le possibili combinazioni di output di sistema implementate usando la regressione logistica.

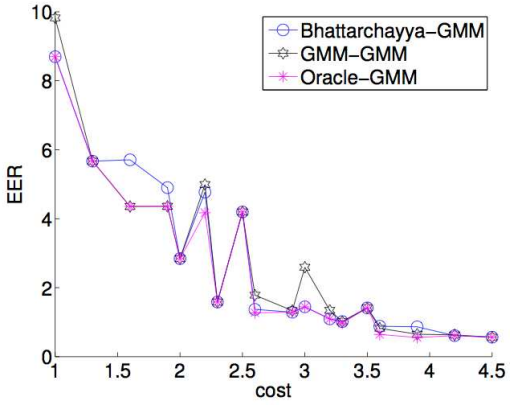

Figura 2.9: Cost-sensitive performance curve, ottenuta usando un classificatore Bayesiano

Bhattarcharyya, e due misure di errore empiriche di EER basate su classificatori bayesiani, ovvero, Gaussian Mixture Model come stimatore di densitè (indicata come MGM) e un'analisi discriminante quadratica (QDA). Queste quattro misure sono state applicate agli insiemi di sviluppo e valutazione, che consistono in popolazioni di autentici impostori e utenti qualsiasi. Per l'errore empirico, sull'insieme di sviluppo è stata applicata una procedura two-fold cross validation. L'errore medio nei due insiemi (in termini di EER) è usato come indicatore d'errore sull'insieme di sviluppo. Il classificatore addestrato sull'intero set di sviluppo viene quindi utilizzato per valutare l'errore empirico di valutazione del set.

Per esempio, nel problema di fusione in figura 2.9, il costo varia da 1 (usando un sistema semplice) a 4.5 (usando tutti gli 8 sistemi). L'intero spazio di ricerca è $2^8 - 1 = 255$. Viene così rappresentata una curva cost-sensitive "rank-one" delle performance (performance vs costo). Dato che l'obiettivo è ottenere il minimo errore di generalizzazione col minor costo,

una curva verso l'angolo in basso a sinistra è il target ideale. Questa curva è chiamata "rank-one" perchè viene mostrata solo la generalizzazione di un sistema superiore. Una curva "rank-two" cost-sensitive sarebbe il minimo degli errori di generalizzazione dei primi due candidati trovati sul set di sviluppo. Con un rank di ordine sufficiente, la curva si approssima a quella ideale (con l'errore stimato sul test set). Mentre la curva rank-one trovata con il Bhattarcharyya è soddisfacente, la curva rank-three mostra esattamente le stesse caratteristiche dell'oracolo per QDA e la rank-five per GMM. Comparativamente, usando l'errore medio cross-validated empirico, una curva rank-six è necessaria per ottenere performance obiettivo per QDA a oltre rank-ten è necessario per GMM.

Capitolo 3

Deployment

Contents

In questo capitolo verrà descritto il processo di deployment [40], cosa si intende per deployment e quali sono i metodi per effettuare il deployment presenti in letteratura.

3.1 Che cos'è il Deployment

Il deployment di un software può essere definito come il processo che si interpone tra l'acquisizione e l'esecuzione del software. E' in questo preciso momento che vengono fatte tutte le customizzazzioni e configurazioni. Il deployment di un software può essere considerato come un processo che consiste in un certo numero di attività correlate che includono:

- **rilascio**: è l'interfaccia tra gli sviluppatori e i restanti attori nel ciclo di vita del software. Al momento del rilascio il software è pachettizzato con l'aggiunta di una quantità sufficiente di metadati per descrive le risorse da cui dipende;
- **intallazione**: è il processo di trasferimento all'utente del software;
- **configurazione**: è il processo di customizzazione del software. In questo momento il software deve essere preparato per l'attivazione;
- **attivazione**: è il processo in cui si inizia l'esecuzione del software o vengono settati delle specifiche configurazioni che eseguono il software

in un preciso momento. Di solito questo processo viene fatto tramite interfaccie grafiche o tramite script o demoni;

- **disattivazione**: è il contrario del processo di attivazione, quindi è il processo che rende il software non più eseguibile;
- **monitoraggio**: è il processo in cui lo stato del software viene monitorato in modo da correggere possibili problematiche o crash in tempo reale;
- **aggiornamento**: è il processo in cui parti del software vengono cambiate con altre parti appartenenti ad una nuova versione. L'aggiornamento è un caso speciale di attivazione e molte volte richiede che il software venga prima disattivato;
- **adattamento**: è il processo di modifica del software installato in modo che possa reagire ai cambiamenti dell'ambiente in cui il software è installato;
- **Undeployment** o disinstallazione: è il processo di rimozione del software dalla macchina in cui è stato fatto il deployment;

3.2 Casi di Studio

Vengono presentati di seguito alcuni casi di studio in modo da comprendere meglio il problema e per poter studiare i principali problemi che si presentano nel caso di deplyment di software.

3.2.1 Java Beans

Enterprise JavaBeans (EJB) è uno standard per la creazione di applicazioni lato server. Java Beans è stato progettato per semplificare l'implementazione, il deployment e il mantenimento di soluzioni aziendali. Java Beans è stato progettato per semplificare la progettazione, deployment e la gestione di soluzioni aziendali. Java Beans è formato da unità di business logic contenuto all'interno di componenti che vengono eseguiti all'interno di container. Quando una business logic è inserita all'interno di un Bean questa può essere riutilizzata più volte all'interno di contesti differenti. Il container astrae l'ambiente di hosting e offre alcuni servizi quali: sicurezza, longevità di gestione e un servizio di database per le transazioni oltre ad altri servizi "low-level" quali pulizia della memoria e "caching". Ogni istanza di Bean è implementata da (almeno) una coppia di oggetti e presenta almeno tre diffenti interfacce:

- l'interfaccia *home* che viene usata per il ciclo di management;

- l'interfaccia *remota* che contiene metodi che possono essere chiamati da remoto da un client;
- l'interfaccia *locale* che definisce metodi che possono essere chiamati da clients residenti in container locali.

Java Beans deve essere pachetizzata in accordo con le linee guida rilasciate da Sun Microsystems. Il pacchetto standard contiene sia gli strumenti per la gestione che quelli per il deployment. Java Beans sono pacchettizzati in archivi JAR standard con un descrittore del deplpoyment in formato XML che descrive tutte le proprietà pertinenti al Bean. Tuttavia, quando Bean multipli vengono pacchettizzati insieme non possono essere separati e il container li gestisce come una singola unita. Il descrittore tipicamente contiene informazioni a riguardo delle transazioni del Bean, della sicurezza e della persistenza ai requisiti ti ogni specifica proprietà del Bean. Il manifesto del file JAR contiene un attributo "DependsOn" che specifica il componente da cui il/i componenti del pacchetto dipendono.

Il ciclo di vita di un Bean, come descritto da Sun, consiste di 4 fasi:

- development;
- deployment;
- serivizi di disponibilità;
- undeployment.

Nella prima fase il Bean è scritto e pacchettizzato in un file JAR come descritto sopra. Nella seconda fase un'applicazione di assemblaggio può aggiustare le proprietà di deployment del Bean tramite il settaggio degli attributi come: gli attributi di sicurezza, meccanismi di persistenza, proprietà di transazione ed ogni proprietà che deve essere fatta su misura per un certo ambiente di deployment. Questa fase finisce con la pacchettizzazione del Bean in una certa directory dove può essere attivato dal container che lo gestisce.

Il supporto è specialmente per il supporto al mantenimento degli aggiornamenti. Per concudere, Enterprise Java Beans (EJB) sono relativamente di grana fine e dipendenti dal linguaggio di programmazione utilizzato. La soluzione EJB isola i Bean dall'ambiente con dei specifici contenitori che espongono delle interfacce. EJB non hanno nessuna opzione di installazione remota dei componenti. Ci sono oltretutto dei problemi rispetto al modo in cui i Bean vengono nominati.

3.2.2 Linux

Il principale metodo per il deployment di software in Linux è il "Red Hat Package Manager" (RPM). Il manager supporta un certo numero di

operazioni quali: installazione, ricerca, verifica, aggiornamento e rimozione di pacchetti. Queste operazioni sono supportate da un database contente i dettagli dei pacchetti che devono essere installati in una particolare distribuzione Linux. Un pacchetto RPM tipicamente contiene dei file binari eseguibili e dei file di configurazione e documentazione. Dato che i file binari sono contenuti nel paccheto implicitamente esistono delle dipendenze tra il pacchetto e il sitema operativo e l'architettura host. Questo è affrontato usando un set standard di librerie C e annotando i pacchetti insieme all'architettura con i quali sono stati compilati. E' inoltre possibile creare dei pacchetti con i sorgenti, il problema è che questo approcio che elimina questo problema però aumentano il numero di dipendenze con pacchetti aggiuntivi tipo "make" e "gcc".

Ogni pacchetto RPM è nominato con una etichetta che contiene il nome del software contenuto, la versione e il numero, la distribuzione Linux e l'architettura per la quale è stato compilato. Ogni pacchetto RPM contiene 4 sezione:

- *Lead* usata per operazioni di tipo Unix come il comando "file";
- *Signature* che contiene informazioni crittografiche che possono essere usate per valutare l'integrità del pacchetto e in alcuni casi dello header e dell'archivio. L'archivio comprende una collezione di File compressi con GZip.
- *Header* Una volta utilizzate le informazione nel lead vengono sostituite da quello nello header a causa di inflessibilità;
- *Archive* Sono contenuti i file binari.

I file RPM possono oltretutto contenere un certo numero di script usando il linguaggio di scripting di Unix. Gli script sono organizzati in insiemi responsabili della compilazione del codice, dell'installazione e della cancellazione del software. Gli script di compilazione sono responsabili dell'unpacking, della compilazione, dell'installazione e della rimozione del software e eseguono delle customizzazioni del software dopo l'installazione. Gli script di post-installazione sono responsabili di controllare che l'installazione sia avvenuta correttamente. Gli script di installazione e di cancellazione vengono eseguiti 4 volte: prima e dopo l'installazione e prima e dopo la cancellazione. Dei tool di alto livello sono stati construiti usando RPM come YUM (Yellowdod Updater Modified) che è stato progettato per determinare le dipendenze intra-pacchetto e aturomatizzare l'installazione di pacchetti. Fornisce oltretutto dei tool per la gestione dei pacchetti in modo da non dover configurare ogni pacchetto RPM manualmente.

Per concludere, RPM è largamente utilizzato nel mondo reale. E' un tool di deployment a grana grossa, indipendente dal linguaggio di programmazione utilizzato e dipendente dal sistema operativo utilizzato. La sua

principale debolezza è che non tutte le sue dipendenze sono modellate esplicitamente e quelle che lo sono, non sono modellate in termini di pacchetti ma in base al loro contenuto.

Capitolo 4

Rilevamento e Riconoscimento Facciale

Contents

In questo capitolo verrà introdotto il processo di rilevamento e il riconoscimento facciale, descrivendo i principali metodi presenti in letteratura con particolare attenzione a quelli utlizzati in questa tesi. Il rilevamento facciale ha come obiettivo determinare la posizione e la dimensione di faccie umane in immagini digitali arbitrarie. Il riconoscimento facciale, invece, ha come principale obiettivo caratterizzare in modo univoco immagini facciali appartenenti a persone differenti.

Mentre il rilevamento facciale può essere considerato come uno specifico caso della "object-class" detection[1], quindi appartenente alle ricerche nel campo della computer vision e dell'image processing, il riconoscimento facciale può essere considerato come uno specifico settore della ricerca biometrica.

Vengono di seguito discusse le tecniche per il rilevamento facciale e il riconoscimento facciale usate nel seguito della tesi.

[1]Nella "object-class" detection l'obiettivo è quello di localizzare oggetti che appartengono ad una specifica classe in immagini arbitrarie

4.1 Rilevamento Facciale

In letteratura esistono diversi approci al rilevamento facciale. Uno degli algoritmi più famosi è quello di "Haar like feature" spiegato nel lavoro di Viola et al.[17] motivato in parte dal lavoro di Papageorgiou at al.[19]. Tale algortitmo è implementato nelle librerie di Computer Vision OpenCV descritte nella appendice A. Tale lavoro si basa su un'approcio "machine learning" supervisionato e definisce un framework robusto e rapido per il rilevamento di oggetti. Anche se tale framework è un framework generale per un qualsiasi problema di "object detecion" il lavoro di Viola et al. è motivato e dimostrato sul rilevamento facciale.

Verrà di seguito spiegato nel dettaglio il processo di rilevazione facciale proposto da Viola et al. Nel primo sottoparagrafo verranno introdotte le caratteristiche usate per il rilevamento, mentre nei restanti verrà introdotto un nuovo metodo di rappresentazione delle immagini in modo da rendere il processo di rilevazione più efficiente e poi verrà visto il processo di training e rivelazione con dei risultati sperimentali.

4.1.1 Feature

Il processo di rilevazione classifica le immagini basandosi sui valori di specifiche feature piuttosto che sul valore di intensità dei singoli pixel. Ci sono molte motivazioni sull'uso delle feature piuttosto che nell'uso diretto dei singoli pixel. La motivazione principale è che un sistema che opera su feature è molto più veloce di uno che lavora su pixel oltretutto, usando feature, è molto più facile rappresentare un dominio di conoscenza che sarebbe molto più difficile rappresentare usando i singoli pixel. Nel lavoro di Viola et al. vengolo usate tre tipi di feature (vedi Figura 4.1).

La feature più semplice usata (la feature A nella figura 4.1) si basa sulla funzione "Haar basis" usata da Papageorgiou et al. che a sua volta si basa sulla "Haar wavelet"[2] [20] da cui ne deriva anche il nome dell'algortimo. Il valore delle feature si ottiene sottraendo la somma dei valori di intensità dei pixel all'interno della regione scura con la somma dei valori di intensità dei pixel all'interno della regione chiara. Il numero di feature presenti in una immagine 24×24 è di oltre 180.000 un numero molto maggiore del numero totale di pixel dell'immagine. Per questo motivo verranno presentati di seguito dei metodi per selezionare un sottoinsieme di feature tra quelle più rappresentative e un modo per togliere parti dell'immagine che non devono essere processate perchè non importanti.

[2]Le Haar wavelets sono un set naturale di funzioni che codificano le differenze medie di intensità tra differenti regioni, per uno studio dettgliato delle Haar wavelets si fa riferimento alla bibliografia

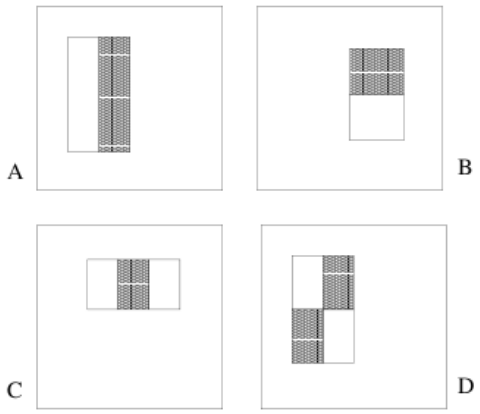

Figura 4.1: Esempi di feature rettangolari usate da Viola et al.

4.1.2 Immagini Integrali

Per il calcolo veloce delle feature viene usata una rappresentazione delle immagini adeguata chiamata "Integral Image". L'immagine integrale nella posizione (x, y) contiene la somma dei pixel sopra e a sinistra inclusi x, y:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{4.1}$$

dove $ii(x, y)$ è l'immagine integrale e $i(x, y)$ è l'immagine originale. Usando la seguente coppia di ricorrenze:

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

(dove $s(x, y)$ è la somma della riga, $s(x, -1) = 0$ e $ii(-1, y) = 0$) l'immagine integrale può essere calcolata in un solo passo dall'immagine originale. Si veda la Figura 4.2 dove la somma dei pixel all'interno del rettangolo D può essere calcolato come 4 referenze ad array. Il valore dell'immagine intgrale nel punto 1 è la somma dei pixel contenuti nel rettangolo A. Il valore nella posizione 2 è $A + B$, nella posizione 3 è $A + C$ e nella posizione 4 è $A + B + C + D$. Quindi la somma dei pixel all'interno del rettangolo D può essere calcolata come $4 + 1 - (2 + 3)$.

4.1.3 Funzione di Learning e Classificazione

Dato un set di feature e un training set di immagini positive e negative possono essere applicati molti algoritimi di training presenti in letteratura. Nel lavoro di Viola et al. viene usato una variante di AdaBoost [18]

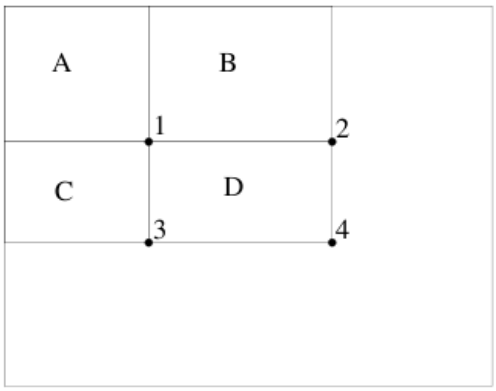

Figura 4.2: Esempio di calcolo della rappresentazione di un immagine in immagine integrale.

per selezionare un sottoinsieme di feature e per il training del classificatore. Come detto prima essendoci un numero molto grande di feature rispetto al numero di pixel, anche se una feature può essere calcolata molto velocemente grazie all'utilizzo delle immagine integrali, calcolare il set completo di feature risulta molto oneroso in termini computazionali. L'ipotesi su cui si basa tutto il lavoro è però che solo un piccolo sottoinsieme di feature può essere combinato per formare un classificatore. Il problema a questo punto è trovare questo sottoinsieme di feature. La classificazione viene quindi fatto in questo modo: per ogni feature, il classificatore AdaBoost determina la soglia ottimale della funzione di classificazione, cosicchè il minor numero di immagini risulti non-classificato. Ogni classificatore $h_j(x)$ è così definito:

$$h_j(x) = \begin{cases} 1 \; se \; p_j f_j < p_j \theta_j \\ 0 \; altrimenti \end{cases}$$

dove f_j è una feature, θ_j una soglia e $p_j = 1, -1$ che indica il segno delle disuguaglianza. Il funzionamento preciso del classificatore AdaBoost è descritto di seguito:

- Siano date n immagini di training $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ dove $y_i = 0, 1$ con $i \in 1, n$ per immagini positive e negative rispettivamente.
- Vengono inizializzati dei pesi $w_1, i = \frac{1}{2m}, \frac{1}{2l}$ per $y_i = 0, 1$ rispettivamente, dove m e l sono il numero di negativi e positivi rispettivamente.
- Per $t = 1, ..., T$

  1. Vengono normalizzati i pesi

$$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

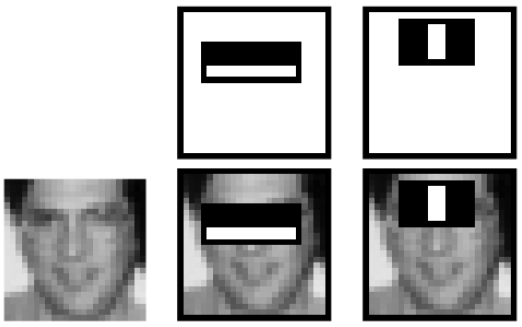

Figura 4.3: Le feature selezionate dall'algoritmo AdaBoost in alcune immagini di training del lavoro di Viola et al.

in modo che w_t sia una distribuzione di probabilità.

2. Per ogni feature j viene allenato un classificatore h_j che viene limitato all'uso di una sola feature. L'errore viene valutato in relazione a w_t, $\varepsilon_t = \sum_i w_i |h_j(x_i) - y_i|$.
3. Viene selezionato il classificatore h_t con l'errore ε_t minore.
4. Vengono aggiornati i pesi:

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$

dove $e_i = 0$ se l'immagine di training x_i è classificata correttamente, $e_i = 1$ altrimenti, e $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.

- Il risultante classificatore è:

$$h(x) = \begin{cases} 1 \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \; altrimenti \end{cases}$$

Risultati Sperimentali della Classificazione

Viene di seguito mostrato il risultato sperimentale di classificazione del lavoro di Viola et al. La feature selezionate dall'algoritmo AdaBoost sono le feature A e C dell'immagine 4.1 il cui significato può essere facilmente interpretato dato che di solito la regione degli occhi è più scura di quella della parte superiore al naso. Si veda la Figura 4.3.

4.1.4 Classificazione in Cascata

Per ridurre ancora di più il tempo di computazione, nel lavoro di Viola et al., si fa uso di una cascata di classificatori che cerca di incrementare

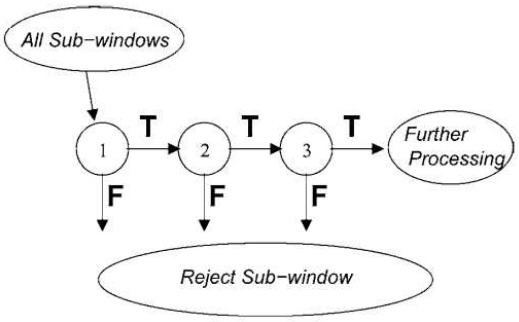

Figura 4.4: Processo di classificazione a cascata proposto nel lavoro di Viola et al.

le performance di rilevamento. Il punto chiave è costruire dei classificatori semplici che possano scartare la maggior parte delle sotto-finestre prima dell'esecuzione di classificatori più complessi. Il processo di classificazione in cascata viene mostrato in Figura 4.4.

Una classificazione positiva nel primo classificatore rende possibile la classificazione nel secondo classificatore che è stato regolato in modo da ottenere un rate di rilevamento maggiore. Una classificazione positiva nel secondo classificatore rende possibile una classificazione nel terzo classificatore, e così via. Una classificazione negativa in un qualsiasi punto scarta immediatamente la sotto-finestra. Ciò che si ottiene è quindi un albero di decisione degenerato, chiamato "cascade". I vari stage di calssificazione sono costruiti regolando la soglia di classificazione in modo da minimizzare il numero di falsi negativi. Come un albero di decisione i successivi classificatori sono addestrati con questi esempi in modo da passare direttamente gli stage precedenti.

4.2 Riconoscimento Facciale

Il riconoscimento facciale automatico, a partire da sorgenti come immagini e video, è attualmente un'area di ricerca molto attiva, sopratutto per la crescente domanda di dispositivi per la verifica (match 1:1) e per l'identificazione (match 1:N) facciale da parte di realtà commerciali e governative, per scenari di accesso controllato a spazi fisici o virtuali con un alto livello di sicurezza o come aiuto nella difesa al terrorismo o criminalità. Nonostante la ricerca avanzi negli anni, il riconoscimento facciale è ancora un'attività molto impegnativa. Le principali difficoltà da superare nel riconoscimento facciale sono dovute sopratutto alla larga variabilità di aspetti come: espressione, posa, trucco, rotazioni, differenti punti di vista, occhiali, illuminazione ed età.

In letteratura esistono diverse tipologie di algoritmi che cercano di risolvere uno o più problemi tra quelli esposti e di aumentare la perfomance in termini di FAR e FRR. Gli algoritmi di riconoscimento facciale si possono classificare in due grandi categorie, in base alla modalità di estrazione delle feature: "feature-based" o "appearence-based". Nella prima categoria vengono inseriti tutti gli algoritmi in cui le feature vengono estratte usando dei descrittori facciali come: aree, distanze e angoli fra punti caratteristici delle faccia. Nella seconda categoria vengono invece inseriti tutti gli algoritmi in cui l'esatrazione delle feature utilizza proprietà globali delle immagini facciali come intensità e variazione dei colori.

Nel resto del paragrafro ci concentremo sugli algoritmi appartenti alla categoria "appearence-based" [7]. Gli algoritmi appartenti a questa categoria tipicamente procedono calcolando un vettore n-dimensionale con cui rappresentare la faccia efficientemente. Lo step successivo è quello di proiettare le immagini facciali su questi vettori e usare i coefficienti per rappresentare le faccie. I Principali algoritmi appartenti a queste categorie sono Principal Component Analysis (PCA) o Eigenfaces [8], Linear Disciminant Analisys (LDA) [9], Indipendent Component Analisys (ICA) [10], Local Feature Analysis (LFA) [11], Correlation Filters [3], Neural Networks [12] e Tensorfaces [13].

Per uno studio completo di questi algoritmi si rimanda alla letteratura presente in bibliografia. Nel resto del capito verranno illustrati i due algoritmi utilizzati in questo lavoro: Local Binary Patter (LBP) [2] e Minimum Average Correlation Energy (MACE) [1].

4.2.1 Local Binary Pattern

L'algoritmo Local Binary Pattern (LBP) appartiene alla famiglia di algoritmi di Local Feature Analisys (LFA). In questa famiglia sono raggruppati tutti quegli algoritmi in cui l'estrazione delle feature si basa su feature locali. Il vantaggio di avere delle feature locali e sicuramente quello della robustezza alla variabilità di illuminazione e la possibilità di stimare facilmente le rotazioni. In particolare l'algoritmo LBP è invariante a trasformazioni su scala di grigio questo comporta una ancor minore sensibilità alle variazioni di luce. Questa proprietà degli algoritmi LFA è molto interessante dato che, come detto in precedenza, uno dei maggiori problemi nel campo del riconoscimento facciale è quello della variazione di luminosità

Rappresentazione di un'immagine tramite LBP

L'algoritmo di Local Binary Pattern si basa su un'operatore non paramentrico che sintetizza la struttura locale di una immagine. Ad un particolare pixel p_c dell'immagine con coordinate (x_c, y_c), l'operatore LBP è definito

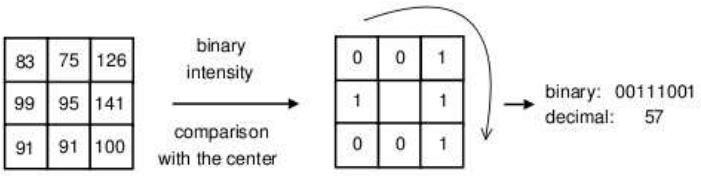

Figura 4.5: Calcolo del codice LBP per un pixel p_c

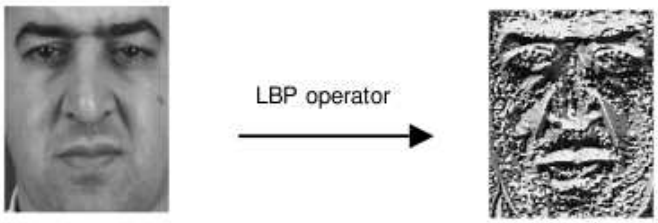

Figura 4.6: Immagine originale (sinistra) processata dall'operatore LBP (destra)

come una sequenza binaria ordinata di confronti di intensità di colore fra il pixel p_c e i suoi otto pixel adiacenti (figura 4.5).

La forma decimale della risultante parola da 8-bit (LBP-Code) letta in senso orario partendo dal pixel in alto a sinistra può essere così espressa

$$LBP(x_c, y_c) = \sum_{n=0}^{7} (i_n - i_c)2^n$$

dove i_c corrisponde al valore di grigio del pixel p_c, i_n al volre di grigio dei pixel vicini e la funzione $s(x)$ è così definita:

$$s(x) = \begin{cases} 1 \text{ se } x \geq 0 \\ 0 \text{ se } x \leq 0 \end{cases}$$

Si noti che ogni singolo bit nello LBP-code ha la stessa importanza degli altri e che due successivi valori hanno un significato totalmente differente. L'operatore LBP non è influenzato da trasformazioni monotoniche su scala di grigi e mantiene l'ordine di intensità dei pixel vicini. (figura 4.6)

Grazie alla sua proprietà discriminativa della texture e dalla sua bassa complessità computazionale, l'algoritmo LBP è molto diffuso nella "pattern recognition research".

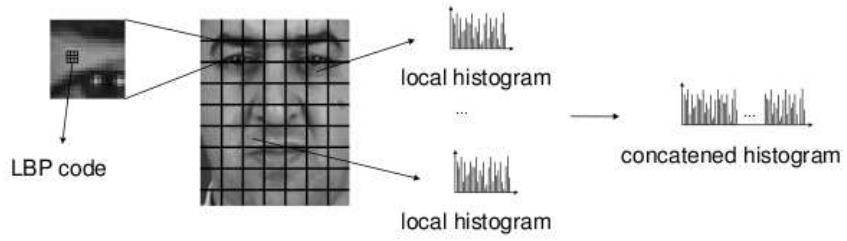

Figura 4.7: Immagine originale (sinistra) processata dall'operatore LBP (destra)

Uso di LBP per il riconoscimento facciale

Ahonen [16] ha proposto un sistema basato sulla rappresentazione della faccia tramite LBP per il riconoscimento facciale. L'immagine di test viene divisa in R piccole regioni non sovrapposte della stessa grandezze. Gli istogrammi di LBP-code H^r, con $r \in \{1, 2, ..., R\}$, sono calcolati per ogni regione e vengono successivamente concatenati in un singolo istogramma rappresentante l'immagine della faccia, si veda la figura 4.7.

Dato che alcune regioni contengono più informazioni di altre (come gli occhi), Ahonen ha proposto un metodo empirico per assegnare pesi ad ogni regione. Per la classificazione è stato usato un classificatore di tipo "nearest-neighbor" con una misura di dissimilarità di tipo Chi square (χ^2) definita in questo modo:

$$\chi^2(S, M) = \sum_{r,i} \frac{(S^r(i) - M^r(i))^2}{S^r(i) + M * r(i)} \tag{4.2}$$

dove S ed M corrispondono alla sezione e al modello di istogramma rispettivamente.

4.2.2 Filtri di Correlazione

L'approcio al riconoscimento facciale tramite filtri di correlazione si basa sull'idea di processare l'immagine nel dominio della frequenza usando delle formule chiuse di correlazione progettate per delle specifiche ottimizzazioni.

La correlazione è una misura per caratterizzare la similarità fra un pattern di referenza $r(x, y)$ e un pattern di test $t(x, y)$. Dato che, di solito, i due pattern presentano degli shift, ha senso calcolare la cross-correlazione $c(\tau_x, \tau_y)$ tra i due pattern per alcuni possibili shift τ_x e τ_y come in (4.3). Successivamente ha senso prendere il massimo tra tutte le correlazioni trovate

come misura di silimarità tra i due pattern e il picco di correlazione come shift stimato del pattern di test in riferimento al pattern di riferimento.

$$c(\tau_c, \tau_y) = \int\int t(x,y) r(x-\tau_x, y-\tau_y) dxdy \tag{4.3}$$

L'operazione di correlazione può equivalentemente essere espressa come:

$$c(\tau_c, \tau_y) = \int\int T(f_x, f_y) R^*(f_x, f_y) e^{j2\pi(f_x\tau_x + f_y\tau_y)} df_x df_y = \tag{4.4}$$

$$= FT^{-1}\{T(f_x, f_y) R^*(f_x, f_y)\} \tag{4.5}$$

dove $T(f_x, f_y)$ e $R(f_x, f_y)$ sono le trasformate di Fourier 2D di $t(x, y)$ e $r(x, y)$ rispettivamente con f_x e f_y che denotano le frequenze spaziali. L'equazione (4.4) può essere interpretata come il pattern di test $t(x, y)$ che viene filtrato da un filtro con frequenza di risposta $H(f_x, f_y) = R^*(f_x, f_y)$ per produrre in output $c(\tau_x, \tau_y)$. Da questo la terminologia "filtri di correlazione".

Matched Filter

Il filtro di correlazione in (4.4) è conosciuto in letteratura come "Matched Filter" (MF). Il filtro $H(f_x, f_y) = R^*(f_x, f_y)$ è semplicemnte il complesso coniugato della trasformata di fourier 2D del pattern di riferimento $r(x, y)$. Si può dimostrare [14] che tale filtro è ottimo per la rilevazione di pattern corrotti da rumore additivo bianco. L'ottimalità di MF però è garantita solo se il pattern di riferimento e quello di test sono identici ad esclusione del rumore additivo bianco e a traslazioni. Se invece il pattern di test differisce dal pattern di riferimento per rotazioni, scalature, ecc il filtro di correlazione MF non funziona a dovere. Un'altro problema del MF è che l'immagine di riferimento deve essere unica.

Synthetic Discrminant Function Filters

Per ovviare a questo problema Hester e Casasent [15] hanno introdotto il concetto di Synthetic Discrminant Function (SDF) Filter. Questo approcio da la possibilità di rappresentare un set di immagini di training invece che una immagine di training unica. Il filtro SDF è la somma ponderata dei singoli MF dove i pesi sono scelti in modo che la correlazione di uscita nell'origine sia entro valori pre-specificati. Per esempio la correlazione (nell'origine) corrispondente a immagini di persone autentiche può essere settata a 1 mentre quella di impostori a 0. Ciò che ci si aspetta, quindi, è che l'output del filtro di correlazione dia in uscita un picco di correlazione pari ad uno per immagini autentiche e pari a zero per immagini di impostori.

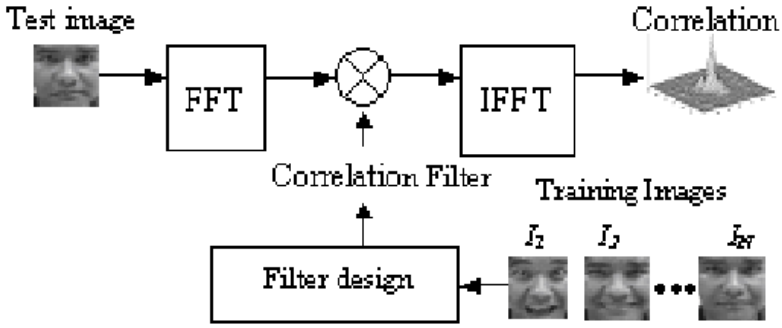

Figura 4.8: Diagramma a blocchi del processo di correlazione

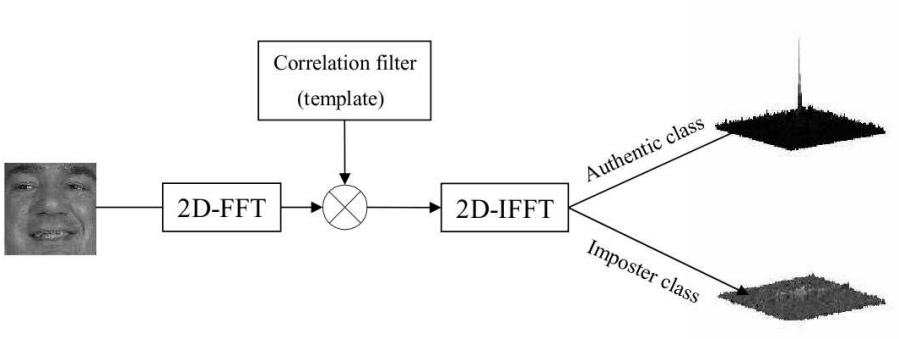

Figura 4.9: Differenza tra la correlazione di un immagine autentica e dell'immagine di un impostore

Come mostrato in figura 5.2 il riconoscimento facciale viene fatto filtrando l'immagine di test con un filtro SDF. Nella correlazione in uscita viene ricercato il picco: l'altezza del picco (relativamente al background) è usata per determinare quanto l'immagine di test è correlata alle immagini di training 5.3, la locazione del picco, inoltre, indica lo shift dell'immagine di test rispetto alle immagini di training

Minimum Average Correlation Energy Filter

Anche se il filtro SDF produce pre-specificati valori di correlazioni di uscita, può controllare solo il valore nell'origine per immagini di training centrate. Dal momento in cui le immagini di test non sono neccessariamente centrate, è praticamente impossibile conoscere dove sono collocati questi valori controllati di correlazione nell'output, a meno che non sia possibile controllare tutto il resto del piano di correlazione di uscita in modo che

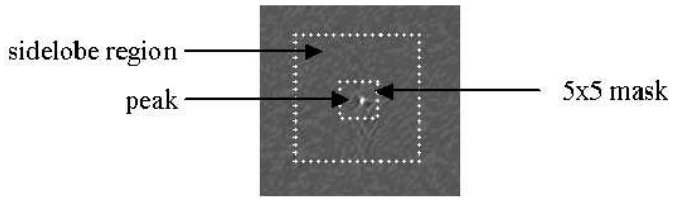

Figura 4.10: Descrizione delle regioni del piano di correlazione per il calcolo del peak-to-sidelobe-ratio

ci siano solo valori di correlazione bassi. Il filtro di correlazione Minimum Average Correlation Energy (MACE) cerca di rendere il valore controllato di correlazione (quello all'origine per immagini di training centrate autentiche) più alto possibile rendendo l'energia media della correlazione d'uscita il più basso possibile. La correlazione in uscita ad un filtro MACE ben progettato tipicamente mostra dei netti picchi per immagine autentiche, rendendo il processo di localizzazione del picco semplice e robusto.

Verrà di seguito spiegato nel dettaglio come è progettato un filtro MACE. Si supponga di avere N immagini di training di taglia $d \times d$. Per prima cosa vengono calcolare le trasformate di Fourier 2D e i risultanti array complessi sono vettorizzati in colonne di $d^2 \times N$ complessi ottenendo la matrice X. Viene oltretutto usata una matrice $d \times d$ diagonale D nella cui diagonale sono presenti i valori medi di potenza dello spettro delle N immagini di training. Dal momento che D è diagonale possiamo salvare solo la diagonale e non la matrice completa. Il filtro è rappresentato da un vettore colonna h di d^2 elementi. Infine, il filtro h deve produrre dei valori u_i pre-specificati di correlazione all'origine in risposta alle N immagini di training. Questo vincolo può essere così scritto:

$$X^+ h = u \tag{4.6}$$

con $u = [u_1 u_2 \dots u_N]^T$ dove T significa traspospo e $+$ complesso coniugato. L'energia di correlazione media del piano può essere espressa con il termine quadratico $h^+ Dh$, minimizzando poi questo quadrato in modo da soddisfare il vincolo in (4.6) porta alla seguente soluzione in forma chiusa del filtro MACE:

$$h = D^{-1} X (X^+ D^{-1} X)^{-1} u \tag{4.7}$$

Dal momento che D è una matrice diagonale, per determinare h è richiesta l'inversione della matrice $N \times N$) $(X^+ D^{-1} X)$ dove N è il numero di training images (di solito piccolo).

Per quantificare il picco di correlazione viene usata la "peak-to-sidelobe ratio" (PSR) cos+ definita:

$$PSR = \frac{mean - peak}{std} \tag{4.8}$$

dove *peak* è il valore più grande nel piano di correlazione di uscita e *mean* e *std* sono il valore medio e la deviazione standard della correlazione di uscita in una regione anulare (20×20) centrata nel picco ed escludendo la regione interna (5×5) come mostrato in figura 4.10.

PSR è progettato per dare una misura relativa dell'altezza del picco di correlazione rispetto al background e si è osservato come non sia sensibile alle dimensioni della regione. Uno dei benefici del PSR è che non è sensibile a variazioni costanti di luminosità nelle immagini di input. Per i filtri MACE ben progettati il valore di PSR deve essere alto per immagini autenitche e piccole per immagini di impostori.

Capitolo 5

Introduzione e Descrizione del Progetto

Contents

Dopo aver descritto il funzionamento di un sistema di autenticazione multibiometrico, in questo capitolo verranno descritte in dettaglio le fasi di analisi dei requisiti e di progettazione e infine illustrata l'implementazione del software sviluppato riportando, quando oppurtuno, alcuni pezzi di codice che possano aiutare a compredere meglio il funzionamento.

Nel primo paragrafo introduttivo verrà presentato il progetto, le motivazioni per le quali il progetto è nato, i vincoli di progettazione che ci siamo posti e verrà data una panoramica generale delle funzioni e delle tecnologie che dovranno essere utilizzate per poter soddisfare i vincoli progettuali. Nella seconda parte, invece, verranno analizzati in dettaglio gli strumenti software implementati, cercando di focalizzarci principalmente sulle modalità di implementazione e sulle scelte implementative che sono state fatte.

5.1 Introduzione

Il progetto nasce dalla neccessità di avere un tool completo di strumenti per la creazione flessibile di architetture per l'autenticazione multibiometrica e per il deployment e l'esecuzione delle stesse. In particolare la tesi ha l'obiettivo di progettare e implementare i seguenti strumenti software:

- Uno strumento software che permetta ad un progettista la possibilità di modellare in modo flessibile una architettura di autenticazione biometrica;
- Uno strumento software che permetta al reparto tecnico il deployment dell'archiettura da remoto;
- Uno strumento software che esegua l'architettura in modo sicuro e distribuito;
- Degli strumenti di enrollment che permettano la registrazione di nuovi utenti nel database dell'organizzazione e che quindi implementino alcuni algoritmi per l'estrazione e la verifica di feature biometriche.

Il progetto può essere collocato nell'area di ricerca che in letteratura viene chiamata "Identity Management"[1] Nel nostro caso abbiamo utilizzato la biometria come uno strumento, infatti, per la memorizzazione e la verifica dell'indentità dell'utente vengono utilizzati complessi algoritmi biometrici.

La progettazione e l'implementazione del software ha seguito un approcio molto modulare. Tale approcio ha come conseguenza una maggiore riutilizzabilità del codice e la possibilità di aggiungere o togliere funzionalità agli strumenti sviluppati in manierà più veloce e semplice. In particolare, l'implementazione deli algoritmi di riconoscimento che vengono usati nella maggioranza del software sviluppato, sono delle classi esterne facilmente collegabili e inseribili in altri progetti simili. In questa ottica abbiamo pensato di rendere disponibile una dettagliata documentazione utilizzando strumenti di documentazione appositi (vedi appendice D).

I requisiti che il progetto deve soddisfare sono i seguenti:

- Possibilità di creare delle architetture "fuzzy" in cui il valore di uscita non sia solo si/no ma possa essere si/no/forse/forse-si/forse-no;
- Possibilità di avere variare una soglia di sicurezza in modo da permettere delle autenticazioni più restrittive o in alternativa delle autenticazioni più permissive;

[1] Con Identity Management (IM), si intendono sistemi integrati di tecnologie, criteri e procedure in grado di consentire alle organizzazioni di facilitare, e al tempo stesso controllare, gli accessi degli utenti ad applicazioni e dati critici, proteggendo contestualmente i dati personali da accessi non autorizzati.

- Un autenticazione e uno scambio di informazioni sicuro basata su RSA;

Nel resto dei paragrafi introduttivi verranno presentati requisiti e obiettivi specifici dei tool sviluppati. Si ricorda che nell'appendice C vengono elencati i requisiti software che servono per la compilazione del codice e l'esecuzione dello stesso e alcuni errori che si sono visti essere riccorrenti quando si và ad compilare, installare ed eseguire il codice con la rispettiva soluzione trovata.

5.1.1 Face Enrollment

Questo strumento ha lo scopo di fornire una interfaccia grafica con cui l'utente[2] possa inserire e registrare facilmente un nuovo utente nel database dell'organizzazione e dare a tale utente i diritti di cui necessità.

Lo strumento di face enrollment deve quindi soddisfare i seguenti requisiti:

- Essere facile da usare e sicuro;
- Implementare funzionalità efficienti in termini di tempo di esecuzione e spazio utilizzato;
- Gestione multiutenza;
- Verificare la robustezza delle feature estratte.

ed è finalizzato per i seguenti obiettivi:

- Inserire, eliminare e modificare l'anagrafica di un nuovo utente o di un utente esistente;
- Inserire, eliminare e modificare una o più feature facciali estratte per ogni utente;
- Verificare direttamente le feature facciali estratte con un tool di autenticazione "live" in cui si possa verificare la robustezza delle stesse modificando nel caso il livello di sicurezza dell'autenticazione. La necessità di questo strumento è nata dal momento che gli algoritmi di face recognition sono sensisbili alla luce, di conseguenza è neccessario estrarre più feature facciali con diverse intensità luminose. Con questo tool è possibilile verificare e garantire l'autenticazione dell'utente in più condizioni di luminosità.

[2]In questo caso ci riferiamo all'utente che usa questo specifico strumento, sia esso l'amministratore di sistema, un tecnico o qualunque altra persona con i diritti di accesso ed esecuzione di questo software

5.1.2 Deployment

Questo strumento ha il compito di facilitare e rendere automatico il processo di deployment dell'architettura, creata con gli strumenti spiegati in [86], nel sistema e, essendo l'architettura eseguita in un ambiente distribuito, configurare tutti i paramentri di connessione all'interno della LAN o di Internet.

Lo strumento di deployment deve quindi soddisfare i seguenti requisiti:

- Essere facile da usare;
- permettere una rapida configurazione dell'architettura;
- garantire la sicurezza dei dati.

ed è finalizzato per i seguenti obiettivi:

- rendere facile e automatico il processo di deployment;
- configuare il modo semplice tutti i parametri di connessione (indirizzi IP, porte, dati di accesso alle macchine remote, ecc.);
- creare degli script di compilazione ed esecuzione remota dell'architettura.

5.1.3 Esecuzione

Questo strumento ha il compito di permettere un'esecuzione distribuita e sicura dell'architettura di autenticazione multibometrica creata con gli strumenti spiegati in [86].

Lo strumento di deployment deve quindi soddisfare i seguenti requisiti:

- Essere user-friendly;
- garantire la sicurezza dei dati inviati e ricevuti;
- essere facilmente espandibile con ulteriori algoritmi di riconiscimento biometrico.

ed è finalizzato per i seguenti obiettivi:

- creare connessioni remote su LAN o WAN tramite protocollo TCP/IP;
- permettere più connessioni contemporanee;
- garantire la concorrenza delle connessioni e l'integrità dei dati;
- implementare strumenti avanzati per connessioni SSL.

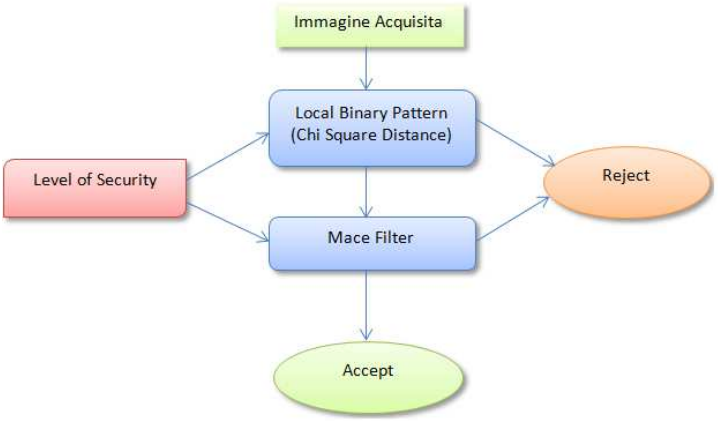

Figura 5.1: Diagramma di autenticazione tramite combinazione degli algoritmi LBP e MACE

5.2 Descrizione e Analisi dell'Implementazione

In questo paragrafo verrà descritta in dettaglio l'implemetazione degli strumenti sviluppati. Nel primo sottoparagrafo verrà trattata l'implemtazione degli algoritmi di face recognition, comuni sia al tool di enrollment che a al software per l'esecuzione dell'architettura, mentre nei restanti paragrafi verrà descritta l'implementazione dello strumento di deployment ed esecuzione dell'architettura.

5.2.1 Algoritmi di Rilevazione e Riconoscimento Facciale

L'implementazione del software di rilevazione e riconoscimento facciale è avvenuto utilizzando l'algoritmo di Viola et al. per la rilevazione e una combinazione degli algoritmi di Local Binary Pattern e Minimum Average Correlation Energy. Come descritto in [37] si è deciso di combinare LBP con MACE dato che LBP riesce a tollereare adeguatamente variazioni di luminosità mentre MACE da bassi valori di FAR e FRR. In particolare i due algoritmi sono stati combinati, come si vede in figura 5.1 in modo che l'immagine acquisita sia prima processata con LBP e poi con MACE. Le soglie di autenticazioni sono variabili e vengono fissate dinamicamente in base al livello di sicurezza fissato che viene passato come input agli algoritmi di autenticazione. In [37] sono state fatte delle misurazioni con il database ORL di Cambrige in modo da verificare in modo sperimentale l'effettivo rate di autenticazione fornito dalla combinazione dei due algoritmi e in modo da confrontarlo con le prestazioni degli stessi algoritmi utilizzati singolarmente.

Haar Like Feature

Come discusso nel paragrafo relativo alla letteratura la rilevazione della faccia è stato fatto usando le librerie openCV riportate in appendice A basato sull'algoritmo di Viola at al. [17]. In realtà l'implementazione attuale è un'estensione dell'algortimo originale fatta da Lienhart et al. [39]. In openCV sono inoltre contenuti i "trained cascade files" in formato "xml" risultato della fase di training del classificatore fatta da AdaBoost e per permettere la classificazione in cascata. In openCV sono disponibili i file di training per la rilevazione di:

- Faccia per soggetto senza occhiali disponibile nel file "haarcascade.xml";
- Occhi per soggetto con occhiali disponibile nel file "haarcascade_eye_tree_eyeglasses.xml";
- Occhi per soggetto senza occhiali disponibile nel file "haarcascade_eye.xml";
- Naso disponibile nel file "haarcascade_nose.xml"

Data la particolare efficienza con cui è stato implementato l'algoritmo di rilevazione facciale in openCV è possibile la rilevazione facciale in circa $30-45$ ms in immagini di dimensione 320×240.

Il primo passo è stato quindi quello di rilevare, tramite una webcam, un flusso video dove fosse presente la faccia dell'utente da autenticare. Questo può essere fatto facilmente con le librerie openCV inizializzando la webcam in questo modo:

```
1 CvCapture* capture =cvCaptureFromCAM(0);
```

e acquisendo un'immagine dal flusso video ogni 40 ms tramite il seguente codice:

```
1 IplImage* frame = cvQueryFrame(capture);
```

Il frame acquisito, visibile in figura 5.2, ora dovrà essere processato dall'algoritmo Haar like in modo da poter rilevare faccia, occhi e naso dall'immagine. Il primo passo è stato quello di caricare i file di training in formato "xml" tramite il seguente codice:

```
1 CvHaarClassifierCascade* cascade = (
    CvHaarClassifierCascade*)cvLoad(HAAR_CASCADE_FACE, 0,
     0, 0 );
2 CvHaarClassifierCascade* cascade = (
    CvHaarClassifierCascade*)cvLoad(HAAR_CASCADE_EYE, 0,
    0, 0 );
```

Figura 5.2: Frame acquisito tramite webcam

Figura 5.3: Frame acquisito convertito in scala di grigi

```
3 CvHaarClassifierCascade* cascade = (
    CvHaarClassifierCascade*)cvLoad(
    HAAR_CASCADE_EYE_GLASSES, 0, 0, 0 );
4 CvHaarClassifierCascade* cascade = (
    CvHaarClassifierCascade*)cvLoad(HAAR_CASCADE_NOSE, 0,
     0, 0 );
```

dove il primo parametro della funzione "cvLoad" è stato inizializzato con il percorso corrente ai file di training specificati precedentemente.

Il frame acquisito viene poi convertito in toni di grigio utilizzando la funzione "cvCvtColor" in questo modo:

```
1 cvCvtColor( input, output, CV_BGR2GRAY );
```

l'immagine in uscita si può vedere in figura 5.3

A questo punto l'immagine acquisita è pronta per essere passata come input all'algoritmo di rilevazione facciale di Viola et al. utilizzando semplicemente la funzione "cvHaarDetectObjects" fornita da openCV in questo modo:

Figura 5.4: Le parti dell'immagine rilevate dall'algoritmo Haar like

```
1 CvSeq* faces = cvHaarDetectObjects(img, cascade,
    storage, 1.4, 2, 0 |CV_HAAR_DO_CANNY_PRUNING, cvSize
    (80/scale, 80/scale));
```

Nell'immagine iniziale sono stati rilevati faccia, naso e occhi, visibile in figura 5.4. Possiamo quindi tagliare l'immagine in modo da eliminare le parti che non contengono informazione e tenere soltanto le parti dell'immagine appena rilevate.

Local Binary Path

Viene di seguito descritta l'implementazione dell'algoritmo di Local Binary Path. Ricordiamo che tale algoritmo, per ogni pixel dell'immagine, deve confrontare il suo valore di intensità con il valore di intensità dei suoi otto vicini in modo da calcolare una 8-bit codeword leggendo i bit di confronto in senso orario partendo dal pixel in alto a sinistra. Tale codeword deve poi essere tranformata in decimale e il risultante valore è il valore di intensità dell'immagine LBP di uscita. Tale algortimo è stato così implementato:

```
1  CvScalar s, s1;
2  //Inizialization
3  int p1x,p2x,p3x,p4x,p5x,p6x,p7x,p8x;
4  int p1y,p2y,p3y,p4y,p5y,p6y,p7y,p8y;
5
6  //Neighbors settings
7  p1x=j-1; p1y=i-1;
8  p2x=j; p2y=i-1;
9  p3x=j+1; p3y=i-1;
10 p4x=j+1; p4y=i;
11 p5x=j+1; p5y=i+1;
12 p6x=j; p6y=i+1;
13 p7x=j-1; p7y=i+1;
14 p8x=j-1; p8y=i;
15 s=cvGet2D(img,i,j);
16
17 //Comparison and decimal conversion
18 double bit1=128*getBIT(img,p1x,p1y,s.val[0]);
19 double bit2=64*getBIT(img,p2x,p2y,s.val[0]);
```

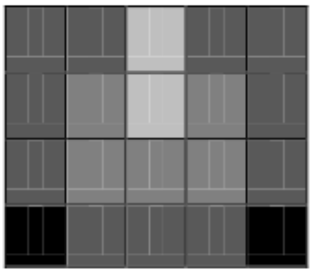

Figura 5.5: Pesi associati alle aree dell'immagine LBP per il calcolo della distanza Chi square. Le aree più chiare hanno peso maggiore.

```
20 double bit3=32*getBIT(img,p3x,p3y,s.val[0]);
21 double bit4=16*getBIT(img,p4x,p4y,s.val[0]);
22 double bit5=8*getBIT(img,p5x,p5y,s.val[0]);
23 double bit6=4*getBIT(img,p6x,p6y,s.val[0]);
24 double bit7=2*getBIT(img,p7x,p7y,s.val[0]);
25 double bit8=1*getBIT(img,p8x,p8y,s.val[0]);
26 s1.val[0]=bit1+bit2+bit3+bit4+bit5+bit6+bit7+bit8;
27 cvSet2D(imgLBP,i,j,s1);
```

Come si può vedere l'implementazione di LBP è molto semplice e veloce con complessità di tempo e di lineare. Nelle righe $7-15$ vengono settati i vicini del pixel corrente. Nelle righe $18-25$ vengono confrontati i valori di intensità e il risultato viene convertito in decimale. Nelle righe $26-27$ viene calcolato il valore finale decimale del pixel e settato nell'immagine finale.

Per quanto riguarda l'autenticazione si è pensato di salvare su disco l'immagine LBP come matrice con la seguente funzione "cvWrite" usata in questo modo:

```
1 cvWrite( fs, "lbp", LBPMatrix, cvAttrList(0,0))
```

Nel momento in cui l'utente si siede di fronte alla webcam per essere autenticato l'immagine acquisita subirà lo stesso processo appena descritto e dopo aver calcolato l'immagine LBP dall'originale viene calcolata la dissimilarità fra le due immagini LBP con la distanza Chi square. Vengono attribuiti dei pesi alle aree delle immagini (si veda fig 5.5) in modo da dar più importanza ad aree come occhi, naso bocca e meno importanza ad aree esterne dell'immagine. L'implementazione è stata così fatta:

```
1 double LBPdiff(CvMat* model,CvMat* test)
2 {
3     double weights[4][5] =
```

```
4       {
5           { 1, 1,  3, 1,  1},
6           { 1, 2,  3,2,  1},
7           { 1, 2,  2, 2,  1},
8           { .3, 1,  1, 1,  .3},
9       };
10      int i,j,k;
11      double chiSquare=0;
12      for (i=0;i<5;i++) \\per ognuna delle celle
            orrizontali
13      {
14          for (j=0;j<4;j++) \\per ognuna delle celle
                verticali
15          {
16              for (k=0;k<59;k++) \\per ogni pixel all'
                    interno della cella
17              {
18                  CvScalar s1,s2;
19                  s1=cvGet2D(model,i*4*59 + j*59 +k,0);
20                  s2=cvGet2D(test,i*4*59 + j*59 +k,0);
21                  double hist1=0,hist2=0;
22                  hist1=s1.val[0];
23                  hist2=s2.val[0];
24                  if ((hist1+hist2)!=0)
25                  chiSquare+=(weights[j][i]*(pow(hist1-
                        hist2,2)/(hist1+hist2)));
26              }
27          }
28      }
29      return chiSquare;
30  }
```

Nella prima parte del codice, da riga 3 ad 11 vengono inizializzate le variabili e i pesi delle aree dell'immagine. Poi nelle successive righe $(12-28)$ i tre cicli for innestati, per ogni regione dell'immagine, applicano la formula per calcolare la distanza Chi square riportata nel paragrafo 4.2.1. Alla fine viene ritornato il valore finale di Chi square dato dalla somma di tutte i risultati parziali.

Un utente viene autenticato da LBP se il valore della variabile Chi square è minore di una soglia calcolata dinamicamente in base al livello di sicurezza settato, in particolare la soglia viene così calcolata: $thresholdLBP = maxLbpThresh - ((1 - security) * 200)$ in cui $maxLbpThresh$ è un valore di default che indica la massima dissimilarità (attualmente impostato a 8000), *security* è i livello di sicurezza impostato dal progettista dell'architettura e 200 è un fattore di cambio scala.

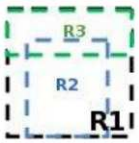

Figura 5.6: Regioni dell'immagine selezionate a cui verrà applicato l'algoritmo MACE per il calcolo del filtro di correlazione

Figura 5.7: Ridimensionamento, equalizzazione della luminosità e aumento del contrasto per le immagini di input ricavate con l'algoritmo Haar like

Minimum Average Correlation Energy

In questo paragrafo verrà descritta l'implementazione dell'algoritmo MACE. La prima operazione che viene fatta è quella di estrarre 13 immagini di training dal flusso video. Tali immagini serviranno per il calcolo del filtro di correlazione h come spiegato nel paragrafo 4.2.2. Prima di applicare la formula chiusa per il calcolo del filtro h vengono fatte alcune operazioni preliminari:

- viene eseguito l'algoritmo Haar like per ognuna di queste immagini in modo da localizzare ed estrarre la faccia, il naso e gli occhi dato che, l'algoritmo MACE, viene eseguito su queste tre regioni dell'immagini in modo da aumentarne l'accuratezza. In figura 5.6 è possibile vedere le tre regioni (R1, R2, R3) selezionate;

- vengono ridimensionate le immagine acquisite in modo da renderle di dimensioni uguali;

- viene euqalizzata l'intensità delle immagini con la funzione $cvEqualizeHist$. Tale funzione ha il compito di equalizzare la luminosità dell'immagine e di aumentarne il contrasto. In figura 5.7 è possibile vedere tale trasformazione con le immagine di input trovate dall'algoritmo Haar like;

- per ogni immagine viene calcolata la trasformata discreta di Fourier con la funzione cvDFT utilizzata in questo modo

```
1 cvDFT(dftImage, dftImage, CV_DXT_FORWARD, 0)
```

dove "CV_DXT_FORWARD" indica la traformata di Fourier diretta non scalata e *"dftImage"* è una una matrice inizializzata in questo modo

```
1 CvMat* dftImage = cvCreateMat\\(d, d, CV_64FC2);
```

dove il primo e il secondo parametro indicano le dimensioni dell'immagine e il terzo parametro "CV_64FC2" indica una matice contenenti valori complessi.

A questo punto dobbiamo andare a inizializzare la matrice X, la matrice X^+ [3] e la matrice D^{-1}. Ricordiamo che la matrice X e X^+ sono matrici $d^2 \times N$ dove d^2 indica il numero totale di pixel di un'immagine $d \times d$ e N indica il numero di immagini di training che nel nostro progetto è settato a 13 come detto precedentemente e contiene, in ogni colonna, l'output della transformata discreta di Fourier su una singola immagine. La matrice D, invece, è una matrice diagonale $d \times d$ (quindi possiamo salvarla in un unico vettore contenente la diagonale della matrice D) e contiene i valori medi di potenza dello spettro delle N immagini di training. Riportiamo di seguito il codice utilizzato per l'inizializzazione delle matrici:

```
1 CvMat * D = cvCreateMat(TOTALPIXEL,1, CV_64FC2 ); //
    Matrice diagonale D
2 CvMat * D = cvCreateMat(TOTALPIXEL,1, CV_64FC2 ); //
    Matrice diagonale D inversa
3 CvMat * X = cvCreateMat(TOTALPIXEL,N, CV_64FC2 ); //
    Matrice X
4 CvMat * XCON = cvCreateMat(N,TOTALPIXEL, CV_64FC2 ); //
    Matrice X trasposta coniugata
```

dove "TOTALPIXEL= $d \times d$".

Giunti a questo punto possiamo cominciare il calcolo della formula $h = D^{-1}X(X^+D^{-1}X)^{-1}u$. Partiamo con il calcolare i sottotermini $D^{-1}X$ e $X^+D^{-1}X$ come mostrato di seguito:

[3] Ricordiamo che in algebra lineare, la matrice trasposta coniugata (o matrice aggiunta) di una matrice X a valori complessi è la matrice ottenuta effettuando la trasposta e scambiando ogni valore con il suo complesso coniugato: $X^+ = (X^T)^* = (X^*)^T$. In termini degli elementi vale la relazione: $X^+_{jk} = X^*_{kj}$ dove j è l'indice di riga e k quello di colonna.

```
int l=0,m=0;
 for (l=0;l<size;l++)
 {
      for (m=0;m<TOTALPIXEL;m++)
      {

           CvScalar s1= cvGet2D(DINV,m,0);
           CvScalar s2= cvGet2D(XCON,l,m);
           CvScalar s3= cvGet2D(X,m,l);
           s2.val[0]*=s1.val[0];
           s2.val[1]*=s1.val[0];
           s3.val[0]*=s1.val[0];
           s3.val[1]*=s1.val[0];
           cvSet2D(XCON_DINV, l, m,s2);
           cvSet2D(DINV_X, m, l,s3);
      }
 }
 cvMatMul(XCON_DINV,X,XCON_DINV_X);
```

dove "DINV_X = $D^{-1}X$", "XCON_DINV_X= $X^{+}D^{-1}X$" e "XCON_DINV= $X^{+}D^{-1}$".

Andiamo poi a calcolare $(X^{+}D^{-1}X)^{-1}$ con la funzione cvInvert usata in questo modo:

```
cvInvert(XCON_DINV_X,XCON_DINV_X_INV);
```

Infine possiamo calcolare il valore del filtro di correlazione andando a moltlipicare "XCON_DINV_X_INV" con "DINV_X" e moltiplicando il vettore risultante con il vettore u contenente i valori pre-specificati di correlazione. Riportiamo di seguito il codice utilizzato per lo scopo:

```
 cvMatMul(DINV_S,SPLUS_DINV_S_INV,Hmace);
 for (l=0;l<size;l++)
 {
      CvScalar s3;
      s3.val[0]=1;
      s3.val[1]=0;
      cvSet2D(Uvalue, l, 0,s3);
 }
 cvMatMul(Hmace,Uvalue,Hmace_FIN);
 CvMat * maceFilterVisualize = cvCreateMat(
     SIZE_OF_IMAGE_2X, SIZE_OF_IMAGE_2X, CV_64FC2 );
 for (l=0;l<SIZE_OF_IMAGE_2X;l++)
 {
      for (m=0;m<SIZE_OF_IMAGE_2X;m++)
      {
```

```
15                CvScalar s1= cvGet2D(Hmace_FIN,(l*
                      SIZE_OF_IMAGE_2X +m),0);
16                cvSet2D(maceFilterVisualize, l, m,s1);
17            }
18
19        }
20        CvScalar s1= cvGet2D(Hmace_FIN,0,0);
21        return maceFilterVisualize;
```

Nella riga 1 andiamo a calcolare $D^{-1}X(X^{+}D^{-1}X)^{-1}$, nelle righe $3-8$ andiamo a settare ad 1 i valori del vettore u mentre nella riga 9 andiamo a calcolare il valore finale del filtro h contenuto nella matrice "maceFilterVisualize".

Il valore del filtro viene calcolato per ognuna delle tre regioni dell'immagini specificate in precedenza e salvato su disco con la funzione "cvWrite" usata in questo modo:

```
1 cvWrite( fs, "maceFilter", maceFilter->filter,
      cvAttrList(0,0));
```

A questo punto il filtro di correlazione h è stato calcolato, possiamo calcolare, per ogni immagine di training acquisita, il "peak-to-side-loberatio" (PSLR) che ci servirà più avanti per poterlo confrontare con il PSLR delle immagini di autenticazione acquisite e poter autenticare l'utente confrontando tale valore con la soglia di sicurezza impostata dal progettista dell'architettura. Di seguito viene quindi descritta l'implementazione per il calcolo del PSLR. Come sempre prima le immagini vengono passate in scala di grigi, poi ridimensionate e infine equalizzate con le funzioni viste precedentemente. A questo punto viene calcolata la DFT delle immagini e la correlazione viene calcolata moltlipicando il filtro di correlazione h calcolato in precedenza con la trasformata discreta di Fourier (figura 5.2) appena calcolta usando la funzione cvMulSpectrums usata in questo modo:

```
1 cvMulSpectrums(dftImage , maceFilterVisualize, dftImage,
      CV_DXT_MUL_CONJ);
```

dove il primo parametro è la DFT dell'immagine in ingresso, il secondo paramentro è il filtro di correlazione h, il terzo parametro è la variabile di uscita e "CV_DXT_MUL_CONJ" indica il coniugato del secondo argomento. A questo punto come spiegato nel paragrafo 4.2.2 dobbiamo andare a localizzare il picco di correlazione e a rilevare il suo valore. Per fare questo prima dobbiamo scomporre il risultato dell'operazione precedente in parte reale e immaginaria con la funzione "cvSplit" e poi eseguire la funzione "cvMin-

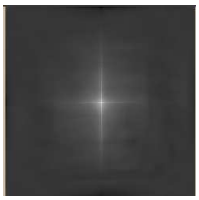

Figura 5.8: Piano di correlazione relativo ad un utente in cui l'autenticazione è andata a buon fine

MaxLoc"[4] che ci ritornarà il picco di correlazione e la sua posizione. Di seguito viene riportato il codice utilizzato per le operazioni descritte:

```
1 cvSplit(dftImage, image_Re, image_Im, 0, 0);
2 double m1,M1;
3 CvPoint p1,p2;
4 cvMinMaxLoc(image_Re, &m1, &M1, &p1, &p2, NULL);
```

In figura 5.8 viene mostrato il piano di correlazione contenuto nell'immagine "image_Re" realtivo ad un utente in cui l'autenticazione è andata a buon fine. Si può notare al centro un punto bianco che indica il picco di correlazione, come descritto nel paragrafo 4.2.2 relativo all'algoritmo MACE e come mostrato in figura 4.10.

Infine dobbiamo calcolare il valore del PSLR utilizzando la formula 4.8 in una regione anulare 20×20 centrata nel picco ed escludendo la regione interna 5×5 con il seguente codice:

```
1     int rad1=int(floor((double)(45.0/64.0)*(double)
          SIZE_OF_IMAGE));
2     int rad2=int(floor((double)(27.0/64.0)*(double)
          SIZE_OF_IMAGE));
3     int l=0,m=0;
4     double value=0;
5     double num=0;
6     for (l=0;l<SIZE_OF_IMAGE_2X;l++)
7     {
8         for (m=0;m<SIZE_OF_IMAGE_2X;m++)
9         {
10            double rad=sqrt((pow(m-SIZE_OF_IMAGE,2)+pow(
                  l-SIZE_OF_IMAGE,2)));
11            if (rad<rad1)
12            {
```

[4] la funzione cvMinMaxLoc trova il minimo e il massimo tra gli elementi di un array e la loro posizione.

```
13                  if (rad>rad2)
14                  {
15                      CvScalar s1= cvGet2D(image_Re,l,m);
16                      value+=s1.val[0];
17                      num++;
18                  }}}}
19      value=value/num;
20      double std2=0;
21      for (l=0;l<SIZE_OF_IMAGE_2X;l++)
22      {
23          for (m=0;m<SIZE_OF_IMAGE_2X;m++)
24          {
25              double rad=sqrt((pow(m-SIZE_OF_IMAGE,2)+pow(
                    l-SIZE_OF_IMAGE,2)));
26              if (rad<rad1)
27              {
28                  if (rad>rad2)
29                  {
30                      CvScalar s1= cvGet2D(image_Re,l,m);
31                      std2+=(pow(value-s1.val[0],2));
32                  }}}}
```

Nel momento in cui l'utente si deve autenticare le immagini acquisite dalla webcam subiranno lo stesso processo appena descritto. Alla fine quindi quello che ci servirà per l'autenticazione è il valore del PSLR dell'immagine acquisita. Un utente verrà autenticato se il valore del PSLR totale cioè al PSR dato dalla somma dei PSR parziali di faccia, bocca e occhi dell'immagine diviso il valore del PSLR dell'immagine di training originaria moltiplicato 100 (stiamo cioè calcolando una percentuale) è maggiore della soglia impostata dal progettista dell'architettura più un certo valore di step che attualmente è pari ad un quarto di tale soglia. Per precisione viene di seguito mostrato il codice che implementa quanto appena descritto:

```
1 int pcent=int(((double)value/(double)PSLR)*100);
2 int upperPcent=int((security+((1-security)/4))*100.0);
3 if (pcent>=upperPcent)
4 {
5 return true; //Utente autenticato
6 }
```

5.2.2 Base di Dati

A supporto dei tool decritti in precedenza è stata progettata una semplice basi di dati su database MySql. Di seguito verranno descritte le tabelle implementate.

La prima tabella che è stata creata è la tabella dove vengono memorizzate le informazioni relativa all'anagrafica degli utenti al momento dell'enrollment. Tale tabella è stata nominata "*users*" ed è così specificata:

Field	Type	Null	Key	Default	Extra
idusers	int(11)	NO	PRI	NULL	auto_increment
FirstName	varchar(45)	NO		NULL	
LastName	varchar(45)	NO		NULL	
Birth	date	NO		NULL	
Password	varchar(45)	NO		NULL	
Date	date	NO		NULL	
cellularPhone	varchar(45)	NO		NULL	
email	varchar(45)	NO		NULL	

La seconda tabella creata è a supporto dell'autenticazione. In questa tabella nominata *"authList"* vengono salvate, da parte del server, tutte le autenticazioni fatte dagli utenti con rispettiva data e ora di autenticazione, risultato dell'autenticazione con un flag che indica se l'autenticazione non è stat usata come input nei successivi livelli dell'architettura ($authFlag = 0$) oppure se è stato usato ($authFlag = 1$).

Field	Type	Null	Key	Default	Extra
idAuthList	int(11)	NO	PRI	NULL	auto_increment
type	varchar(40)	NO		NULL	
input	varchar(40)	NO		NULL	
date	date	NO		NULL	
time	time	NO		NULL	
user	varchar(40)	NO		NULL	REF. users(idusers)
output	varchar(40)	NO		NULL	
authFlag	int(11)	NO		NULL	

5.2.3 Face Enrollment

Come detto in precedenza questo tool ha il compito di fornire una interfaccia grafica con cui l'amministratore di sistema o un suo delegato possa inserire e registrare facilmente un nuovo utente nel database dell'organizzazione e dare a tale utente i diritti di cui necessità. Di seguito vengono mostrati due screeshot del tool, il primo in figura 5.9 che mostra la prima schermata relativa alla selezione di un utente e alla gestione del database e e la seconda 5.10 relativa al training dell'utente e alla gestione delle precedenti immagini con cui il training è già stato effettuato.

Vereà di seguito descritto in dettaglio il funzionamento e l'implementazione delle due schermate.

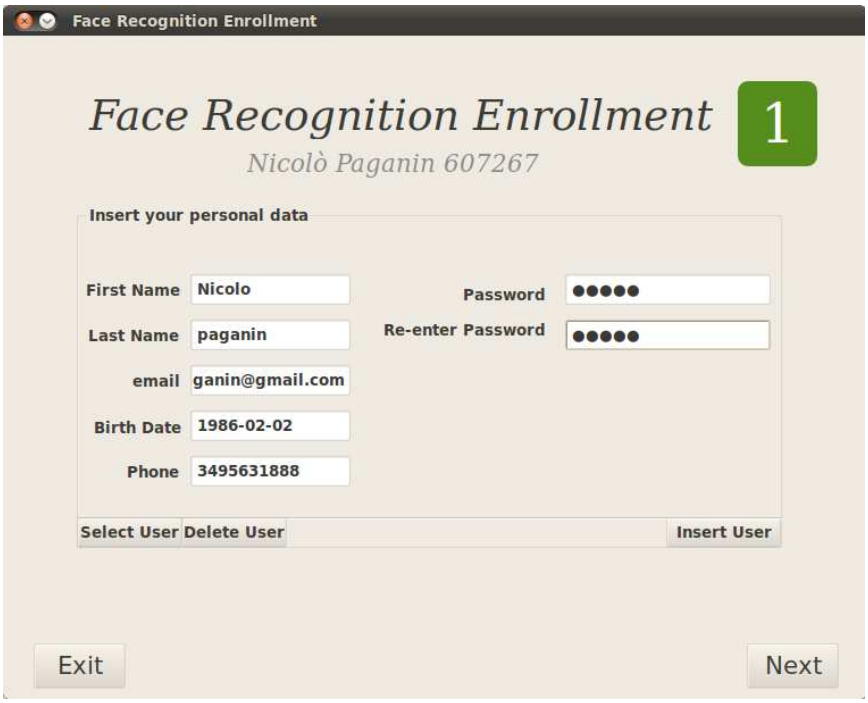

Figura 5.9: Prima schermata del tool di training di immagini facciali relativa alla selezione di un utente e alla gestione del database

Prima Schermata

La prima schermata ha lo scopo di permettere a chi deve effettuare l'enrollment degli utenti di:

- poter selezionare un utente già presente nel db, cliccando sul punsalte "Select User" comparirà un box, come mostrato in figura 5.11, dove l'utente può essere selezionato. In questo modo tutti i campi della prima schermata verranno automaticamente compilati;

- poter gestire il database, cliccando sul pulsante "Delete User" l'utente selezionato verrà cancellato dal database;

- poter inserire un nuovo utente, inserendo tutti i dati nei campi e cliccando sul pulsante "Insert User".

  A questo punto è possibile cliccare sul pulsante "Next". Se tutti i campi sono compilati correttamente e se l'utente esiste nel database comparirà la seconda schermata relativa al training delle immagini facciali.

Figura 5.10: Seconda schermata del tool di training di immagini facciali relativa al training dell'utente e alla gestione delle precedenti immagini con cui il training è già stato effettuato

Seconda Schermata

La seconda schermata ha lo scopo di effettuare il training di immagini facciali. Come si può vedere nell'immagine 5.9 a sinistra compare un box al cui interno vengono mostrate le immagini acquisite con la webcam. Tali immagini vengono date come input all'algoritmo Haar like che cercherà di localizzare la faccia dell'utente da registrare nel database. Nella parte sottostante verranno anche mostrati dei commenti relativi alla posizione che l'utente deve tenere cioè se si deve allontanare o avvicinare alla webcam o se la localizzazione è stata fatta nel modo corretto.

A questo punto se la localizzazione viene eseguita è possibile cliccare sul pulsante "Capture". Verrano estratte dal flusso video della webcam 13 immagini e saranno date come imput agli algoritmi di LBP e MACE in modo da poter salvare su disco l'immagine LBP e il filtro di correlazione h con le ripettive soglie. A tal proposito bisogna dire che il filtro di correlazione e l'immagine lbp vengono salvate nella home del pc attualmente in uso. Nella home verà creata una cartella nominata "mac" al cui interno saranno presenti 2 cartelle:

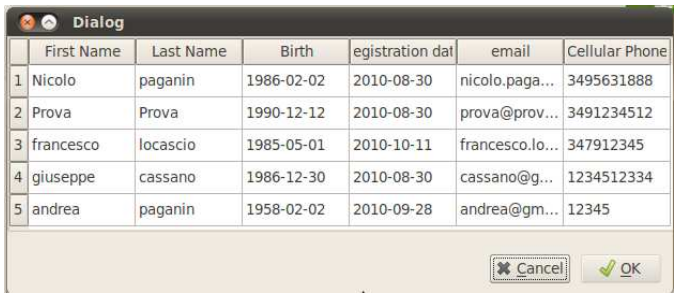

Figura 5.11: Box per la selezione di un utente già presente nel database

- una cartella nominata faces, al cui interno sono memorizzati tutti i set di 13 immagini estratte. Le cartelle contenenti tali set saranno nominate con un nome univoco così formato: (idUtente)-(anno)(mese)(giorno)(ora)(millisecondi)(microsecondi) (es. $74 - 11010513282123238505$).
- una cartella nominata model, al cui interno sono presenti diversi file ".xml" che congono l'immagine LBP e il filtro di correlazione h. In particolare per ogni set di 13 immagini estratte verranno creati 4 file:
  1. (nome_univoco)_eye_mace.xml in cui viene salvato il filtro di correlazione h per gli occhi;
  2. (nome_univoco)_face_mace.xml in cui viene salvato il filtro di correlazione h per la faccia;
  3. (nome_univoco)_inside_face_mace.xml in cui viene salvato il filtro di correlazione h per la bocca;
  4. (nome_univoco)_face_lbp.xml in cui viene salvata l'immagine LBP sottoforma di matrice.

Per ridurre al minimo la possibilità di non-autenticazione dovuta a differenti intensità di illuminazione si consiglia di realizzare più training set con differente luminosità dell'ambiente. Il tool mette a disposizione uno strumento per verificare sul momento la robustezza dei training set acquisiti e la soglia di sicurezza minima di autenticazione. Cliccando su "Advanced Settings" compare il box visualizzato in figura 5.12. Come si vede da tale figura è possibile modificare manualmente la solgia di sicurezza e verificare istantaneamente se l'autenticazione ha esito positivo oppure no. In questo modo si può decidere di creare ulteriori training set o di rimuovere quelli che non comportanto un miglioramento dell'accuratezza.

5.2.4 Deployment

In questo paragrafo verrà descritta l'implementazione del tool di deployment dell'architettura. Dato che l'architettura di autenticazione biometrica

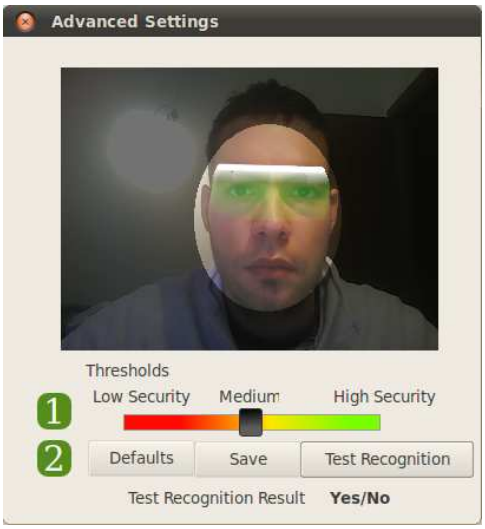

Figura 5.12: Box per verificare la robustezza dei trining set acquisiti e per verificare l'autenticazione con differenti livelli di sicurezza

è distribuita, a questo tool spetta il compito di dare, all'amministratore di sistema o un suo delegato, uno strumento facile e immediato per la configurazione dei client in cui andranno a essere eseguiti i blocchi di input specificati nell'architettura e per la configurazione del server. In figura 5.13 e 5.14 vengono mostrati degli screenshot di tale tool. In particolare la figura 5.13 è relativa alla prima schermata di configurazione degli input e la figura 5.14 è relativa alla configurazione di client e server.

Di seguito andremo a descrivere l'implementazione del tool. Nel primo sottoparagrafo verrà descritto il procedimento di configurazione dell'architettura in modo che possano essere settati prima i client in cui dovranno

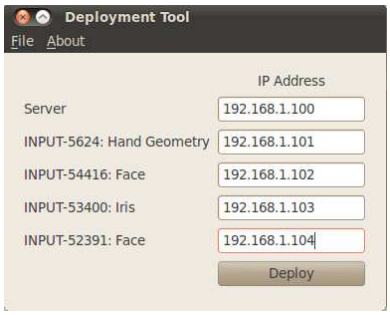

Figura 5.13: Screenshot del tool di deployment, schermata 1 relativa alla configurazione degli input

Figura 5.14: Screenshot del tool di deployment, schermata 2 relativa alla configurazione di client e server

essere eseguiti uno o più autenticazioni e in seguito alcuni parametri per l'accesso remoto a tali macchine come indirizzo IP, username e password di un account con privilegi root.

Configurazione dell'architettura

Il primo passo che si deve fare è quello di aprire un file ".xml" di architettura generato con la suite a supporto della progettazione in [86]. Per una spiegazione dettagliata di come è composto tale file fare riferimento allì'appendice E. A questo punto il tool di deployment processerà tale file e selezionerà tutti gli input presenti (corrispondenti ai blocchi di autenticazione biometrica) e li mostrerà come in figura 5.13. All'utente spetta il compito di assegnare ad ogni input una macchina client su cui verrà eseguita l'autenticazione, fornendo l'indirizzo IP di tale macchina oltre a quello del server che compare in testa alla lista.

Quando a tutti gli input è stata assegnata una macchina in cui essere eseguiti si può premere il pulsante deploy. A questo punto verrà visualizzata la schermata come quella mostrata in figura 5.14 che chiederà di specificare, per ogni indirizzo IP impostato nella schermata precedente username e password per l'accesso remoto ad un account con privilegi root di questa macchina in modo da poter installare i client o il server. Cliccando sul pulsante "Create Script" verranno generati i seguenti file:

- All'interno della cartella Script:
    1. un file nominato "deployment_script" contenente lo script di installazione remota;
    2. un file nominato "start_script" contentente uno script di avvio dell'architettura.

  Nel seguito verrà spiegato nel dettaglio il funzionamento di tali script.

- All'inteno della cartella "resources":

  1. un file, per ogni client univoco specificato, nominato "input.list.(Indirizzo_IP_inserito)" contenente la lista degli input che spettano al client installato all'indirizzo IP specificato e una serie di altre informazioni quali: il nome dell'input, in numero di output di tale input, il tipo di tale input e il livello di sicurezza associato in questo formato "`Tipo:Nome:#Output: :Sicurezza`" (es."`Face:INPUT-54416:2:45`").
  2. un file nominato "server.ip" contenente nella prima riga indirizzo IP del server e nella seconda riga il numero di porta in cui il server si porta in ascolto.

- Oltre a questi file, all'interno della cartella "resources" dovranno essere presenti i seguenti altri file (creati manualmente dall'utente che sta effettuando il deployment:

  1. Un file nominato "haarcascades.tar.bz2" con all'interno una cartella contenente i file di training per l'algoritmo Haar like elencati nel sottoparagrafo 5.2.3;
  2. Un file nominato "client.tar.bz2" con all'interno la cartella con i sorgenti del client compressi in formato Bzip2;
  3. Un file nominato "client" con all'interno il seguente script bash:

```
1 #!/bin/bash
2
3 cd ~
4 tar xjvf haarcascades.tar.bz2
5 tar xjvf client.tar.bz2
6 cd client
7 qmake
8 make
9 ./client -display :0
```

     che ha il compito di estrarre i file di training e la cartella con i sorgenti del client, compilarli localmente ed eseguirli;
  4. un file nominato "client_start" con all'interno il seguente codice bash:

```
1 #!/bin/bash
2
3 cd ~
4 cd client
5 ./client -display :0
```

per l'avvio del client;

5. Un file nominato "server.tar.bz2" con all'interno la cartella con i sorgenti del client compressi in formato Bzip2;
6. Un file nominato "server" con all'interno il seguente script bash:

```
#!/bin/bash

cd ~
tar xjvf haarcascades.tar.bz2
tar xjvf server.tar.bz2
cd server
qmake
make
./server -display :0
```

che ha il compito di estrarre i file di training e la cartella con i sorgenti del client, compilarli localmente ed eseguirli;

7. un file nominato "server_start" con all'interno il seguente codice bash:

```
#!/bin/bash

cd ~
cd server
./server -display :0
```

per l'avvio del client;

Esecuzione del Deployment

In questo sottoparagrafo verrà descritta la procedura per eseguire il deployment. Dopo aver creato con il tool di deployment tutti gli script e le risorse utili possiamo passare all'esecuzione del deployment. Per essere più precisi e permettere al lettore di comprendere bene il funzionamento dello script verrà analizzato un caso di esempio. Supponiamo di aver creato con la suite per la progettazione una architettura con 4 input: un autenticazione della geometria della mano, una autenticazione dell'iride e due autenticazioni facciali e di averla salvata in un rispettivo file ".xml". A questo punto possiamo aprire tale file con il tool per il deployment. Supponiamo anche di aver configurato l'architettura come specificato di seguito:

- assegnata al server la macchina con indirizzo IP "192.168.1.100" con username:"server" e password:"server";

- assegnato al client con indirizzo IP "192.168.1.101" username:"client1" e password:"client1" un autenticazione facciale e una della geometrica della mano;
- assegnato al client con indirizzo IP "192.168.1.102" username:"client2" e password:"client2" un autenticazione dell'iride;
- assegnato al client con indirizzo IP "192.168.1.103" username:"client3" e password:"client3" un autenticazione facciale.

Cliccando sul pulsante "Create script" verranno creati tutti i file specificari nel sottoparagrafo precedente. In a particolare andremo ad esamire il file "deployment_script" contenente il codice bash per il deployment. Di seguito viene mostrato il codice bash creato per l'esempio presentato sopra:

```
1 sshpass -p 'server' scp ./resources/haarcascades.tar.bz2
      server@192.168.1.100:~
2 sshpass -p 'server' scp ./resources/server.tar.bz2
    server@192.168.1.100:~
3 sshpass -p 'server' scp /home/nicolo/Desktop/Tesi/
    src_Paganin/src/server/arch.xml server@192
    .168.1.100:~/arch.xml
4 sshpass -p 'server' scp ./resources/server server@192
    .168.1.100:~
5 sshpass -p 'server' ssh -n -o StrictHostKeyChecking=no -
    l server 192.168.1.100 ./server &
6 sshpass -p 'client1' scp ./resources/haarcascades.tar.
    bz2 client1@192.168.1.101:~
7 sshpass -p 'client1' scp ./resources/client.tar.bz2
    client1@192.168.1.101:~
8 sshpass -p 'client1' scp ./resources/input.list
    .192.168.1.101 client1@192.168.1.101:~/input.list
9 sshpass -p 'client1' scp ./resources/server.ip
    client1@192.168.1.101:~
10 sshpass -p 'client1' scp ./resources/client client1@192
    .168.1.101:~
11 sshpass -p 'client1' ssh -n -o StrictHostKeyChecking=no
    -l client1 192.168.1.101 ./client &
12 sshpass -p 'client2' scp ./resources/haarcascades.tar.
    bz2 client2@192.168.1.102:~
13 sshpass -p 'client2' scp ./resources/client.tar.bz2
    client2@192.168.1.102:~
14 sshpass -p 'client2' scp ./resources/input.list
    .192.168.1.102 client2@192.168.1.102:~/input.list
15 sshpass -p 'client2' scp ./resources/server.ip
    client2@192.168.1.102:~
16 sshpass -p 'client2' scp ./resources/client client2@192
    .168.1.102:~
```

```
17 sshpass -p 'client2' ssh -n -o StrictHostKeyChecking=no
    -l client2 192.168.1.102 ./client &
18 sshpass -p 'client3' scp ./resources/haarcascades.tar.
    bz2 client3@192.168.1.103:~
19 sshpass -p 'client3' scp ./resources/client.tar.bz2
    client3@192.168.1.103:~
20 sshpass -p 'client3' scp ./resources/input.list
    .192.168.1.103 client3@192.168.1.103:~/input.list
21 sshpass -p 'client3' scp ./resources/server.ip
    client3@192.168.1.103:~
22 sshpass -p 'client3' scp ./resources/client client3@192
    .168.1.103:~
23 sshpass -p 'client3' ssh -n -o StrictHostKeyChecking=no
    -l client3 192.168.1.103 ./client &
```

Il codice è così strutturato: le righe $1-5$ sono relative al deployment del server mentre le restanti righe sono relative ai client. In particolare:

- la riga 1 copia nella home dell'indirizzo fornito per il server la cartella compressa contenente i file di training;
- la riga 2 copia nella home dell'indirizzo fornito per il server la cartella compressa contenente i sorgenti del server;
- la riga 3 copia nella home dell'indirizzo fornito per il server il file "arch.xml" aperto in fase iniziale;
- la riga 4 copia nella home dell'indirizzo fornito per il server lo script di decompressione e compilazione;
- la riga 5 esegue quest'ultimo script.

Le restanti righe relative al client eseguono le stesse operazioni a differenza che al posto di essere copiato il file "arch.xml" con l'architettura vengono copiati i relativi file "input.list" contenenti le informazioni relative agli input che ogni client deve eseguire.

Per l'esecuzione di tale script sono neccessari alcuni programmi esterni che sono spiegati nell' appendice **??**.

5.2.5 Esecuzione

In questo paragrafo verrà desritta l'implementazione dei tool per eseguire l'architettura. Essendo l'architettura distribuita, si è pensato ad un tool client/server di esecuzione dove:

- i client hanno il compito di eseguire l'acquisizione dei dati neccessati all'autenticazione e di mandarli tramite procedura remota sicura al server;

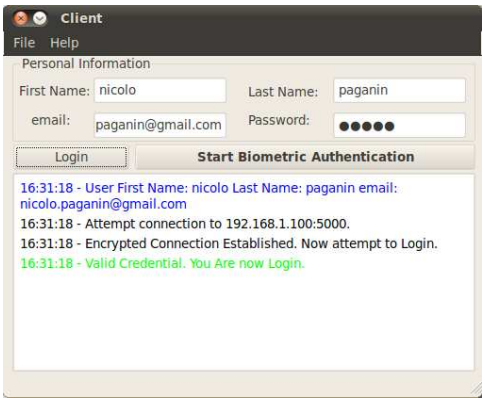

Figura 5.15: Screenshot del tool di esecuzione lato client

- il server ha il compito di eleborare tali dati con gli algoritmi di autenticazioni, procedere alla verifica dell'identità del soggetto ed eseguire l'architettura.

Le tecnologie principali che sono stati usate per soddisfare tali vincoli, e che verranno spiegate nei prossimo sottoparagrafi, sono:

- **TCP/IP** per il collegamento remoto.
- **SSL** per il collegamento sicuro tra client e server;
- **Multitrheding** per la gestione indipendente di ogni connessione;

In figura 5.15 è mostrato uno screenshot del client mentre in figura 5.16 è mostrato quello del server.

Protocollo d'Esecuzione

Di seguito verrà spiegato il protocollo d'esecuzione dell'architettura in relazione all'autenticazione facciale. Nel momento in cui viene fatto il deployment del software il sistema può essere eseguito mediante lo script "start_script". A questo punto il server è attivo e in ascolto e le sue informazioni sono mostrate nella parte superiore del tool e i client sono attivi e in attesa di eseguire l'acquisizione di dati biometrici. Di seguito verrà descritto il protocollo di autenticazione:

1. Quando l'utente arriva di fronte alla postazione per eseguire l'autenticazione inserisce nella parte superiore del programma i dati richiesti e preme il pulsante "Login". A questo punto viene instaurata una connessione sicura con il server a cui vengono spediti i dati immessi dall'utente.

Figura 5.16: Screenshot del tool di esecuzione lato server

2. Il server controlla la valità di tali dati con una query nel database. Se i dati sono corretti manda conferma al client dell'esito positivo della connessione altrimenti disconnette la connessione.

3. Se il server risponde con un messaggio positivo e l'utente viene autenticato il pulsante "Start Biometric Authentication" diventerà attivo e sarà possibile far partire l'autenticazione cliccandolo.

4. Verranno acquisiti, nell'ordine specificato nel file "input.list", i parametri biometrici richiesti. Nel caso dell'autenticazione facciale comparirà il box mostrato in figura 5.17. In questo caso le immagini contenenti la faccia dell'utente da autenticare vengolo acquisite e spedite al server. Per ridurre il tempo di invio delle immagini, l'algoritmo "Haar like" di localizzazione viene eseguito direttamente nel client, in questo modo vengono spedite al server immagini di pochi Kb il cui invio impiega pochi ms.

5. nel momento in cui il server riceve i parametri biometrici acquisiti dal client esegue gli algoritmi di autenticazione, scrive il risultato nella tabella "authList" del database che è stata specificata nel sottoparagrafo 5.2.2 e informa il client dell'esito dell'autenticazione.

6. Ogni volta che il server esegue un algoritmo di autenticazione e inserisce un record nella tabella "authList" esegue l'archiettura quindi, partendo dai blocchi di fusione al livello 0 fino al livello del blocco di output:

   (a) cerca nel database se gli input di tali blocchi sono già stati acquisiti e se non sono mai stati usati per precedenti autenticazioni;

Figura 5.17: Box di autenticazione facciale lato client

(b) se sono disponibili delle acquisizioni che non sono mai state utilizzate (quindi hanno il campo "authFlag" impostato a 0) seleziona l'output dell'ultima acquisizione in ordine temporale e la utilizza come input del blocco di fusione selezionato;

(c) calcola l'output di tale blocco e inserisce nella tabella "authList" il suo valore, aggiornando il campo "authFlag" di tutti i record che corrispondevano agli input usati della tabella "authList" ad 1;

(d) continua il calcolo dei blocchi di fusione fino a quando non trova un blocco i cui input non sono ancora stati acquisiti;

(e) se tutti gli input sono stati acquisiti esegue l'architettura completamente e per ogni blocco di fusione che calcola notifica il client del risultato.

Implementazione di TCP/IP e Multithreading

In questo sottoparagrafo verrà descritta l'implementazione del protocollo TCP/IP. Il protocollo TCP/IP in Qt viene implementato con l'utilizzo di due classi. La prima è "QTcpServer" è ha il compito di creare un oggetto server che resta in attesa di connessioni da parte da client e la seconda è "QAbstrctSocket" che ha il compito di implementare tutte le funzioni base di un socket, sia esso un socket TCP, UDP con o senza SSL. In particolare nell'implementazione la classe QServerSocket è stata estesa creando una nuova classe "STcpServer" che sovrascrive la funzione "incomingConnection", questo perchè avevamo la necessità di rendere ogni nuova connessione un thread indipendente. In Qt esiste una classe specifica che implementa il threading, tale classe si chiama "QThread". Per rendere possibile il multithreding tale classe va estesa, è stata quindi creta una nuova classe "ServerThread" che estende "QThread".

Quindi, per prima cosa è stato creato un oggetto server della classe "STcpServer" portandolo successivamente nello stato di attesa con "server-¿listen();". Appena un client effettua una connessione viene richiamato il metodo "incomingConnection". In tale metodo è contenuto il seguente codice:

```
1 ServerThread *thread = new ServerThread(socketDescriptor
    , this, archMap, CA_CERTIFICATES_PATH,
    KEY_CERTIFICATES_PATH, KEY_PASS);
2 connect(thread, SIGNAL(finished()), thread, SLOT(
    deleteLater()));
3 thread->start();
```

La prima riga crea un nuovo thread e lo inizializza con i dati della connessione, mentre la riga 3 esegue il thread. Nel momento in cui il thread viene eseguito, la sua funzione "run()" viene eseguita. L'estensione della classe "QThread" ha l'obiettivo di fare "l'overrriding" di tale funzione. Il codice contenuto all'interno della funzione "run()" è il seguente (vengono di seguito mostrate solo le linee di codice essenziali):

```
1 sslSocket = new QSslSocket;
2 if (sslSocket->setSocketDescriptor(socketDescriptor)) {
3    connect(sslSocket, SIGNAL(encrypted()), this, SLOT(
       ready()));
4   sslSocket->startServerEncryption();
5    } else {
6        delete sslSocket;
7    }
8
9    connect(sslSocket, SIGNAL(readyRead()), this, SLOT(
       readData()), Qt::DirectConnection);
10   connect( sslSocket, SIGNAL(sslErrors(QList<QSslError
       >)),this, SLOT(sslErrors(QList<QSslError>)) );
11   connect( sslSocket, SIGNAL(disconnected()),this,
       SLOT(disconnect()) );
12   exec();
```

Come si può vedere nella prima riga viene inizializzato un nuovo oggetto socket, a tale oggetto viene passato in inizializzazione il descrittore della connessione contenuto all'interno della variabile "socketDescriptor" e alla fine viene avviato con la funzione "sslSocket-¿startServerEncryption();". A questo punto il socket è attivo e funzionante non appena verranno scritti dei dati verrò eseguita la funzione "readyRead()" che ha il compito di leggere i dati in entrata.

In modo simile nel client viene creato un oggetto "SslSocket" che ha il compito di effettuare la connessione al server con il seguento codice:

```
1 sslSocket->connectToHostEncrypted(IP,PORT.toInt(),cert.
    at(0).issuerInfo(QSslCertificate::CommonName));
```

e quando la connessione è stata instaurata di spedire al server le informazioni: Di seguito viene mostrato un esempio in cui vengono spedite al server le informazioni relative all'anagrafica dell'utente che si stà autenticando:

```
1  QString fisrtName = ui->lineEdit_3->text();
2      QString lastName = ui->lineEdit_5->text();
3      QString email = ui->lineEdit_6->text();
4      QString password = ui->lineEdit_4->text();
5      QByteArray block;
6      QDataStream out(&block, QIODevice::WriteOnly);
7      out.setVersion(QDataStream::Qt_4_0);
8      out << (qint64)0;
9      out << mode.at(0); //user data mode
10     out << fisrtName;
11     out << lastName;
12     out << email;
13     out << password;
14     out.device()->seek(0);
15     sslSocket->write(block);
```

I dati che si devono spedire vengono inseriti in una variabile "QDataStream", che ha il compito di inserire in un array binario i dati, e poi vengono spediti con la funzione "write()" nel socket.

Implementazione di SSL

Dato che nelle connessioni da client a server viaggiano dati personali, la connessione deve essere resa sicura con un algoritmo clittografico quale RSA. Qt mette a disposizione la class "QSslSocket" che permette di instaurare connessioni criptate. Per fare questo devono essere creati le chiavi private e pubbliche del server e del certificati che ne garantisca l'autenticità. Per la creazioni di chiavi pubbliche e private è stato utilizzato "OpenSSL" che è un programma opensource che implementa SSL. Sono stati creati i seguenti file:

- ca.crt contenente la chiave pubblica del server;
- ca.key contente la chiave privata del server;

I file sono stati creati con i seguenti comandi:

```
1 openssl req -new -x509 -days 1757 -sha1 -newkey rsa:4096
     -keyout ca.key -out ca.crt
2 openssl req -new -sha1 -newkey rsa:1024 -nodes -keyout
    client.key -out client_req.pem
3 openssl ca -out client_sign.crt -cert ca.crt -keyfile ca
    .key -infiles client_req.pem
```

Tali file devono essere quindi associati alla connessione SSL. L'oggetto SslSocket ha implementate le funzioni "setPrivateKey()" e "addCaCertificate()" che permettono di associare certificato e chiave privata al socket in modo da permettere delle connessioni criptate. Le due funzioni sono state usate nel seguente modo:

```
1    sslSocket->setPrivateKey( myKey );
2     sslSocket->addCaCertificate(myCert);
```

Capitolo 6

Conclusioni

L'obiettivo principale del progetto di cui fa parte questa tesi era lo sviluppo di un tool completo di strumenti per la creazione flessibile di architetture per l'autenticazione multibiometrica, per il deployment e l'esecuzione delle stesse.

Questa tesi sviluppa obiettivi riguardanti la realizzazione degli strumenti software per il deployment automatico e l'esecuzione dell'architettura di un sistema di autenticazione multibiometrico e implementa uno dei più noti algoritmi per il rilevamento e riconoscimento facciale. In particolare, si è cercato di rendere questi strumenti user-friendly, semplificando al massimo la configurazione degli stessi e creando delle comode interfacce grafiche che permettano un facile apprendimento e una rapida esecuzione dell'architettura.

Uno dei principali obiettivi che è stato seguito durante la progettazione e l'implementazione del software è la modularità del codice. Per rendere il progetto flessibile si è tentato, il più possibile, di sfruttare la potenza della programmazione ad oggetti per rendere il codice facilmente estendibile nel caso, in lavori futuri, si renda neccessario aggiungere ulteriori configurazioni dell'architettura, differenziare ancora di più le possibilità di fusione dei risultati intermedi oppure modificare il protocollo di deployment ed esecuzione dell'architettura.

Per rendere più facile ed efficiente la gestione dei dati si è usato un DBMS in modo che le operazioni sui dati siano veloci e i dati siano accessibili in modo sicuro da postazione remote. Il database è stato pensato per essere condiviso da tutte gli strumenti quindi la sua progettazione è stata influenzata anche dal rendere le tabelle il più possibile generiche e usare la potenza dei DBMS relazionali per creare una base di dati facilmente modificabile ed estendibile. Molta importanza è stata data alla sicurezza, trovandosi in un sistema distribuito, i cui dati viaggiano attraverso reti locali o reti internet, si è reso necessario utilizzare tecnologie per criptare i dati e rendere le

connessioni sicure.

Sviluppi futuri potranno interessare l'implementazione di algoritmi di riconoscimento più sofisticati, di protocolli di comunicazione più sicuri in grado di gestire scenari più complessi, l'accesso al sistema con smart card o dispositivi integrati e l'esecuzione in dispositivi con meno capacità computazionali come smartphone, netbook o altri dispositivi embedded.

Bibliografia

[1] Marios Savvides, B.V.K. Vijaya Kumar and Pradeep Khosla *Face Verification using Correlation Filters* Proc. Of the Third IEEE Automatic Identification Advanced Technologies, pp.56-61, Tarrytown, NY, March 2002.

[2] Sébastien Marcel, Yann Rodriguez and Guillaume Heusch *On the Recent Use of Local Binary Patterns for Face Authentication* in: International Journal on Image and Video Processing Special Issue on Facial Image Processing, 2007

[3] Kumar B.V.K.V., Savvides M. and Chunyan Xie *Correlation Pattern Recognition for Face Recognition* in: Proceedings of the IEEE, Vol. 94, No. 11, November 2006 pag.1963-1976

[4] Anil K. Jain, Patrick Flynn and Arun A. Ross *Handbook of Biometrics* ISBN-13: 978-0-387-71040-2 Springer pag:43-70

[5] Amira Beccheroni. *Biometria: storia, futuro ed etica.* URL: `http://www.lswn.it/biologia/articoli/biometriastoriafuturoedetica`

[6] Bojan Cukic. *Introduction to Biometrics.* Master's thesis, West Virginia University, CITeR Center for Identification Technology Research.

[7] W. Zhao, R. Chellappa, A. Rosenfeld, and P. J. Phillips. *Face Recognition: A Literature Survey.* ACM Computing Surveys, pages 399–458, 2003.ù

[8] M. Turk and A. Pentland. *Eigenfaces for Recognition.* Journal of Cognitive Neuroscience, 3(1):71–86, 1991.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience Publication, 2000.

[10] M. S. Bartlett, J. R. Movellan, and T. J. Sejnowski. *Face recognition by indepen- dent component analysis.* IEEE Transactions on Neural Networks, 13(6):1450– 1464, 2002.

[11] P. S. Penev and J. J. Atick. *Local Feature Analysis: A General Statistical Theory for Object Representation.* Network: Computation in Neural Systems, 7(3):477–500, 1996.

[12] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. *Face Recognition: A Con- volutional Neural-Network Approach.* IEEE Transactions on Neural Networks, 8(1):98–113, 1997.

[13] M. A. O. Vasilescu and D. Terzopoulos. *Multilinear analysis of image ensembles: TensorFaces.* In Proceedings European Conference on Computer Vision, pages 447–460, 2002.

[14] D. O. North, *analysis of the factors which determine signal/noise discriminations in pulsed carrier systems*,Proc. IEEE, vol. 51, no. 7, pp. 1016–1027, Jul. 1963.

[15] C. F. Hester and D. Casasant, BMultivariant technique for multiclass pattern recognition,[Appl. Opt., vol. 21, pp. 4016–4019, 1982.

[16] T. Ahonen, A. Hadid and M. Pietik inen, *Face recognition with local binary patterns*, European Conference on Computer Vision, a Prague, 469–481, 2004.

[17] Paul A. Viola, Michael J. Jones: *Rapid Object Detection using a Boosted Cascade of Simple Features.* CVPR (1) 2001: 511-518

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

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

[20] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. IEEE Transactions on Pattern Analyszs and Machine Intelligence 11(7):674-93, July 1989.

[21] Arun Ross Anil K. Jain and Salil Prabhakar. *An introduction to Biometric Recognition.* Master's thesis, West Virginia University, Michigan State University, January 2004.

[22] Lin Hong Anil Jain and Sharath Pankanti. *Biometric identification.* Communications of the ACM, 43(2):91[U+FFFD]98, February 2000.

[23] Dina Sanchez Uwe Bubeck. *Biometric Authentication.* Master's thesis, San Diego State University, Spring 2003.

[24] Simon Liu and Mark Silverman. *A Practical Guide to Biometric Security Technology.* Communications of the ACM, IT Pro(01):27-32, January- February 2001.

[25] A. Ross and A.K. Jain. *Multibiometric Systems.* Communications of the ACM, 47(1): 34-40, January 2004.

[26] Arun Ross. *An introdution to multibiometrics.* Master's thesis, West Virginia University, Morgantown, WV 26506 USA, September 2007.

[27] Uwe M. Bubeck. *Multibiometric Authentication* An overview of Recent Developments. Master's thesis, San Diego State University, Spring 2003.

[28] L. et al. Hong. *Can Multibiometrics Improve Performance?* Proceeding AutoID, 1999.

[29] A. Ross and A.K. Jain. *Information Fusion in Biometrics.* Master's thesis, Michigan State University, March 2003.

[30] Arun Ross and Robin Govindarajan. *Feature Level Fusion Using Hand and Face Biometrics.* Master's thesis, West Virginia University, Motorola Inc., Anaheim, February 2001.

[31] Arun Ross and Anil K. Jain. *Multimodal Biometrics: an overview.* Master's thesis, West Virginia University, Michigan State University, September 2004.

[32] Anil K. Jain Salil Prabhakar. *Decision-level fusion in fingerprint verification.* Master's thesis, Algorithms Research Group, Digital Persona, Inc. Redwood City; Michigan State University, February 2001.

[33] Arun Ross Anil K. Jain and Sharath Pankanti. *Biometrics: a tool for infomation security.* IEEE Transactions on Information Forensic and Security, 1(2):125-143, June 2006.

[34] Arun Ross Anil Jain, Karthik Nandakumar. *Score normalization in multimodal biometric systems.* Science direct, 38(38): 2270-2285, December 2005.

[35] E.M. Ronchetti W.A. Stahel F.R. Hampel, P.J. Rousseeuw. *Robust Statistics: The Approach Based on Influence Functions.* Master's thesis, Whiley, New York 1986.

[36] A. Ross and A.K. Jain. *Learning user-specific parameters in a multibiometric system.* Master's thesis, Michigan State University, department of Computer Science and Engineering, September 2002.

[37] R. Anil, A.L.C. Yin, and O. Küçük *PAM Face Authentication: An Opensource Security Module for Linux Systems* Proceeding of Artificial Intelligence and Applications (AIA 2010)

[38] G. Garcia-Mateos. *Refining face tracking with integral projections.* In 4th Intl. Conf. on Audio- and Video-based Biometric Person Authentication (AVBPA), volume LNCS 2688,pages 360–368, Guildford, UK, 2003.

[39] Lienhart, R. and Maydt, J. *An extended set of haar-like features for rapid object detection*, ICIP, (2002)

[40] A. Dearle *Software Deployment, Past, Present and Future*, Future of Software Engineering(FOSE'07)

[41] S.N. Srihari T.K. Ho, J.J. Hull. *Decision Combination in Multiple Classifier Systems.* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 16, no. 1, pp. 66-75, January 1994.

[42] S. Lucey. *Audio Visual Speech Processing.* Ph.D. thesis, Queensland University of Technology, 2002.

[43] J. Luettin. *Visual Speech and Speaker Recognition.* Ph.D. thesis, Department of Computer Science, University of Sheffield, 1997.

[44] T. Chen and R. Rao. *Audio-Visual Integration in Multimodal Communications.* Proc. IEEE, vol. 86, no. 5, pp. 837-852, 1998.

[45] C. Cerisara. *Contribution de l'Approache Multi-Bande [U+FFFD] la Reconnaissance Automatic de la Parole.* Ph.D. thesis, Institute Nationale Polyt[U+FFFD]chnique de Lorraine, Nancy, France, 1999.

[46] S. Dupont. *Etude et D[U+FFFD]velopment de Nouveaux Paradigmes pour la Reconnaissance Robuste de la Parole.* Ph.D. thesis, Laboratoire TCTS, Universite de Mons, Belgium, 2000.

[47] Astrid Hagen. *Robust Speech Recognition Based on Multi-Stream Processing.* Ph.D. thesis, Ecole Polytechnique Federale de Lausanne, Switzerland, 2001.

[48] L. Shire. *Discriminant Training of Front-End and Acoustic Modeling Stages to Heterogeneous Acoustic Environments for Multi-Stream Automatic Speech Recognition.* Ph.D. thesis, University of California, Berkeley, USA, 2001.

[49] Ji Ming and F. Jack Smith. *Speech Recognition with Unknown Partial Feature Corruption - a Review of the Union Model.* Computer Speech and Language, vol. 17, pp. 287-305, 2003.

[50] C. Sanderson. *Automatic Person Verification Using Speech and Face Information.* Ph.D. thesis, Griffith University, Queensland, Australia, 2002.

[51] K. Nandakumar. *Integration of Multiple Cues in Biometric Systems.* M.S. thesis, Michigan State University.

[52] N. Poh. *Multi-system Biometric Authentication: Optimal Fusion and User-Specific Information.* Ph.D. thesis, Swiss Federal Institute of Technology in Lausanne (Ecole Polytechnique F[U+FFFD]d[U+FFFD]rale de Lausanne), 2006.

[53] K. Kryszczuk. *Classification with Class-independent Quality Information for Biometric Verification.* Ph.D. thesis, Swiss Federal Institute of Technology in Lausanne (Ecole Polytechnique F[U+FFFD]d[U+FFFD]rale de Lausanne), 2007.

[54] J. Richiardi. *Probabilistic Models for Multi-Classifier Biometric Authentication Using Quality Measures.* Ph.D. thesis, Swiss Federal Institute of Technology in Lausanne (Ecole Polytechnique F[U+FFFD]d[U+FFFD]rale de Lausanne), 2007.

[55] A. Ross, K. Nandakumar, and A.K. Jain. *Handbook of Multibiometrics.* Springer Verlag, 2006.

[56] A. Fejfar. *Combining Techniques to Improve Security in Automated Entry Control.* In Carnahan Conf. On Crime Countermeasures, 1978, Mitre Corp. MTP-191.

[57] J. Bigun, J. Fierrez-Aguilar, J. Ortega-Garcia, and J. Gonzalez- Rodriguez. *Multimodal Biometric Authentication using Quality Signals in Mobile Communications.* In 12th Int'l Conf. on Image Analysis and Processing, Mantova, 2003, pp. 2-13.

[58] J. Fierrez-Aguilar, J. Ortega-Garcia, J. Gonzalez-Rodriguez, and J. Bigun. *Kernel-Based Multimodal Biometric Verification Using Quality Signals.* In Defense and Security Symposium, Workshop on Biometric Technology for Human Identification, Proc. of SPIE, 2004, vol. 5404, pp. 544-554.

[59] J. Kittler, N. Poh, O. Fatukasi, K. Messer, K. Kryszczuk, J. Richiardi, and A. Drygajlo. *Quality Dependent Fusion of Intramodal and Multimodal Biometric Experts.* In Proc. of SPIE Defense and Security Symposium, Workshop on Biometric Technology for Human Identification, 2007, vol. 6539.

[60] K. Nandakumar, Y. Chen, S. C. Dass, and A. K. Jain. *Likelihood ratio based biometric score fusion.* IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 30, pp. 342-347, 2008.

[61] K. Kryszczuk and A. Drygajlo. *Credence estimation and error prediction in biometric identity verification.* Signal Processing, vol. 88, pp. 916-925, 2008.

[62] H. Fronthaler, K. Kollreider, J. Bigun, J. Fierrez, F. Alonso-Fernandez, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. *Fingerprint image-quality estimation and its application to multialgorithm verification.* IEEE Trans. on Information Forensics and Security, vol. 3, pp. 331-338, 2008.

[63] Y. Chen, S.C. Dass, and A.K. Jain. *Fingerprint Quality Indices for Predicting Authentication Performance.* In LNCS 3546, 5th Int'l. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA 2005), New York, 2005, pp. 160-170.

[64] Y. Chen, S. Dass, and A. Jain. *Localized iris image quality using 2-d wavelets.* In Proc. Int'l Conf. on Biometrics (ICB), Hong Kong, 2006, pp. 373-381.

[65] X. Gao, R. Liu, S. Z. Li, and P. Zhang. *Standardization of face image sample quality.* In LNCS 4642, Proc. Int'l Conf. Biometrics (ICB'07), Seoul, 2007, pp. 242-251.

[66] National Institute of Standards and Technology. *Nist speech quality assurance package 2.3 documentation.*

[67] S. Muller and O. Henniger. *Evaluating the biometric sample quality of handwritten signatures.* In LNCS 3832, Proc. Int'l Conf. Biometrics (ICB' 07), 2007, pp. 407-414.

[68] S. Bengio, C. Marcel, S. Marcel, and J. Marithoz. *Confidence Measures for Multimodal Identity Verification.* Information Fusion, vol. 3, no. 4, pp. 267-276, 2002.

[69] N. Poh and S. Bengio. *Improving Fusion with Margin-Derived Confidence in Biometric Authentication Tasks.* In LNCS 3546, 5th Int'l. Conf. Audio- and Video-Based Biometric Person Authentication (AVBPA 2005), New York, 2005, pp. 474-483.

[70] K-A. Toh, W-Y. Yau, E. Lim, L. Chen, and C-H. Ng. *Fusion of Auxiliary Information for Multimodal Biometric Authentication.* In LNCS 3072, Int'l Conf. on Biometric Authentication (ICBA), Hong Kong, 2004, pp. 678-685.

[71] O. Fatukasi, J. Kittler, and N. Poh. *Quality Controlled Multimodal Fusion of Biometric Experts.* In 12th Iberoamerican Congress on Pattern Recognition CIARP, Via del Mar-Valparaiso, Chile, 2007, pp. 881-890.

[72] D. E. Maurer and J. P. Baker. *Fusing multimodal biometrics with quality estimates via a bayesian belief network.* Pattern Recognition, vol. 41, no. 3, pp. 821-832, 2007.

[73] N. Poh, G. Heusch, and J. Kittler. *On Combination of Face Authentication Experts by a Mixture of Quality Dependent Fusion Classifiers.* In LNCS 4472, Multiple Classifiers System (MCS), Prague, 2007, pp. 344-356.

[74] F. Alonso-Fernandez, J. Fierrez, D. Ramos, and J. Ortega-Garcia. *Dealing with sensor interoperability in multi-biometrics: The upm experience at the biosecure multimodal evaluation 2007.* In Proc. of SPIE Defense and Security Symposium, Workshop on Biometric Technology for Human Identification, 2008.

[75] N. Poh, T. Bourlai, and J. Kittler. *Improving Biometric Device Interoperability by Likelihood Ratio-based Quality Dependent Score Normalization.* In accepted for publication in IEEE Conference on Biometrics: Theory, Applications and Systems, Washington, D.C., 2007, pp. 1-5.

[76] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas. *Score Normalization for Text-Independant Speaker Verification Systems.* Digital Signal Processing (DSP) Journal, vol. 10, pp. 42-54, 2000.

[77] Krzysztof Kryszczuk, Jonas Richiardi, Plamen Prodanov, and Andrzej Drygajlo. *Reliability-based decision fusion in multimodal biometric verification systems.* EURASIP Journal of Advances in Signal Processing, vol. 2007, 2007.

[78] W. Li, X. Gao, and T.E. Boult. *Predicting biometric system failure.* Computational Intelligence for Homeland Security and Personal Safety, 2005. CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on, pp. 57-64, 31 2005-April 1 2005.

[79] B. Xie, T. Boult, V. Ramesh, and Y. Zhu. *Multi-camera face recognition by reliability-based selection.* Computational Intelligence for Homeland Security and Personal Safety, Proceedings of the 2006 IEEE International Conference on, pp. 18-23, Oct. 2006.

[80] T. P. Riopka and T. E. Boult. *Classification enhancement via biometric pattern perturbation.* In AVBPA, 2005, pp. 850-859.

[81] U.R. Sanchez and J. Kittler. *Fusion of talking face biometric modalities for personal identity verification.* In IEEE Int'l Conf. Acoustics, Speech, and Signal Processing, 2006, vol. 5, pp. V-V.

[82] K Messer, J Matas, J Kittler, J Luettin, and G Maitre. *Xm2vtsdb: The extended m2vts database.* In Second International Conference on Audio and Video-based Biometric Person Authentication, 1999.

[83] N. Poh and J. Kittler. *On Using Error Bounds to Optimize Cost-sensitive Multimodal Biometric Authentication.* In Proc. 19th Int'l Conf. Pattern Recognition (ICPR), 2008.

[84] G. E. Box and D. R. Cox. *An Analysis of Transformations.* Automatic Identification Advanced Technologies, 2007 IEEE Workshop on, vol. B, no. 26, pp. 211-246, 1964.

[85] N. Poh and S. Bengio. *How Do Correlation and Variance of Base Classifiers Affect Fusion in Biometric Authentication Tasks?* IEEE Trans. Signal Processing, vol. 53, no. 11, pp. 4384-4396, 2005.

[86] Francesco Locascio. *Sistemi di Autenticazione Multibiometrica: strumenti a supporto della progettazione.* Master's thesis, Padova University, December 2010.

Appendice A

OpenCV

Da sempre uno dei campi più affascinanti della ricerca scientifica è volto alla riproduzione artificiale delle capacità umane. Tra queste capacità la visione, intesa come pura acquisizione di immagini, attualmente considarabile un problema già risolto, o almeno un punto già segnato, visto che le capacità visive di sistemi ottici e relativi sensori hanno ampiamente superato le possibilità dell'occhio umano in quanto a sensibilità, velocità e risoluzione. Il passo successivo, cioè la capacità di interpretare ed utilizzare correttamente le informazioni acquisite, presenta invece ancora molti problemi insoluti. Convertire un'immagine in informazioni "oggettive" astraendone il contenuto dalla pura rappresentazione luminosa, sebbene sia un'operazione banale per un cervello umano adulto è, a tutt'oggi, un problema di elevata complessità per un sistema automatico. Oltretutto il campo di ricerca è evidentemente molto giovane, con meno di trent'anni di esperienza. In quest'ottica si inserisce la necessità di una base comune di strumenti analitici, primo dei quali una libreria che raccolga le funzionalità degli algoritmi più utilizzati, oltre che una serie di formati di rappresentazione dei dati secondo standard aperti e condivisi. Le librerie OpenCV (Open Computer Vision) nascono appunto a questo scopo. Lo sviluppo prende le mosse da un gruppo di ricerca sponsorizzato da Intel. E' infatti parzialmente basata sulla Intel Image Processing Library (IPL). Tale prodotto è oggi integrato nella libreria commerciale IIPP (Intel Integrated Performance Primitives), con cui conserva piena compatibilità, e che può eventualmente rendere disponibili un completo ventaglio di funzioni di trattamento segnali (audio, video, sintesi vocale, crittografia, ecc) oltre che una migliore ottimizzazione delle prestazioni. Si avvale inoltre

di numerosi contributi dalle più svariate provenienze. A tal proposito l'elenco di credits citati dalla pagina ufficiale è quantomeno impressionante, si va da ricercatori del MIT, fino a docenti della Berkley University. Inutile sottolineare come questo offra già una certa garanzia relativamente alla buona qualità del codice e degli algoritmi applicati. Tra i punti di forza sottolineiamo inoltre la politica di licenza utilizzata, in stile BSD.

A.1 Panoramica

Con il termine "libreria grafica" infatti si identificano genericamente almeno tre famiglie di librerie i cui scopi sono sostanzialmente differenti:

- I *Toolkit*, ovvero librerie di primitive per la creazione di oggetti grafici di interfaccia (finestre, icone, bottoni, etc.);
- *librerie di rendering e multimedia*, come DirectX e OpenGL, orientate alla massima performance nella creazione di effetti poligonali o vettoiali. L'utilizzo più comune è teso all'ottenimento di elevate prestazioni grafiche sfruttate ad esempio nei videogiochi o nelle applicazioni multimediali;
- *librerie di gestione hardware grafico*, come digitalizzatori e frame grabber. Pur includendo tipicamente una base di funzioni di trattamento sono generalmente da considerarsi come API dei relativi "driver" hardware.

Le OpenCV, pur includendo alcune funzionalità tipiche di ciascuna delle famiglie citate, non fanno parte di nessuno di questi gruppi. L'utilizzo primario è infatti quello collegato alla visione artificiale, il cui problema principale, come già visto, è quello di estrarre dalle immagini dati significativi e trattabili in modo automatico. Tale campo di studio trova le sue applicazioni più comuni nella robotica, nei sistemi di videosorveglianza evoluti e nei sistemi di monitoraggio e sicurezza, oltre che in ogni sistema di archiviazione automatica di informazioni visive. La libreria include attualmente più di 300 funzioni, che coprono le più svariate esigenze di trattamento di immagini, comprese funzioni matematiche ottimizzate (elevamento a potenza, logaritmi, conversioni cartesiane-polari, ecc.) ed un completo pacchetto di algebra matriciale, sviluppato funzionalmente al resto del sistema.

A.1.1 Informazioni Pratiche

I siti principali del progetto sono due, il primo facente capo ad Intel (`http://intel.com/technology/computing/opencv/`) ed il secondo il classico hosting presso sourceforge (`http://sourceforge.net/projects/`

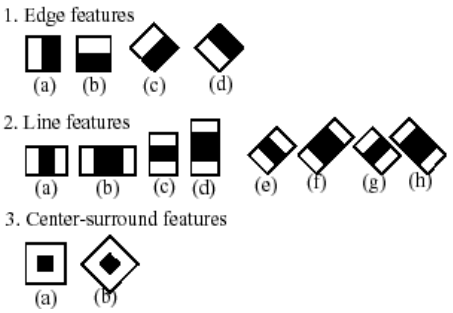

Figura A.1: Elenco delle feature implementate nell'algoritmo Haar like nell'attuale versione delle librerie di computer vision openCV

`opencvlibrary/`) dal quale è possibile procedere al download sia dei sorgenti che degli installativi per le piattaforme Microsoft. A loro volta la maggior parte delle distribuzioni linux includono OpenCV tra i pacchetti disponibili. La portabilità è quindi decisamente estesa, essendo disponibili versioni per tutte le varianti di Ms-Windows e per tutti i sistemi POSIX (Linux/BSD/UNIX/MacOsX), anche se il supporto ufficiale è per ora, limitato ai primi. A questo proposito non ci si faccia spaventare dai numeri di versione (non ancora giunta alla 1.0) e dalla etichetta "beta", in quanto il livello di stabilità attualmente offerto assicura la possibilità di utilizzo anche all'interno di ambienti di produzione. Il classico pacchetto installativo include, oltre ai binari ed agli header necessari alla compilazione, anche una discreta varietà di esempi, molto utili ad apprendere la sintassi e le tecniche di base. Sono inoltre disponibili diversi documenti introduttivi, oltre ad una reference in formato HTML. Per esigenze di spazio non ci dilungheremo sulle impostazioni di base necessarie ai differenti ambienti di sviluppo. Sottolineiamo comunque come il problema della compatibilità sia stato affrontato e risolto in modo particolarmente capillare, e quindi, tramite poche e semplici operazioni sarà possibile essere subito produttivi all'interno dei più comuni ambienti di sviluppo disponibili nel mondo Windows (VisualC++, Borland C++Builder, l'ottimo DevC++, ecc.), Linux (gcc, Kdevelop, Eclipse, ecc.) e Unix in generale. La libreria divisa in alcuni binari distinti ("dll" o "so" a seconda dei sistemi), motivo per cui il cui linking completo non è sempre necessario:

- **CxCore**. E l'oggetto principale nonchè l'unico strettamente indispensabile. Include infatti tutte le funzioni di inizializzazione delle strutture utilizzate, l'algebra lineare e le altre funzioni di base.

- **Cv e CvAux**. Includono praticamente tutte le funzioni di trattamento ed analisi, quindi il cuore delle OpenCv;

- **HighGui**. Include alcune comode funzioni GUI, come ad esempio la tipica finestra di display, oltre alle funzioni; di salvataggio e caricamento immagini da file.

- **CvCam**. Include funzioni di acquisizione video e di gestione delle telecamere (per il momento ancora piuttosto limitate).

A.1.2 Implementazione delle Haar Like Features

In openCV non poteva mancare l'implementazione dell'algortimo "Haar like" di Viola et al. [17]. Nell'attuale implementazione è stato esteso il numero di feature presenti nei classificatori. Per l'elenco completo delle feature implementate si veda la figura A.1. Si può trovare una descrizione esaustiva dell'implementazione dell'algoritmo di rilevazione al seguente sito web: `http://opencv.willowgarage.com/documentation/object_detection.html`.

Appendice B

Qt

Qt è l'ambiente di sviluppo software di Nokia, esso rappresenta nel contempo una libreria multipiattaforma e un insieme di strumenti per lo sviluppo software. Qt, originariamente sviluppato dalla Norvegese Trolltech (acquistaa da Nokia nel 2008) permette lo sviluppo di applicazioni per Ms Windows, UNIX/Linux, Mac OS X, Embedded Linux, Windows CE/Mobile, Symbian e Maemo. I linguaggi di programmazione più comunemente impiegati sono C++, Java e Python. Sono inoltre disponibili diversi bindings per altri linguaggi. Qt è disponibile sia con licenza opensource LGPL v2.1 e GPL v3.0, sia con licenza commerciale. Il motto di Qt è "Code Less. Create more. Deploy everywhere", ovvero "Scrivi meno codice. Crea di più e Distribuisci ovunque". Una breve descrizione per punti:

Che cosa sono:

- "Le qt sono librerie multi piattaforma per lo sviluppo di programmi con gui ..." (da en.wikipedia.org);
- non solo, contengono anche strumenti per applicazioni non necessariamente con gui (es: xml, sql, dbus...);
- sono scritte in c++ ma esistono binding per diversi linguaggi (es: python, java..);
- sono rilasciate sotto licenza LGPL[1].

[1]LGPL: programmi che linkano le lib possono essere close source

Perchè e chi?

- Sono multipiattaforma;
- sono orientate agli oggetti;
- sono performanti e complete;
- sono ben documentate;
- sono supportate da una grossa comunità;
- sono opensource;
- sono la base di kde4;
- software come google-earth, skype, opera10, adobe photoshop album ecc. sono scritti utilizando le qt. (fonte wikipedia.org).

Cosa possono fare?

- Interfacce grafiche per i programmi (multi piattaforma) che si integrano sia su kde4 che su gnome (e xfce);
- possibilità di personalizzare l'aspetto delle proprie applicazioni utilizzando semplici file CSS;
- tutti gli strumenti per interfacciarsi con dbus, database sql;
- integrano al loro interno classi per l'utilizzo di openGL;
- classi per rendering html attraverso il motore webkit (base di Safari,Arora, Midori..);
- Qt Designer per costruire le propie interfacce in modo intuitivo oppure Qt Creator che integra oltre al Designer anche un Ide completo;
- strumenti per l'embedded (es: gestures per multi/touchscreen, possibilità di utilizzare direttamente il framebuffer senza X);
- molto altro ancora...;

Come si possono utilizzare?

- I requisiti: Tutto il necessario per compilare (gcc, g++, linker, make etc), le Qt (con gli headerfile), Qmake (di norma con le qt) eventualmente cmake QtCreator o QtDesigner;
- Sviluppo senza IDE (sconsigliato):

  - il progetto è formato normalmente da almeno 4 file, file progetto, il main e una coppia file di intestazione (.h) e implementazione (.cpp);
  - il file progetto conterrà le informazioni per generare il make file;
  - il main includerà l'header file e conterrà pochi comandi per visualizzare la finestra;
  - Tutto il resto è nei due file rimanenti.

- creando un nuovo progetto (così crea automaticamente il file .pro);
- creare la propria GUI attraverso il designer integrato;
- sviluppare l'applicazione vera e propria;
- collegare la UI al resto.

Appendice C

Requisti software e Risoluzione dei problemi

In questo appendice verranno elencati tutti i requisiti software e i principali problemi che si sono visti durante l'installazione degli strumenti, sviluppati in questa tesi, in differenti macchine e architetture.

C.1 Requisiti Software

Per il corretto funzionamento dei tool sviluppati è necessario avere a disposizione una macchina in cui sia presente come sistema operativo una qualsiasi distribuzione Linux. Il software è stato sviluppato su Ubuntu 10.04, ma non ci sono particolari controindicazioni nell'usare un'altra distribuzione Debian-like o Red-Hat. E' inoltre necessaria una webcam collegata e correttamente istallata. Di seguito una lista con il software necessario:

- Qt;
- Librerie Intel openCv;
- openssh-client;
- openssh-server;
- Mercurial;
- Librerie Google Re2;
- build-essential (installabile con apt o yum);
- mysql-client 5.1 o versioni successive;
- mysql-server 5.1 o versione successive;

- cmake;
- cmake-gui;
- libgtk2.0-dev
- pkg-config
- sshpass

C.2 Problemi riscontrati

Vengono di seguito riportati i problemi che si sono riscontrati nell'installazione e nella compilazione del codice in alcune macchine:

- Dopo aver installato le librerie openCv al momento della compilazione del tool di enrollment viene stampato il seguente errore error **while** loading shared libraries : libcv.so.4: cannot open shared. Per la risoluzione di questo problema è necessario seguire questi punti:
  1. Aggiungere al file /usr/local/lib il seguente percorso /etc/ld.so.conf
  2. da terminale dare il comando sudo ldconfig
- Durante l'esecuzione dei tool che usano il database mysql non viene rilevato il plugin di connessione ad DBMS "libqsqlmysql.so". Per risolvere questo problema seguire la seguente procedura:
  1. Se non lo avete già fatto passate Qt in versione OpenSource cliccando sul pulsan te in basso a sinistra sopra la freccia verde che permette la compilazione;
  2. copiate nella directory plugin nella cartella di installazione di Qt il file "libqsqlmysql.so" facilmente recuperabile in rete.

Appendice D

Doxygen

Doxygen è una applicazione per la generazione automatica della documentazione a partire dal codice sorgente di un generico software. È un progetto open source rilasciato sotto licenza GPL, scritto per la maggior parte da Dimitri van Heesch a partire dal 1997.

Doxygen è un sistema multipiattaforma (Windows, Mac OS, Linux, ecc.) ed opera con i linguaggi C++, C, Java, Objective C, Python, IDL (versioni CORBA e Microsoft), Fortran, PHP, e D. Nell'ambito del C++, è compatibile con le estensioni Qt.

È il sistema di documentazione di gran lunga più utilizzato nei grandi progetti open source in C++. Due esempi per tutti, sono l'adozione di doxygen da parte di ACE e KDE. In Java invece, la posizione leader viene meno, in virtù della presenza del concorrente Javadoc.Il sistema estrae la documentazione dai commenti inseriti nel codice sorgente e dalla dichiarazione delle strutture dati.

D.1 La documentazione prodotta

Il risultato finale è disponibile sotto forma di pagine HTML oppure nei formati CHM, RTF, PDF, LaTeX, PostScript o man pages di Unix. Il formato HTML prodotto si giova di un sistema di hyperlink molto curato che permette al lettore una agevole navigazione della struttura dei file sorgenti. La documentazione prodotta riporta anche il diagramma delle classi, nei casi in cui sono presenti relazioni di ereditarietà tra strutture dati. Grazie

all'impiego sinergico di Doxygen con Graphviz, è possibile includere nella documentazione diagrammi delle classi per tutti gli altri tipi di relazioni tra strutture dati. I documenti possono essere generati in diverse lingue, tra cui è compreso l'italiano.

Infine, il sistema è altamente e facilmente configurabile al fine di permettere all'utilizzatore di intervenire su tutti gli aspetti della documentazione prodotta.

D.2 Il formato dei documenti

Il funzionamento di Doxygen richiede una particolare formattazione dei commenti inseriti nel codice sorgente. Le regole di formattazione, oltre ad essere analoghe a quelle degli altri prodotti della categoria, sono chiaramente documentate nel manuale. Riportiamo di seguito un esempio.

```
1  /**
2   * The time class represents a moment of time.
3   *
4   * \author My Name
5   */
6  class Time {
7
8    /**
9     * Constructor that sets the time to a given value.
10    * \param timemillis Number of milliseconds passed
          since Jan 1. 1970
11    */
12   Time(int timemillis) {
13     ...
14   }
15
16   /**
17    * Get the current time.
18    * \return A time object set to the current time.
19    */
20   static Time now() {
21     ...
22   }
23 };
```

D.3 Il file di configurazione

Doxygen associa ad ogni progetto da documentare un file di configurazione che contiene le impostazioni da utilizzare per la generazione. Questo file è un elenco di assegnazioni di opportuni valori a determinati parametri

(TAG). Ogni tag è formato dalla coppia di informazioni "NOME_PARAMETRO = VALORE_PARAMETRO" analogamente a quanto avviene nei file di configurazione di numerosi altri prodotti open source. Un frammento di un esempio del file di configurazione è il seguente:

```
1 # The PROJECT_NAME tag is a single word (or a sequence
      of words surrounded
2 # by quotes) that should identify the project.
3
4 PROJECT_NAME           = MyProject
5
6 # The OUTPUT_DIRECTORY tag is used to specify the (
      relative or absolute)
7 # base path where the generated documentation will be
      put.
8 # If a relative path is entered, it will be relative to
      the location
9 # where doxygen was started. If left blank the current
      directory will be used.
10
11 OUTPUT_DIRECTORY        =
12
13 # The OUTPUT_LANGUAGE tag is used to specify the
      language in which all
14 # documentation generated by doxygen is written.
15
16 OUTPUT_LANGUAGE         = English
17
18 # The INPUT tag can be used to specify the files and/or
      directories that contain
19 # documented source files. You may enter file names like
       "myfile.cpp"
20 # or directories like "/usr/src/myproject".
21 # Separate the files or directories with spaces.
22
23 INPUT                   =
24
25 # If the value of the INPUT tag contains directories,
      you can use the
26 # FILE_PATTERNS tag to specify one or more wildcard
      pattern (like *.cpp
27 # and *.h) to filter out the source-files in the
      directories. If left
28 # blank the following patterns are tested:
29 # *.c *.cc *.cxx *.cpp *.c++ *.java *.ii *.ixx *.ipp *.i
     ++ *.inl *.h *.hh *.hxx *.hpp
30 # *.h++ *.idl *.odl *.cs *.php *.php3 *.inc *.m *.mm
31
```

```
32 FILE_PATTERNS        = *.h *.hh *.idl
33
34 # The RECURSIVE tag can be used to turn specify whether
     or not subdirectories
35 # should be searched for input files as well. Possible
     values are YES and NO.
36 # If left blank NO is used.
37
38 RECURSIVE               = YES
39
40 # If the GENERATE_HTML tag is set to YES (the default)
     Doxygen will
41 # generate HTML output.
42
43 GENERATE_HTML           = YES
44
45 # The HTML_OUTPUT tag is used to specify where the HTML
     docs will be put.
46 # If a relative path is entered the value of
     OUTPUT_DIRECTORY will be
47 # put in front of it. If left blank -html- will be used
     as the default path.
48
49 HTML_OUTPUT             = html
50
51 # If the GENERATE_LATEX tag is set to YES (the default)
     Doxygen will
52 # generate Latex output.
53
54 GENERATE_LATEX          = NO
55
56 # The LATEX_OUTPUT tag is used to specify where the
     LaTeX docs will be put.
57 # If a relative path is entered the value of
     OUTPUT_DIRECTORY will be
58 # put in front of it. If left blank -latex- will be used
      as the default path.
59
60 LATEX_OUTPUT            = latex
```

Come si vede, attraverso il file di configurazione, l'utente stabilisce:

- Il nome del progetto;
- la directory dove verrà generato il materiale (directory di destinazione);
- la lingua della documentazione prodotta;
- la directory dove si trovano i file sorgente da documentare (directory di origine);

- l'estensione dei file di input da considerare come origine;
- l'indicazione di ricorsività nella directory di origine;
- l'indicazione di generazione del formato HTML;
- nome della directory di destinazione per il formato HTML;
- l'indicazione di generazione del formato LaTeX;
- nome della directory di destinazione per il formato LaTeX.

Doxygen è in grado di generare un file di configurazione generico con il comando `doxygen -g <config-file>` il file generato automaticamente da doxygen contiene dei parametri generici che l'utente può personalizzare o lasciare invariati.

D.4 Utilizzo

Dopo aver installato Doxygen ed aver generato ed eventualmente modificato il file di configurazione, si può invocare l'esecuzione di Doxygen con il comando `doxygen <config-file>` al termine della elaborazione, il materiale prodotto sarà disponibile nella directory di destinazione indicata nel file di configurazione.

Appendice E

Formato dei dati

In questo appendice verrà descritto il formato dei dati in cui le architetture vengono salvate dai tools a supporto della progettazione in [86] e letti dal tool di deployment.

Nel progetto in considerazione, si è scelto di salvare l'architettura sottoforma di file XML per i seguenti motivi:

- La scelta dei nomi degli elementi può essere fatta per facilitare la comprensione del ruolo strutturale dell'elemento;
- La rigida struttura ad albero e l'assenza di regole di minimizzazione rendono semplice la visualizzazione e l'analisi della struttura del documento;
- XML è uno standard aperto e chiunque può realizzare strumenti che lo usino come formato dati;
- Esistono formati di dati generici per l'interscambio di dati, ma sono tutti organizzati linearmente. XML consente la creazione di strutture ad albero;
- XML permette di definire formalmente elementi ripetibili. Questo permette strutture più flessibili e complesse di altri formati di dati.
- XML si propone come la sintassi intermedia più semplice per esprimere dati complessi in forma indipendente dall'applicazione che li ha creati.

Per la traduzione del disegno dell'architettura in sintassi XML si è sfruttata la classe QXmlStreamWriter che fornisce un XML writer con un semplice streaming API. QXmlStreamWriter è la controparte di QXmlStreamReader per scrivere XML. Come la relativa classe, essa opera su un QIODevice specificato con setDevice(). La API è semplice: per ogni token XML o un evento che si desidera scrivere, il writer fornisce una funzione specializzata.

Il codice seguente descrive in maniera dettagliata gli aspetti del salvataggio e della conversione dell'architettura per ogni elemento presente nella scena, prestando attenzione a conservare le proprietà (nome, tipo, soglia di sicurezza, output, identificativo, livello, ecc.) di ciascun oggetto riportandole nel codice.

```
1  void MainWindow::saveFile() {
2      if (!scene->validateDiagram()) {
3          writeInStatusBar("(E) INVALID Diagram!
               Impossible to save.", Qt::red);
4          return;
5      }
6      QString fileName = QFileDialog::getSaveFileName(this
           ,
7                          tr("Choose a file name"), ".",
8                          tr("XML Files (*.xml)"));
9      if (fileName.isEmpty())
10         return;
11     QFile file(fileName);
12     if (!file.open(QFile::WriteOnly | QFile::Text)) {
13         QMessageBox::warning(this, tr("Dock Widgets"),
14                              tr("Cannot write file %1:\n
                                   %2.")
15                              .arg(fileName)
16                              .arg(file.errorString()));
17         return;
18     }
19
20     InputItem *inputItem;
21     OutputItem *outputItem;
22     AndItem* andItem;
23     OrItem* orItem;
24     PriorityItem* priorityItem;
25     VotingItem* votingItem;
26     Arrow *arrowItem;
27     QString inputName = "INPUT";
28     QString outputName = "OUTPUT";
29     QString andName = "AND";
30     QString arrowName = "ARROW";
31     QString orName = "OR";
32     QString priorityName = "PRIORITY";
33     QString votingName = "VOTING";
34     QList<QGraphicsItem *> sceneItems = scene->items(Qt
           ::DescendingOrder);
35
36     QXmlStreamWriter xmlWriter(&file);
37     xmlWriter.setAutoFormatting(true);
38     xmlWriter.writeStartDocument();
```

```
39      xmlWriter.writeStartElement("Diagram");
40
41      foreach (QGraphicsItem *item, sceneItems) {
42          if (item->type() == InputItem::Type) {
43
44              int pos = sceneItems.indexOf(item);
45
46              QString input = QString("\%1-\%2").arg(
                    inputName).arg(pos);
47              inputItem = qgraphicsitem_cast<InputItem *>(
                    item);
48              xmlWriter.writeStartElement(input);
49              xmlWriter.writeTextElement("Name",inputItem
                    ->getName());
50              xmlWriter.writeTextElement("Type",inputItem
                    ->getType());
51              xmlWriter.writeTextElement("Security",
                    QString::number(inputItem->getSecurity())
                    );
52              xmlWriter.writeTextElement("Output",
                    inputItem->getOutput());
53              xmlWriter.writeTextElement("InputNumber",
                    QString::number(scene->getNumberOfInput()
                    ));
54              xmlWriter.writeTextElement("Position",
                    QString::number(inputItem->getTimelinePos
                    ()));
55              QList<bool> repeatOn = inputItem->
                    getRepeatOn();
56              QList<int> repeatOnInt;
57              if (repeatOn.at(0)) repeatOnInt.append(1);
58              else repeatOnInt.append(0);
59              if (repeatOn.at(1)) repeatOnInt.append(1);
60              else repeatOnInt.append(0);
61              if (repeatOn.at(2)) repeatOnInt.append(1);
62              else repeatOnInt.append(0);
63              if (repeatOn.at(3)) repeatOnInt.append(1);
64              else repeatOnInt.append(0);
65              QString repeatOnStr = QString("\%1\%2\%3\%4"
                    ).arg(repeatOnInt.at(0)).arg(repeatOnInt.
                    at(1)).arg(repeatOnInt.at(2)).arg(
                    repeatOnInt.at(3));
66              xmlWriter.writeTextElement("repeatOn",
                    repeatOnStr);
67              QString x = QString::number(item->x());
68              xmlWriter.writeTextElement("x",x);
69              QString y = QString::number(item->y());
70              xmlWriter.writeTextElement("y",y);
71              xmlWriter.writeEndElement();
```

```
72
73          }
74
75          if (item->type() == AndItem::Type) {
76
77              int pos = sceneItems.indexOf(item);
78              QString anditem = QString("\%1-\%2").arg(
                    andName).arg(pos);
79              andItem = qgraphicsitem_cast<AndItem *>(item
                    );
80              xmlWriter.writeStartElement(anditem);
81              xmlWriter.writeTextElement("Name",andItem->
                    getName());
82              xmlWriter.writeTextElement("Level",QString::
                    number(andItem->getSettedLevel()));
83              QString x = QString::number(item->x());
84              xmlWriter.writeTextElement("x",x);
85              QString y = QString::number(item->y());
86              xmlWriter.writeTextElement("y",y);
87              QList<QList<int> > customList = andItem->
                    getCustomList();
88              if (!customList.empty()) {
89
90                  for (int i=0; i< customList.size(); i++)
                        {
91
92                      QString index = QString("custom_\%1"
                            ).arg(i);
93                      QString value = QString("
                            \%1:\%2=\%3-\%4").arg(customList.
                            at(i).at(0)).arg(customList.at(i)
                            .at(1)).arg(customList.at(i).at
                            (2)).arg(customList.at(i).at(3));
94                      xmlWriter.writeTextElement(index,
                            value);
95                  }
96
97
98              }
99
100             xmlWriter.writeEndElement();
101         }
102 [ ... ]
103     }
104
105     foreach (QGraphicsItem *item, sceneItems) {
106         if (item->type() == InputItem::Type) {
107             xmlWriter.writeStartElement("input");
108             inputItem = qgraphicsitem_cast<InputItem *>(
```

```
            item);
109         xmlWriter.writeTextElement("Name",inputItem
                ->getName());
110         xmlWriter.writeTextElement("Type",inputItem
                ->getType());
111         xmlWriter.writeTextElement("Output",
                inputItem->getOutput());
112         xmlWriter.writeTextElement("Security",
                QString::number(inputItem->getSecurity())
                );
113         xmlWriter.writeTextElement("Timeline",
                QString::number(inputItem->getTimelinePos
                ()));
114         QList<bool> repeatOn = inputItem->
                getRepeatOn();
115         QList<int> repeatOnInt;
116         if (repeatOn.at(0)) repeatOnInt.append(1);
117         else repeatOnInt.append(0);
118         if (repeatOn.at(1)) repeatOnInt.append(1);
119         else repeatOnInt.append(0);
120         if (repeatOn.at(2)) repeatOnInt.append(1);
121         else repeatOnInt.append(0);
122         if (repeatOn.at(3)) repeatOnInt.append(1);
123         else repeatOnInt.append(0);
124         QString repeatOnStr = QString("\%1\%2\%3\%4"
                ).arg(repeatOnInt.at(0)).arg(repeatOnInt.
                at(1)).arg(repeatOnInt.at(2)).arg(
                repeatOnInt.at(3));
125         xmlWriter.writeTextElement("repeatOn",
                repeatOnStr);
126         xmlWriter.writeEndElement();
127     }
128     if (item->type() != InputItem::Type && item->
            type() != OutputItem::Type && item->type() !=
             Arrow::Type) {
129         if (item->type() == AndItem::Type) {
130             andItem = qgraphicsitem_cast<AndItem *>(
                    item);
131             xmlWriter.writeStartElement("and");
132             xmlWriter.writeTextElement("Name",
                    andItem->getName());
133             xmlWriter.writeTextElement("Level",
                    QString::number(andItem->
                    getSettedLevel()));
134             QList<QString> inputNameList = andItem->
                    getNameOfInputItem();
135             if (!inputNameList.empty()) {
136
137                 for (int i=0; i< inputNameList.size
```

```
                    (); i++) {
138
139                     QString index = QString("input_
                            \%1").arg(i);
140                     QString value = QString("\%1").
                            arg(inputNameList.at(i));
141                     xmlWriter.writeTextElement(index
                            ,value);
142                 }
143             }
144             QList<QList<int> > customList = andItem
                    ->getCustomList();
145             if (!customList.empty()) {
146                 for (int i=0; i< customList.size();
                        i++) {
147                     QString index = QString("custom_
                            \%1").arg(i);
148                     QString value = QString("
                            \%1:\%2=\%3-\%4").arg(
                            customList.at(i).at(0)).arg(
                            customList.at(i).at(1)).arg(
                            customList.at(i).at(2)).arg(
                            customList.at(i).at(3));
149                     xmlWriter.writeTextElement(index
                            ,value);
150                 }
151             }
152             xmlWriter.writeEndElement();
153         }
154         if (item->type() == OrItem::Type) {
155             orItem = qgraphicsitem_cast<OrItem *>(
                    item);
156             xmlWriter.writeStartElement("or");
157             xmlWriter.writeTextElement("Name",orItem
                    ->getName());
158             xmlWriter.writeTextElement("Level",
                    QString::number(orItem->
                    getSettedLevel()));
159
160             QList<QString> inputNameList = orItem->
                    getNameOfInputItem();
161             if (!inputNameList.empty()) {
162                 for (int i=0; i< inputNameList.size
                        (); i++) {
163                     QString index = QString("input_
                            \%1").arg(i);
164                     QString value = QString("\%1").
                            arg(inputNameList.at(i));
165                     xmlWriter.writeTextElement(index
```

```
                                ,value);
166                         }
167                     }
168                     QList<QList<int> > customList = orItem->
                            getCustomList();
169                     if (!customList.empty()) {
170                         for (int i=0; i< customList.size();
                                i++) {
171                             QString index = QString("custom_
                                \%1").arg(i);
172                             QString value = QString("
                                \%1:\%2=\%3-\%4").arg(
                                customList.at(i).at(0)).arg(
                                customList.at(i).at(1)).arg(
                                customList.at(i).at(2)).arg(
                                customList.at(i).at(3));
173                             xmlWriter.writeTextElement(index
                                ,value);
174                         }
175                     }
176                     xmlWriter.writeEndElement();
177                 }
178                 if (item->type() == PriorityItem::Type) {
179                     priorityItem = qgraphicsitem_cast<
                            PriorityItem *>(item);
180                     xmlWriter.writeStartElement("priority");
181                     xmlWriter.writeTextElement("Name",
                            priorityItem->getName());
182                     xmlWriter.writeTextElement("Level",
                            QString::number(priorityItem->
                            getSettedLevel()));
183                     xmlWriter.writeTextElement("priorityItem
                            ",priorityItem->getPriorityItem());
184                     QList<QString> inputNameList =
                            priorityItem->getNameOfInputItem();
185                     if (!inputNameList.empty()) {
186                         for (int i=0; i< inputNameList.size
                                (); i++) {
187                             QString index = QString("input_
                                \%1").arg(i);
188                             QString value = QString("\%1").
                                arg(inputNameList.at(i));
189                             xmlWriter.writeTextElement(index
                                ,value);
190                         }
191                     }
192                     xmlWriter.writeEndElement();
193                 }
194                 if (item->type() == VotingItem::Type) {
```

```
195                     votingItem = qgraphicsitem_cast<
                            VotingItem *>(item);
196                     xmlWriter.writeStartElement("voting");
197                     xmlWriter.writeTextElement("Name",
                            votingItem->getName());
198                     xmlWriter.writeTextElement("Level",
                            QString::number(votingItem->
                            getSettedLevel()));
199                     QList<QString> inputNameList =
                            votingItem->getNameOfInputItem();
200                     if (!inputNameList.empty()) {
201                         for (int i=0; i< inputNameList.size
                                (); i++) {
202                             QString index = QString("input_
                                \%1").arg(i);
203                             QString value = QString("\%1").
                                arg(inputNameList.at(i));
204                             xmlWriter.writeTextElement(index
                                ,value);
205                         }
206                     }
207                     xmlWriter.writeEndElement();
208                 }
209             }
210         }
211         xmlWriter.writeEndElement();
212         xmlWriter.writeEndDocument();
213         file.close();
214         writeInStatusBar("XML saved", Qt::blue);
215         QApplication::setOverrideCursor(Qt::WaitCursor);
216         QApplication::restoreOverrideCursor();
217         statusBar()->showMessage(tr("Xml Saved"),2000);
218 }
```

Per ciò che concerne l'apertura del file XML e la relativa interpretazione, come già detto, viene sfruttata la classe QXmlStreamReader e la libreria RE2 sviluppata da Google per le espressioni regolari.

RE2 è una libreria C++ veloce, sicura e thread-friendly che si pone come valida alternativa al backtracking di espressioni regolari.

I motori di backtracking sono tipicamente ricchi di caratteristiche e funzionalità, ma possono avere prestazioni esponenziali nella taglia dell'input. RE2 sfrutta la teoria degli automi per garantire che le espressioni regolari vengano risolte in tempo lineare rispetto alla taglia dell'input. RE2 implementa limiti di memoria, in maniera tale che le ricerche siano vincolate ad un quantità fissa di memoria. RE2 è stato progettato per utilizzare un piccolo stack di dimensione fissa che non tiene conto degli input o delle espressioni regolari da processare: così RE2 è utile in ambienti multithread, dove gli stack di thread non possono crescere in maniera arbitraria. Su

input di grandi dimensioni, RE2 è spesso molto più veloce dei backtracking engine perchè l'uso della teoria degli automi consente l'applicazione di ottimizzazioni che gli altri non possono attuare. L'interfaccia di matching prevede l'uso di due operatori base: RE2::FullMatch che richiede l'espressione regolare per il match dell'intero testo in input, e RE2::PartialMatch che cerca un match per una sottostringa del testo di input. è possibile vedere i dettagli dell'implementazione nel codice seguente:

```
1  void MainWindow::openFile() {
2      int i=0;
3      int j=0;
4      filename = QFileDialog::getOpenFileName(this,
5                                              tr("Open Xml
                                                   "), ".",
6                                              tr("Xml
                                                   files (*.
                                                   xml)"));
7      QFile file(filename);
8      if (!file.open(QFile::ReadOnly | QFile::Text)) {
9          QMessageBox::warning(this, tr("Dock Widgets"),
10                               tr("Cannot write file %1:\n
                                     %2.")
11                               .arg(filename)
12                               .arg(file.errorString()));
13     }
14     Rxml.setDevice(&file);
15     Rxml.readNext();
16     while (!Rxml.atEnd()) {
17         if (Rxml.isStartElement())
18         {
19             if (Rxml.name() == "Diagram") Rxml.readNext
                   (); //file start
20             if (RE2::FullMatch(Rxml.name().toString().
                   toAscii().data(), "ARROW-(\\d+)")) //
                   Arrow
21             {
22                 QList<QGraphicsItem *> sceneItems =
                       scene->getSceneItems();
23                 Rxml.readNext();
24                 Rxml.readNext();
25                 if (Rxml.isStartElement())
26                 {
27                     Rxml.readElementText();
28                     Rxml.readNext();
29                     Rxml.readNext();
30                     RE2::FullMatch(Rxml.readElementText
                           ().toAscii().data(),".*-(\\d+)",&
                           i);
```

```
31                      Rxml.readNext();
32                      Rxml.readNext();
33                      RE2::FullMatch(Rxml.readElementText
                            ().toAscii().data(),".*-(\\d+)",&
                            j);
34                      scene->insertOpenRow(sceneItems.at(i
                            ), sceneItems.at(j));
35                  }
36              }
37              if (RE2::FullMatch(Rxml.name().toString().
                    toAscii().data(), "INPUT-(\\d+)")) //
                    Input
38              {
39                  Rxml.readNext();
40                  Rxml.readNext();
41                  if (Rxml.isStartElement())
42                  {
43                      QString name = Rxml.readElementText
                            ();
44                      Rxml.readNext();
45                      Rxml.readNext();
46                      QString type = Rxml.readElementText
                            ();
47                      Rxml.readNext();
48                      Rxml.readNext();
49                      int security = Rxml.readElementText
                            ().toInt();
50                      Rxml.readNext();
51                      Rxml.readNext();
52                      int output = Rxml.readElementText().
                            toInt();
53                      Rxml.readNext();
54                      Rxml.readNext();
55                      int inputNumber = Rxml.
                            readElementText().toInt();
56                      Rxml.readNext();
57                      Rxml.readNext();
58                      int pos = Rxml.readElementText().
                            toInt();
59                      Rxml.readNext();
60                      Rxml.readNext();
61                      int onS, onF, onMT, onMF;
62                      RE2::FullMatch(Rxml.readElementText
                            ().toAscii().data(),"(\\d)(\\d)
                            (\\d)(\\d)",&onS,&onF,&onMT,&onMF
                            );
63                      Rxml.readNext();
64                      Rxml.readNext();
65                      qreal x_input = Rxml.readElementText
```

```
                    ().toDouble();
66                  Rxml.readNext();
67                  Rxml.readNext();
68                  qreal y_input = Rxml.readElementText
                        ().toDouble();
69                  Rxml.readNext();
70                  Rxml.readNext();
71                  scene->setItemType(DiagramItem::
                        DiagramType(0),itemMenu);
72                  scene->insertOpenInputItem(x_input,
                        y_input,output,type,inputNumber,
                        pos,onS,onF,onMT,onMF, security,
                        name)
73              }
74          }
75          if (RE2::FullMatch(Rxml.name().toString().
                toAscii().data(), "OUTPUT-(\\d+)")) //
                Output
76          {
77              Rxml.readNext();
78              Rxml.readNext();
79              if (Rxml.isStartElement())
80              {
81
82                  QString name = Rxml.readElementText
                        ();
83                  Rxml.readNext();
84                  Rxml.readNext();
85                  qreal x_output = Rxml.
                        readElementText().toDouble();
86                  Rxml.readNext();
87                  Rxml.readNext();
88                  qreal y_output = Rxml.
                        readElementText().toDouble();
89                  Rxml.readNext();
90                  Rxml.readNext();
91                  scene->setItemType(DiagramItem::
                        DiagramType(1),itemMenu);
92                  scene->insertOpenItem(x_output,
                        y_output,1, name);
93              }
94          }
95          if (RE2::FullMatch(Rxml.name().toString().
                toAscii().data(), "AND-(\\d+)")) //And
96          {
97              Rxml.readNext();
98              Rxml.readNext();
99              if (Rxml.isStartElement())
100             {
```

```
101                     QString name = Rxml.readElementText
                            ();
102                     Rxml.readNext();
103                     Rxml.readNext();
104                     Rxml.readElementText();
105                     Rxml.readNext();
106                     Rxml.readNext();
107                     qreal x_and = Rxml.readElementText()
                            .toDouble();
108                     Rxml.readNext();
109                     Rxml.readNext();
110                     qreal y_and = Rxml.readElementText()
                            .toDouble();
111                     Rxml.readNext();
112                     Rxml.readNext();
113                     QList<QList<int> > customList ;
114                     while (RE2::FullMatch(Rxml.name().
                            toString().toAscii().data(), "
                            custom_(\\d+)")) {
115                         QList<int> customData;
116                         int row, column, output, size;
117                         RE2::FullMatch(Rxml.
                                readElementText().toAscii().
                                data(),"(\\d+):(\\d+)=(\\d+)
                                -(\\d)",&row,&column,&output
                                ,&size);
118                         customData<<row<<column<<output
                                <<size;
119                         customList.append(customData);
120                         Rxml.readNext();
121                         Rxml.readNext();
122                     }
123                     scene->setItemType(DiagramItem::
                            DiagramType(2),andItemMenu);
124                     scene->insertOpenAndItem(x_and,y_and
                            ,customList, name);
125                 }
126             }
127             if (RE2::FullMatch(Rxml.name().toString().
                    toAscii().data(), "OR-(\\d+)")) //And
128             {
129                 Rxml.readNext();
130                 Rxml.readNext();
131                 if (Rxml.isStartElement())
132                 {
133                     QString name = Rxml.readElementText
                            ();
134                     Rxml.readNext();
135                     Rxml.readNext();
```

```
136                     Rxml.readElementText();
137                     Rxml.readNext();
138                     Rxml.readNext();
139                     qreal x_and = Rxml.readElementText()
                            .toDouble();
140                     Rxml.readNext();
141                     Rxml.readNext();
142                     qreal y_and = Rxml.readElementText()
                            .toDouble();
143                     Rxml.readNext();
144                     Rxml.readNext();
145                     QList<QList<int> > customList ;
146                     while (RE2::FullMatch(Rxml.name().
                            toString().toAscii().data(), "
                            custom_(\\d+)")) {
147                         QList<int> customData;
148                         int row, column, output, size;
149                         RE2::FullMatch(Rxml.
                                readElementText().toAscii().
                                data(),"(\\d+):(\\d+)=(\\d+)
                                -(\\d)",&row,&column,&output
                                ,&size);
150                         customData<<row<<column<<output
                                <<size;
151                         customList.append(customData);
152                         Rxml.readNext();
153                         Rxml.readNext();
154                     }
155                     scene->setItemType(DiagramItem::
                            DiagramType(3),orItemMenu);
156                     scene->insertOpenOrItem(x_and,y_and,
                            customList, name);
157                 }
158             }
159             if (RE2::FullMatch(Rxml.name().toString().
                    toAscii().data(), "PRIORITY-(\\d+)")) //
                    And
160             {
161                 Rxml.readNext();
162                 Rxml.readNext();
163                 if (Rxml.isStartElement())
164                 {
165                     QString name = Rxml.readElementText
                            ();
166                     Rxml.readNext();
167                     Rxml.readNext();
168                     Rxml.readElementText();
169                     Rxml.readNext();
170                     Rxml.readNext();
```

```
171                     QString priorityItem = Rxml.
                            readElementText();
172                     Rxml.readNext();
173                     Rxml.readNext();
174                     qreal x_and = Rxml.readElementText()
                            .toDouble();
175                     Rxml.readNext();
176                     Rxml.readNext();
177                     qreal y_and = Rxml.readElementText()
                            .toDouble();
178                     Rxml.readNext();
179                     Rxml.readNext();
180                     scene->setItemType(DiagramItem::
                            DiagramType(4),priorityItemMenu);
181                     scene->insertOpenPriorityItem(x_and,
                            y_and, name, priorityItem);
182                 }
183             }
184             if (RE2::FullMatch(Rxml.name().toString().
                    toAscii().data(), "VOTING-(\\d+)")) //And
185             {
186                 Rxml.readNext();
187                 Rxml.readNext();
188                 if (Rxml.isStartElement())
189                 {
190                     QString name = Rxml.readElementText
                            ();
191                     Rxml.readNext();
192                     Rxml.readNext();
193                     Rxml.readElementText();
194                     Rxml.readNext();
195                     Rxml.readNext();
196                     qreal x_and = Rxml.readElementText()
                            .toDouble();
197                     Rxml.readNext();
198                     Rxml.readNext();
199                     qreal y_and = Rxml.readElementText()
                            .toDouble();
200                     Rxml.readNext();
201                     Rxml.readNext();
202                     scene->setItemType(DiagramItem::
                            DiagramType(5),itemMenu);
203                     scene->insertOpenItem(x_and,y_and,5,
                             name);
204                 }
205             }
206         }
207         Rxml.readNext();
208     }
```

```
209         file.close();
210         writeInStatusBar("XML Opened", Qt::blue);
211 }
```

Per completezza, il codice che segue rappresenta la conversione in XML dell'architettura in figura E.1:

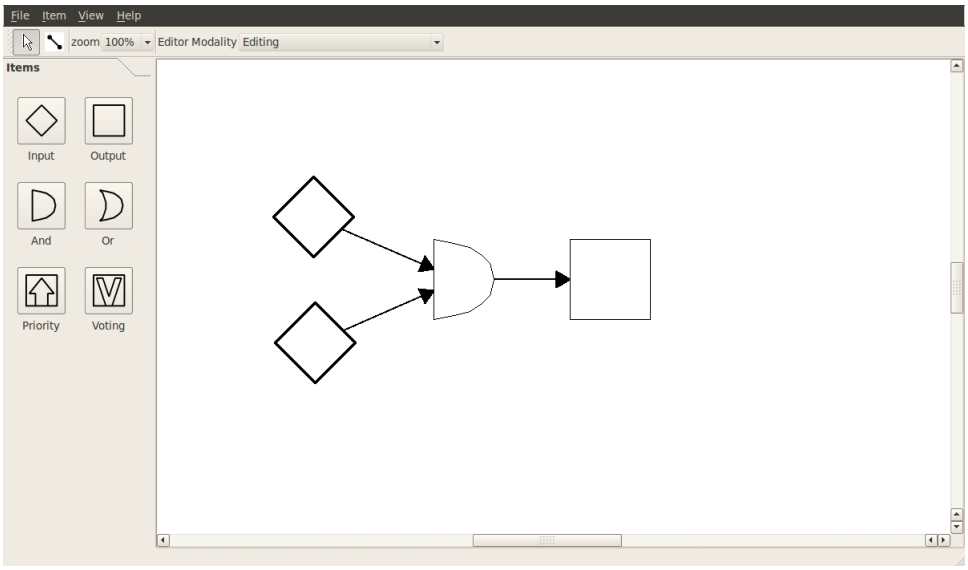

Figura E.1: Screenshot con il disegno di un'architettura esemplificativa

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Diagram>
3      <OUTPUT-0>
4          <Name>OUTPUT-26896</Name>
5          <x>2570</x>
6          <y>2449</y>
7      </OUTPUT-0>
8      <INPUT-1>
9          <Name>INPUT-58358</Name>
10         <Type>Hand Geometry</Type>
11         <Security>12</Security>
12         <Output>2</Output>
13         <InputNumber>2</InputNumber>
14         <Position>1</Position>
15         <repeatOn>1000</repeatOn>
16         <x>2202</x>
17         <y>2528</y>
18     </INPUT-1>
19     <INPUT-2>
20         <Name>INPUT-5567</Name>
```

```
21          <Type>Face</Type>
22          <Security>74</Security>
23          <Output>2</Output>
24          <InputNumber>2</InputNumber>
25          <Position>0</Position>
26          <repeatOn>1000</repeatOn>
27          <x>2200</x>
28          <y>2371</y>
29      </INPUT-2>
30      <AND-3>
31          <Name>AND-52574</Name>
32          <Level>0</Level>
33          <x>2380</x>
34          <y>2449</y>
35      </AND-3>
36      <ARROW-4>
37          <Type>arrow</Type>
38          <startItem>AND-3</startItem>
39          <endItem>OUTPUT-0</endItem>
40      </ARROW-4>
41      <ARROW-5>
42          <Type>arrow</Type>
43          <startItem>INPUT-1</startItem>
44          <endItem>AND-3</endItem>
45      </ARROW-5>
46      <ARROW-6>
47          <Type>arrow</Type>
48          <startItem>INPUT-2</startItem>
49          <endItem>AND-3</endItem>
50      </ARROW-6>
51      <input>
52          <Name>INPUT-58358</Name>
53          <Type>Hand Geometry</Type>
54          <Output>2</Output>
55          <Security>12</Security>
56          <Timeline>1</Timeline>
57          <repeatOn>1000</repeatOn>
58      </input>
59      <input>
60          <Name>INPUT-5567</Name>
61          <Type>Face</Type>
62          <Output>2</Output>
63          <Security>74</Security>
64          <Timeline>0</Timeline>
65          <repeatOn>1000</repeatOn>
66      </input>
67      <and>
68          <Name>AND-52574</Name>
69          <Level>0</Level>
```

```
70              <input_0>INPUT-5567</input_0>
71              <input_1>INPUT-58358</input_1>
72          </and>
73  </Diagram>
```