Università degli studi di Padova

Facoltà di Ingegneria

Tesi di Laurea Magistrale in

Ingegneria Informatica

Sistemi di Autenticazione Multibiometrica: strumenti a supporto della progettazione

Relatore
Prof. Carlo Ferrari

Candidato
Francesco Locascio

Anno Accademico 2010/2011

A mio nonno Francesco,

con la speranza di averti reso orgoglioso

Sommario

Negli ultimi anni, l'autenticazione biometrica ha goduto di considerevoli miglioramenti, specialmente in materia di accuratezza ed affidabilità, con alcuni tratti 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 realizzazione di un insieme di strumenti software che, prendendo in considerazione le molteplici scelte realizzative, supportano la progettazione di un sistema di autenticazione multibiometrico, nel contesto di un progetto più ampio che prevede anche la realizzazione di strumenti per il deployment dell'architettura di sistema e l'esecuzione della stessa [64].

Contents

Chapter 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 allorché 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 l'implementazione di un insieme di strumenti software a supporto della progettazione di un sistema di autenticazione

multibiometrico, nel contesto di un progetto più ampio che prevede anche la realizzazione di strumenti per il deployment dell'architettura di sistema e l'esecuzione della stessa [64].

Il progetto può essere collocato nell'area di ricerca che in letteratura prende il nome di "Identity Management"[1] Nel nostro caso abbiamo utilizzato la biometria come uno strumento per la memorizzazione e la verifica dell'identità, utilizzando complessi algoritmi biometrici.

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 vari sottoprocessi, al fine di non compromettere l'accuratezza complessiva del sistema.

1.3 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, allorché 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*? [1] 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;

[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.

- *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, per 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 [2] 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.

1.4 Sistema Biometrico

Un *sistema biometrico* [3] 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 allorché 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:

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;

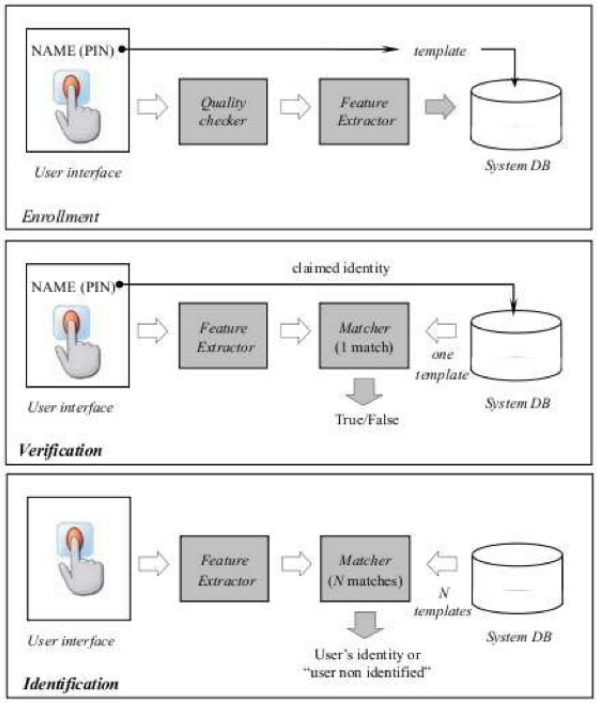

Fig. 1.1: Procedure di enrollment, verifica e identificazione

- 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.

1.5 Metriche e Prestazioni

Valutare in maniera opportuna un sistema di identificazione biometrico è un campo di ricerca ancora aperto. Le prestazioni complessive di un 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 [4].

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 [5].

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

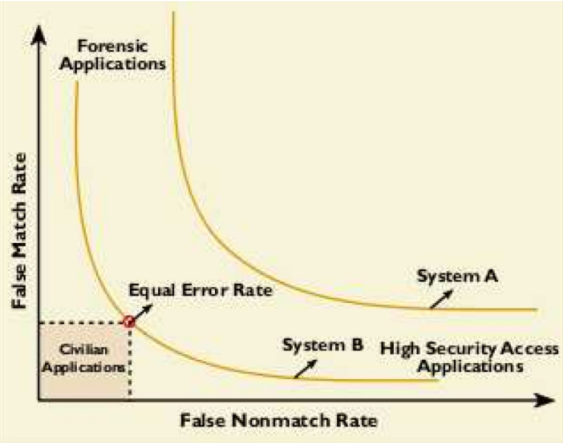

Fig. 1.2: Receiver Operating Characteristics (ROC) di un sistema.

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 sulla 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 pressoché 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.

1.6 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 [6].

1.6.1 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.

1.6.2 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.

1.6.3 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 allorché 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 si aspettano miglioramenti significativi entro breve tempo.

1.6.4 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.

1.6.5 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.

1.6.6 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 perché 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

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.

Fig. 1.3: Confronto tra parametri biometrici

digitali come la forma più alta di autenticazione, la biometria della voce rimpiazzerà o migliorerà l'utilizzo di PIN, password, o nomi di account.

1.7 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).

1.7.1 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.

1.7.2 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), op-

pure interferire con la raccolta del dato stesso (per esempio il rumore di sottofondo quando si sta acquisendo una voce).

1.7.3 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.

1.7.4 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.

1.7.5 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.

1.7.6 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.

1.7.7 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.

Chapter 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 affetto 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 (false non-match).

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. È in realtà possibile, 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. È 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.

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 [7].

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 allorché 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 allorché 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 [8].

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 profili destro e sinistro allo scopo di considerare le possibili variazioni

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

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

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 consid-

erevole 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 [9].

Come suggerito dalla letteratura (riferimenti in [10] e [11]), 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.

2.4.1 Fusione a livello di estrazione delle feature

In questa architettura [12], l'informazione estratta dai diversi sensori viene codificata in un feature vector congiunto, che viene poi confrontato con

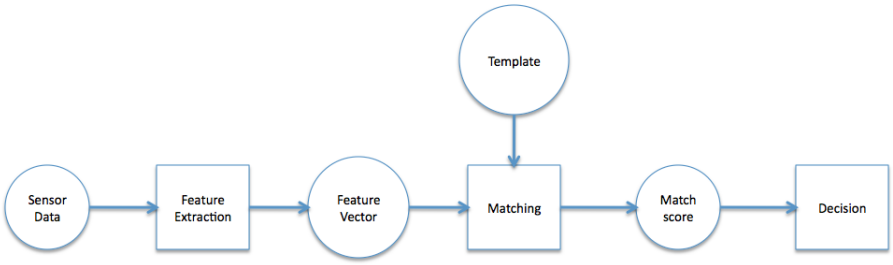

Fig. 2.1: Flusso del processo di autenticazione

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).

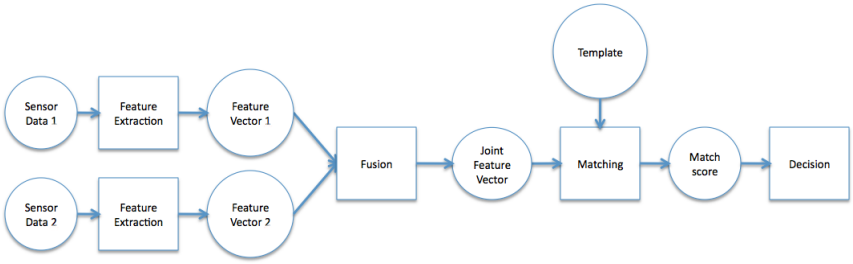

Fig. 2.2: Fusione a livello di estrazione delle feature

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 [13].

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

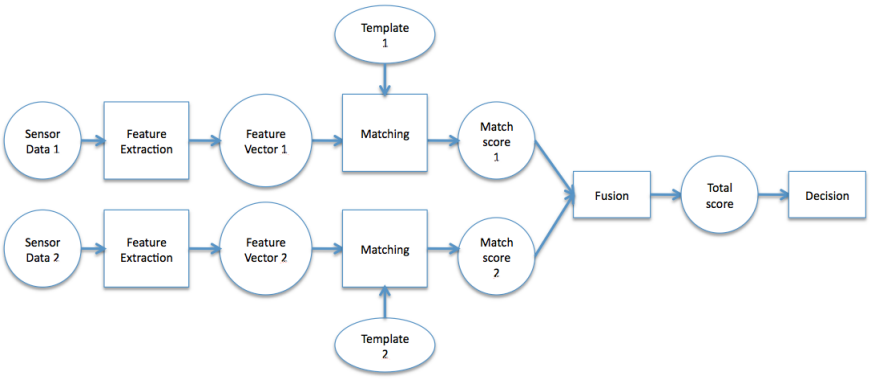

Fig. 2.3: Fusione a livello di matching score

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 de-

cisione 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;

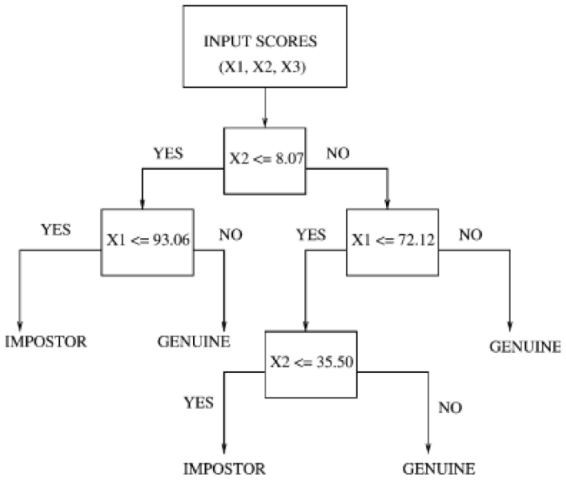

Fig. 2.4: Esempio di decision tree operante su algoritmo C5

- 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.

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.

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

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 [14], 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

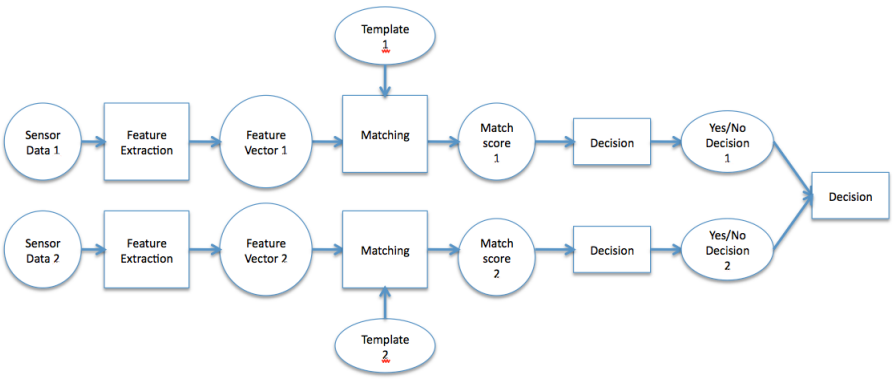

Fig. 2.5: Fusione a livello di decisione

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* [15]. 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.

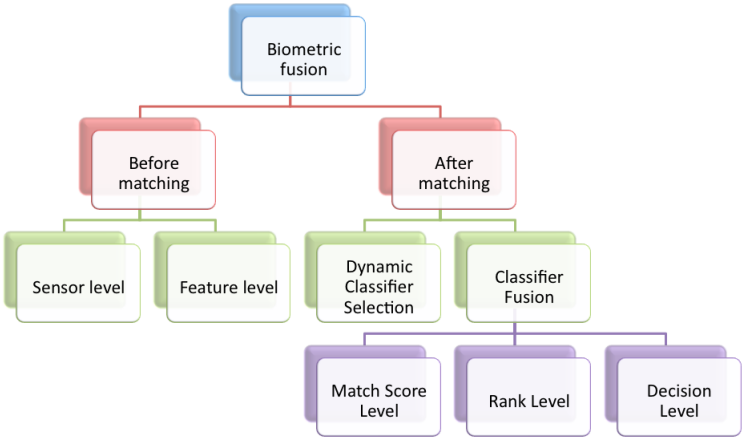

Fig. 2.6: Tipologie di fusione in un sistema multibiometrico

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 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 [16].

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 metaricerca 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 normaliz-

zazione 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 massimo 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 allorché 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 [17], qui tralasciata perché 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.

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). È 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 [18].

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 è piuttosto ampia e in questo paragrafo cercheremo di riassumere le ricerche più rilevanti.

In generale, la combinazione di diversi sistemi basati su pattern recognition è stata studiata in [19]; in applicazioni correlate ad audio-video speech processing [20], [21], [22]; in speech recognition - alcuni esempi di approcci sono: multi-band [23], multi-stream [24], [25], front-end multi-feature [26] e union model [27]; in maniera assemblata [28]; in audio-video person authentication [29]; e in multibiometria [30], [31], [32], [33]. Uno dei primi lavori indirizzati alla fusione multibiometrica risale al 1978 [34], quindi la sua storia risale ad oltre trent'anni fa. Recenti studi hanno focalizzato l'attenzione sulla qualità della fusione dei risultati [35], [36], [37], [38], [39], 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 [40], [41], l'iride [42], il volto [43], la voce [44], la firma [45] e misure classifier-dipendent (confidenza) [46], [47]. 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 influenzano 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" [35], nel classificatore polinomiale ridotto [48], nel quality-controlled support vector [36] e nella regola fissa di fusione quality-based [49]. In ruolo secondario, le misure di qualità sono spesso correlate con degli output prima di essere fornite al classificatore, come accade nella regressione logistica [37] e nei classificatori Gaussiani e Bayesiani [38].

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 [50] e la ricerca di Poh sulla qualità state-dependent [51]. Il lavoro in [48] 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 [52], [53]. Tale approccio è co-

munemente utilizzato nella verifica del parlante [54] 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 [55], l'affidabilità stimata per ogni modalità biometrica è stata usata per combinare decisioni symbolic-level ([56], [57], [58] e [47]), dove erano state considerate fusioni score-level. Tuttavia, in [56], [57], [58], il termine "failure prediction" è stato usato. Tale informazione, proveniente esclusivamente da output noti, è risultata essere efficace per ogni singola modalità biometrica [56], attraverso la fusione di sensori per una singola modalità biometrica [57], e attraverso differenti tecniche di apprendimento automatico [58]. In [47], 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 [59], 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 [60], 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. È interessante notare che la combinazione dei tre migliori parametri è solo migliore marginalmente rispetto alla migliore

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%

Table 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%

Table 2.2: Risultati della fusione (Configurazione 1)

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 [37]. 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 $\otimes$ 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.

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.

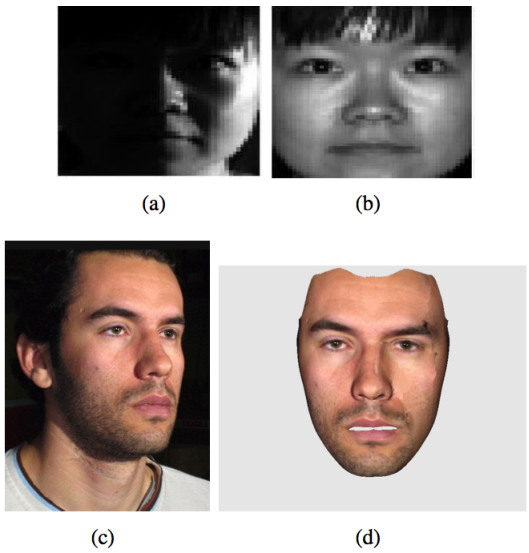

Fig. 2.7: (a) esempio di illuminazione laterale e frontale (b); (c) posa fuori piano e (d) posa corretta usando un modello 3D.

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.

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 [61]. 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

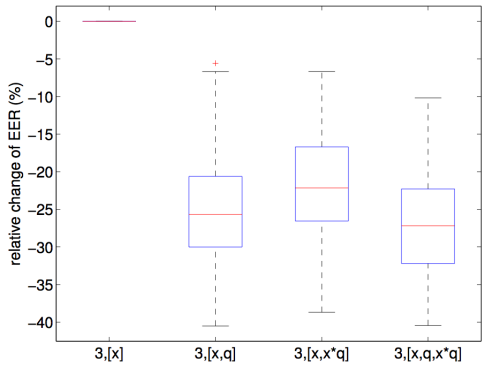

Fig. 2.8: Variazioni relative di *a posteriori EER* (%) di tutte le possibili combinazioni di output di sistema implementate usando la regressione logistica.

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 [62]. In uno studio separato [63], 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 [61], è una generalizzazione di [63] alla fusione multimodale. Un esempio di curva cost-sensitive è mostrata in figura 2.9.

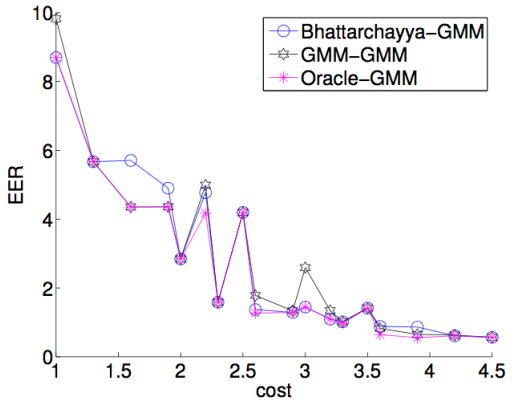

Fig. 2.9: Cost-sensitive performance curve, ottenuta usando un classificatore Bayesiano

Questo esperimento è stato condotto su Biosecure DS2 quality-based fusion[4]. Seguendo l'ordine di esecuzione dell'esperimento, il database di

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

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 Bhattarcharyya, e due misure di errore empiriche di EER basate su classificatori bayesiani, ovvero, Gaussian Mixture Model come stimatore di densità (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" perché 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 e oltre rank-ten è necessario per GMM.

Chapter 3

Progettazione della suite applicativa

Contents

Dopo aver descritto il funzionamento di un sistema di autenticazione multibiometrico, in questo capitolo verranno presentate le modalità di implementazione del software riportando, quando opportuno, alcuni stralci di codice che possano aiutare a comprenderne meglio la struttura. Il progetto è stato interamente sviluppato in linguaggio C++, sfruttando l'ambiente messo a disposizione da Qt SDK (Figura 3.1, Appendice A). Le fasi di sviluppo e testing sono state condotte su un singolo computer con sistema operativo Linux[1].

[1] distribuzione Ubuntu 10.04

3.1 Introduzione al progetto

Il progetto nasce dalla necessità di avere un tool completo di strumenti per la creazione flessibile di architetture per l'autenticazione multibiometrica, 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 fornisca ad un progettista la possibilità di modellare in modo flessibile un'architettura di autenticazione biometrica;
- Uno strumento software che consenta al reparto tecnico il deployment dell'architettura 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 implementino alcuni algoritmi per l'estrazione e la verifica di feature biometriche.

Il progetto può essere collocato nell'area di ricerca che in letteratura prende il nome di "Identity Management"[2] Nel nostro caso abbiamo utilizzato la biometria come uno strumento, infatti, per la memorizzazione e la verifica dell'identità dell'utente vengono utilizzati complessi algoritmi biometrici.

La progettazione e l'implementazione del software ha seguito un approccio modulare. Tale approccio ha come conseguenza una maggiore riutilizzabilità del codice e la possibilità di aggiungere o togliere funzionalità agli strumenti sviluppati in maniera più veloce e semplice. In particolare, l'implementazione degli algoritmi di riconoscimento che vengono usati nella maggior parte 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 (Appendice B).

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 una soglia di sicurezza in modo da permettere delle autenticazioni più restrittive o, in alternativa, delle autenticazioni più permissive;

[2]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.

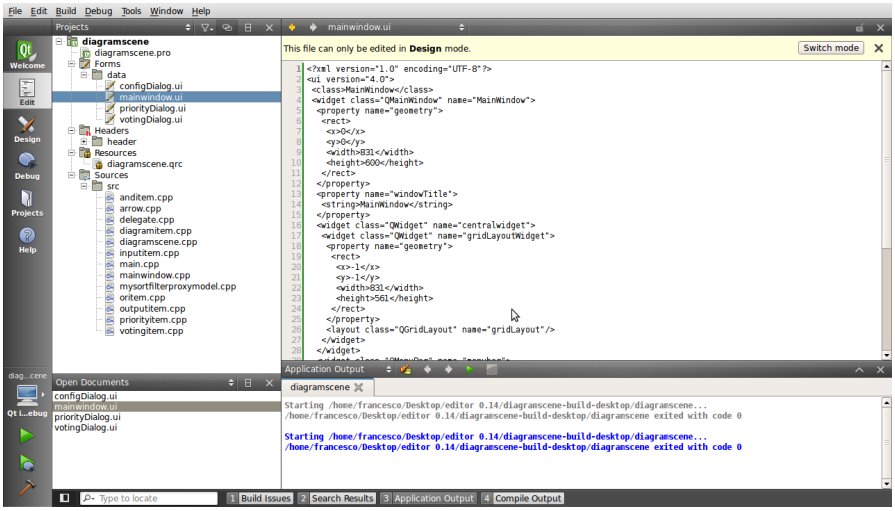

Fig. 3.1: Screenshot dell'ambiente di sviluppo

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

Nel seguenti paragrafi verranno presentati requisiti e obiettivi specifici dei tool sviluppati. Si ricorda che nell'appendice C vengono elencati i requisiti software necessari per la compilazione del codice e l'esecuzione dello stesso.

3.1.1 Motivazioni della suite di supporto alla progettazione

La progettazione di un sistema multibiometrico richiede di considerare quattro punti fondamentali:

1. **Scelta delle biometrie**;
2. **Scelta dell'architettura**;
3. **Scelta di una misura di affidabilità**;
4. **Scelta del metodo di fusione dei risultati**.

Per ciascuno dei precedenti punti, in fase di progettazione, emergono delle criticità che possono essere superate attraverso l'analisi delle prestazioni delle diverse scelte progettuali; la suite oggetto di questa tesi cerca di offrire al progettista del sistema una serie di strumenti integrati che consentono di supportare la progettazione, garantendo:

- *affidabilità*;
- *flessibilità*;

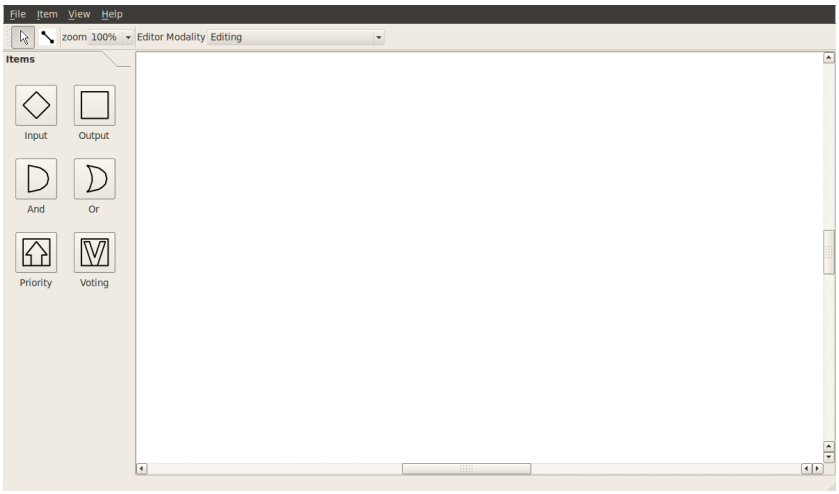

Fig. 3.2: Screenshot dell'interfaccia grafica

- *scalabilità*;
- *semplicità d'uso* (interfaccia user-friendly).

Una volta definita l'architettura di sistema, considerando le criticità di cui sopra, il software la traduce in un file XML, lasciando agli strumenti a valle la responsabilità del deployment [64].

3.2 Descrizione della suite applicativa

L'interfaccia grafica (figura 3.2) della suite si presenta divisa in quattro aree fondamentali:

- *barra dei menu*, in alto;
- *barra degli item* (a sinistra) comprendente Input, Output, AND, OR, Voting Rule e Priority Rule;
- *area di disegno*, tavola bianca al centro della schermata;
- *status bar e proprietà degli oggetti* (a comparsa sulla destra, su richiesta del progettista).

Ciascuna area presenta delle funzionalità che analizzeremo nel dettaglio. La barra dei menu comprende le funzionalità fondamentali del software ed è composta dai sottomenu:

- *File*: comprende i comandi per aprire, chiudere e salvare un progetto;
- *Item*: include i comandi per visualizzare le proprietà dell'oggetto selezionato;

- *View*: consente di visualizzare la barra di stato (cronologia degli eventi) e le proprietà degli oggetti;
- *Help*.

3.2.1 Item

Con il termine *item* definiamo, all'interno dell'applicazione, qualsiasi tipo di oggetto che può essere inserito nello schema dell'architettura definita dall'utente. Come mostrato nell'interfaccia, esempi di item sono gli input, gli operatori per combinare gli input e l'oggetto output (che deve essere inserito obbligatoriamente, per consentire il salvataggio dell'architettura). Ciascun item gode di alcune proprietà caratteristiche, possiede variabili non condivise, ha un proprio livello nell'architettura, un nome univoco all'interno dell'architettura e una propria classe distinta; ogni classe che caratterizza un oggetto estende la classe *DiagramItem*. L'inserimento di un oggetto nello schema *DiagramScene* avviene tramite il seguente metodo (nello stralcio, consideriamo soltanto l'inserimento di una porta logica AND).

```
1  void DiagramScene::mousePressEvent(QGraphicsSceneMouseEvent *
       mouseEvent)
2  {
3      selected = items(mouseEvent->scenePos());
4    {
5                  [ ... ]
6        }
7      if (mouseEvent->button() == Qt::LeftButton) {
8          InputItem *inputItem;
9          OutputItem *outputItem;
10         AndItem *andItem;
11         OrItem *orItem;
12         PriorityItem *priorityItem;
13         VotingItem *votingItem;
14         switch (myMode) {
15         case InsertItem:
16             if (myItemType == 0) { //Input
17                 nInput++;
18                 inputItem = new InputItem(myItemType, myItemMenu);
19                 inputItem->setBrush(myItemColor);
20                 addItem(inputItem);
21                 inputItem->setPos(mouseEvent->scenePos());
22                 emit itemInserted(inputItem);
23                 int pos=0;
24                 foreach (QGraphicsItem *item, this->items(Qt::
                       AscendingOrder)) {
25                     if (item->type() ==  InputItem::Type) {
26                         InputItem *tmpInput = qgraphicsitem_cast<
                               InputItem *>(item);
27                         tmpInput->setMaxTimeline(nInput,pos);
28                         pos++;
29                     }
30             [ ... ]
```

Quando l'oggetto viene inserito nella GraphicScene, è possibile applicare tutti i metodi della classe fra cui QGraphicsScene::items (Qt::SortOrder order) che

ritorna una lista ordinata di tutti gli item che fanno parte della scena e che sarà utile per la gestione degli stessi.

La classe *DiagramScene* estende *QGraphicsScene* ed implementa lo scenario sul quale verranno posti gli item caratterizzanti l'architettura di sistema. Particolare rilevanza per la robustezza del software ha il metodo **bool** DiagramScene::validateDiagram() che si occupa di approntare tutte le verifiche necessarie alla costruzione di un'architettura di sistema valida; ad esempio, vengono effettuati controlli sulla presenza di un blocco output, sulla validità degli input che non devono avere collegamenti in ingresso e così via.

Proprietà degli oggetti

La parte più complessa circa lo sviluppo del software riguarda certamente quella della gestione delle proprietà degli oggetti e la configurazione dei parametri di uscita.

Quest'ultima sarà oggetto di una descrizione dettagliata nei prossimi paragrafi, mentre per quanto concerne le proprietà degli oggetti bisogna distinguere fra proprietà degli input e degli operatori. Per ciascun oggetto input distinguiamo:

- *Type*: riferito al tipo di input che vogliamo rappresentare (ad esempio, Face, Iris, ecc., tenendo conto della possibilità di poter inserire più algoritmi per lo stesso parametro biometrico);
- *Security*: consente di impostare una soglia oltre il quale un output di dubbia valutazione, venga definito true o false;
- *Output*: permette di settare il numero di uscite del parametro biometrico scelto;
- *Timeline*: caratterizza l'ordine di esecuzione degli input in un sistema non distribuito, ovvero quando le acquisizioni vengono fatte su un unico client. Se il sistema è distribuito, tale caratteristica potrebbe perdere senso.
- *Used on*: tale opzione prevede di attivare opportune architetture in maniera condizionata rispetto all'output. Ciò nasce dall'esigenza di poter attivare solo alcuni input per effettuare nuovamente delle verifiche di autenticazione, in condizioni di esiti false, maybe true, maybe false.

3.2.2 Graphic Scene

Come già accennato nella sezione precedente, la parte della suite che consente la maggior interazione con l'utente finale è quella che presenta lo

scenario di rappresentazione dell'architettura di sistema. Nello spazio dedicato al disegno vero e proprio dell'architettura è possibile inserire gli item e i relativi collegamenti, nonché trascinare, modificare ed eliminare tutto ciò che è stato inserito. Per inserire un item è sufficiente cliccare sulla relativa icona e rilasciare l'oggetto dove si preferisce in qualsiasi punto dello spazio di disegno. È possibile salvare un'architettura soltanto quando questa risulta valida, secondo i criteri descritti in precedenza. A seconda delle preferenze dell'utente è possibile vedere la cronologia degli eventi (inserimenti degli oggetti, cancellazioni, ...), nonché le proprietà degli item inseriti; la barra degli strumenti offre la possibilità di visualizzare la *Editor Modality*, ovvero consente all'utente di avere una visione globale delle modalità di funzionamento dell'architettura a seconda della sua configurazione. La classe che implementa lo scenario è ancora DiagramScene che comprende i metodi per gestire le azioni all'interno dello spazio, nonché la verifica di validità dello schema. In Figura 3.3, è possibile vedere l'interfaccia durante il disegno di un'architettura di sistema esemplificativa: l'utente ha collegato due input tramite una porta logica AND. Si noti, che nella parte destra della figura compaiono le proprietà dell'input selezionato: viene specificato il tipo, la soglia di sicurezza adottata per la valutazione, il numero di possibili output, l'ordine cronologico d'attivazione e il tipo di utilizzo nella modalità di funzionamento del sistema.

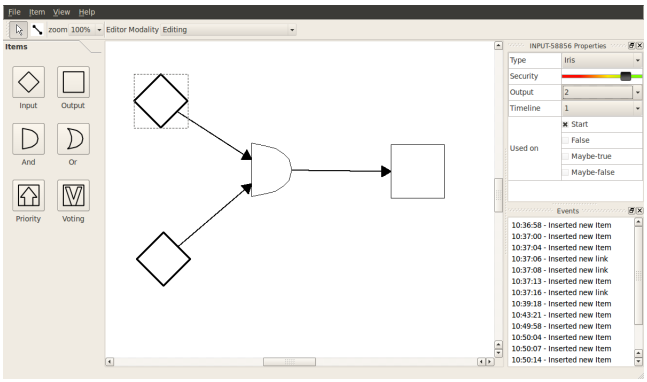

Fig. 3.3: Screenshot con il disegno di un'architettura esemplificativa

3.3 Enrollment Tool

Questo strumento ha lo scopo di fornire una interfaccia grafica con cui l'utente[3] possa inserire e registrare facilmente un nuovo utente nel database dell'organizzazione e dare a tale utente i diritti di cui necessita.

Lo strumento di *iris enrollment* deve quindi soddisfare i seguenti requisiti:

- semplicità d'uso e sicurezza;
- implementare funzionalità efficienti in termini di tempo, esecuzione e spazio utilizzato;
- gestione multiutente;
- verificare la robustezza per le feature estratte.

Lo strumento di *iris enrollment* 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 dell'iride estratte per ogni utente;

3.4 Aspetti implementativi

Dopo aver descritto gli elementi caratteristici della suite applicativa, in questa sezione discuteremo circa gli aspetti implementativi più importanti che caratterizzano il software sviluppato. In particolare, verranno analizzati l'algoritmo di riconoscimento dell'iride utilizzato e le criticità emerse durante la progettazione del software, fra le quali:

- realizzazione del tool di *Iris Enrollment*;
- implementazione del *database*;
- verifiche di *validità dello schema architetturale* prodotto;
- come impostare delle *soglie di sicurezza*, necessarie all'autenticazione finale;
- implementare la possibilità di avere output multipli e relative tabelle di verità;

[3] 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

- *salvare l'architettura* in un formato facilmente interpretabile dall'applicativo di deployment a valle.

Ciascuno dei precedenti aspetti critici viene risolto in maniera efficace secondo le soluzioni riportate nei rispettivi paragrafi che seguono.

3.4.1 Iris 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 screenshot del tool, il primo (figura 3.4) mostra la schermata relativa alla selezione di un utente e alla gestione del database e e la seconda 3.5 relativa al training dell'utente.

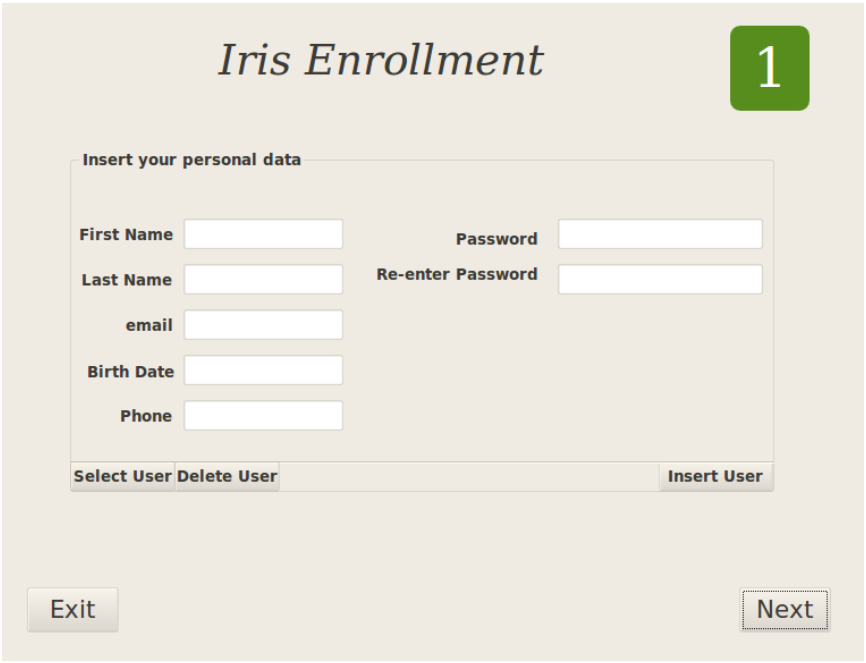

Fig. 3.4: Prima schermata del tool di training di immagini dell'iride relativa alla selezione di un utente e alla gestione del database

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

Prima Schermata

La prima schermata consente, a chi deve effettuare l'enrollment degli utenti, di:

Fig. 3.5: Seconda schermata del tool di enrollment relativa al training dell'utente

- poter selezionare un utente già presente nel db, cliccando sul pulsante "Select User" comparirà un box, come mostrato in figura 3.6, 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 dell'iride.

Seconda Schermata

La seconda schermata ha lo scopo di effettuare il training di immagini dell'iride. Come si può vedere nella figura 3.4, a sinistra compare un box in cui viene mostrata l'immagine dell'iride opportunamente caricata dall'utente. Tale immagine viene fornita in input all'algoritmo *Iris Recognition* che registrerà le feature dell'iride selezionata nel database. Se l'utente risulta già presente nel database viene restituito un messaggio di errore. Una volta terminata l'operazione di inserimento delle feature nel database,

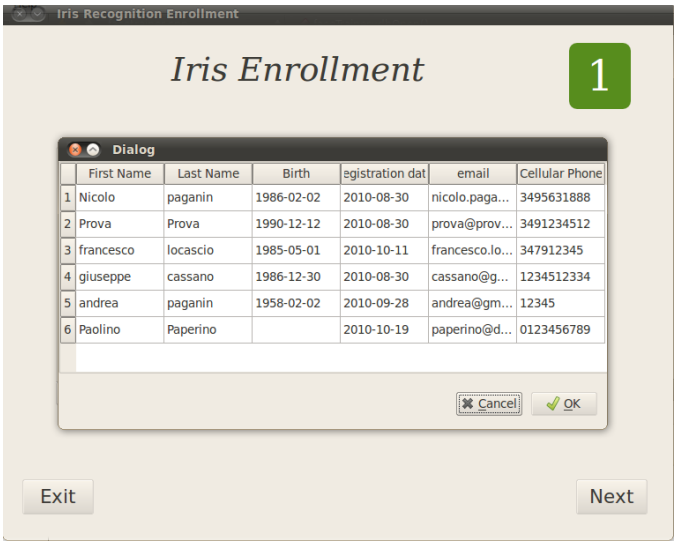

Fig. 3.6: Box per la selezione di un utente già presente nel database

è possibile chiudere il programma mediante l'apposito pulsate, in basso a destra.

3.4.2 Iris Recognition Algorithm

Fra gli altri algoritmi utilizzati nel progetto del sistema di autenticazione, quello per il riconoscimento dell'iride riveste particolare rilevanza ed è oggetto di trattazione in questa tesi. L'algoritmo che verrà descritto è stato modificato ed integrato opportunamente nelle varie componenti del sistema, ma il sorgente originale è disponibile sul sito `http://projectiris.co.uk`, sotto GNU General Public Licence (GPL).

Quella che seguirà sarà una descrizione del funzionamento dell'algoritmo e delle sue caratteristiche principali.

Per ottenere il riconoscimento automatico dell'iride devono essere perseguiti tre obiettivi: localizzare l'iride in un immagine, codificarla in un formato adeguato al calcolo ed alla computazione, e rendere memorizzabili i dati al fine di essere caricati e confrontati.

Iris Location

Quando viene localizzata un iride vi sono due potenziali opzioni. Il software può richiedere all'utente di selezionare i punti nell'immagine, che è la soluzione più realizzabile e garantisce la maggiore accuratezza, ma non può essere plausibile per le tutte le applicazioni del mondo reale. L'altra opzione è quella di sfruttare l'auto-detect software. Questo processo è computazionalmente complesso e introduce un po' di errore relativamente all'interpretazione

della macchina. Tuttavia, poiché il software dovrà pertanto trovare interazione con l'utente meno esperto si tratta di un passo importante verso la produzione di un sistema che è adatto per la distribuzione nel mondo reale, e quindi è diventato una priorità il riconoscimento automatico dell'iride.

L'individuazione dell'iride non è un compito banale, in quanto la sua intensità è vicina a quella della pupilla ed è spesso oscurata da ciglia e dalle palpebre. Tuttavia, la pupilla per la sua forma regolare e uniforme tonalità scura, è relativamente facile da individuare. La pupilla e l'iride possono essere approssimati come concentrici e questo fornisce un affidabile punto di partenza per il rilevamento automatico.

Pupilla

L'intensità della pupilla e la posizione sono abbastanza costanti nella maggior parte delle immagini e quindi si presta bene per il rilevamento automatico. La rilevazione della pupilla può essere effettuata mediante le seguenti operazioni: riduzione del rumore tramite l'applicazione di una sfocatura (median filter), ricerca della soglia per ottenere la pupilla, rilevamento dei bordi per ottenere il confine della pupilla e l'identificazione dei cerchi.

Un median filter è basato su un filtro di convoluzione che offusca l'immagine impostando un valore in pixel medio con quello dei suoi vicini. Tale approccio è basato sull'ordinamento e ha complessità $\mathcal{O}(nlogn)$ in media. Tuttavia, è necessario trovare una soluzione più efficiente per il software.

Per un algoritmo migliore, è stato consultato il paper di Perreault [65], che descrive come creare un median filter che agisca in tempo lineare. Il processo prevede la costruzione di istogrammi singola colonna e la loro combinazione a formare istogrammi centrati attorno a un pixel, conosciuto come un istogramma kernel.

L'effetto complessivo del filtro, riduce il rumore senza perturbare la fedeltà del bordo dell'immagine originale. Questo si traduce in un raggruppamento più forte dei pixel nell'istogramma risultante e un'analisi dinamica delle caratteristiche che occupano discreti range di pixel, come la pupilla (figura 3.7).

Fig. 3.7: Istogramma generato dall'applicazione del median filter su un'immagine del db.

Successivamente all'applicazione di un filtro median, il risultato ottenuto su un'immagine è apprezzabile in figura 3.8. La localizzazione della pupilla è ben definita e il rilevamento del bordo può estrarre infomazioni utili per

formare un'unica immagine di profondità in bit, rintracciabile per cerchi.

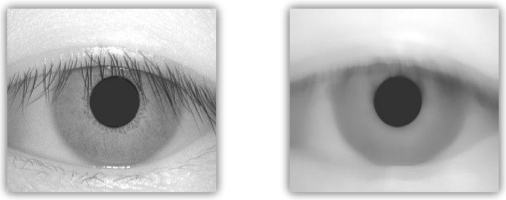

Fig. 3.8: Immagine prima e dopo l'applicazione di un median filter.

Il filtro Sobel è una tecnica di rilevamento dei bordi che calcola i gradienti (in entrambe le direzioni x e y) dell'intensità dell'immagine per ogni pixel e poi li combina per dare un gradiente d'intensità. Questo indica quanto forte è il confine, indicazione di presenza del bordo.

Come si può vedere dalla figura 3.9, il risultato del processo di filtraggio è un artefatto circolare che indica l'estremità della pupilla nell'immagine. Questi dati richiedono ulteriori analisi per ottenere una rappresentazione "best-fit" della pupilla potenzialmente ellittica.

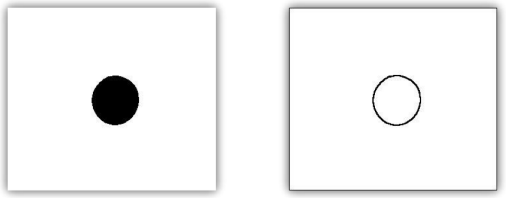

Fig. 3.9: Immagine prima e dopo l'applicazione di un 2D Sobel convolution filter.

La trasformata di Hough è implementata per localizzare la pupilla (e conseguentemente l'iride). Data una curva regolare, in questo caso un cerchio, e un sufficiente range di parametri (coordinate del centro e raggio) la trasformata di Hough generalizzata può considerare i singoli pixel di un'immagine per il loro contributo alla soluzione globale coerente con tutti i parametri entro il range valido. A causa della elevata complessità computazionale della trasformata di Hough (lo spazio dei parametri per un semplice cerchio è in tre dimensioni e può significare crescita dei calcoli in proporzioni di 10^6), una serie di elementi sono stati semplificati per accelerare i calcoli.

Effettuare il thresholding dell'immagine in ingresso ha il vantaggio intrinseco di cambiare la profondità virtuale a 1 (nero o bianco). Inoltre, l'algoritmo implementato consente di risparmiare tempo operando sui pixel neri - una realizzazione di scala di grigi aumenta in modo significativo il tempo di elaborazione. Inoltre, una ricerca sulla soglia fornisce stime fiduciose sui parametri di centro e raggio, che possono essere utilizzati per vincolare e guidare l'algoritmo di Hough.

Come si può vedere in figura 3.10 questo processo fornisce soluzioni rapide, per rappresentazioni con soglia evidenziata e bordo di pupilla rilevato. La base del processo di auto-detection dell'iride si basa su quanto appena descritto.

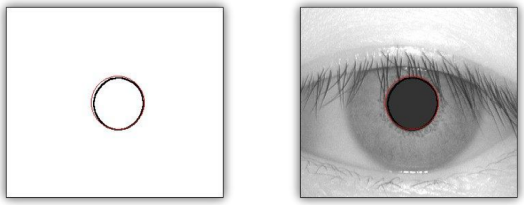

Fig. 3.10: Trasformata di Hough per un cerchio (rosso), data la rilevazione del bordo della pupilla.

Iride

Una volta individuata la posizione della pupilla, la complessità di localizzare l'iride è qualche volta ridotta data la relativa concentricità della pupilla rispetto all'iride. In contrasto con il rilevamento della pupilla, un modo efficiente per localizzare l'iride risulta più complicato a causa dell'ostruzione delle palpebre, la sua trama irregolare e la relativa somiglianza di iride e sclera.

Tuttavia, un metodo simile a quello del rilevamento della pupilla ottiene buoni risultati anche in questo caso. Ancora una volta viene applicato il median filter all'immagine originale per rimuovere il rumore dalle immagini e per rafforzare il raggruppamento di pixel nell'istogramma. L'istogramma risultante viene utilizzato per identificare un punto che può essere utilizzato per la soglia dell'immagine, al fine di ottenere un'immagine simile a quella di figura 3.11. Il metodo consiste nel trovare i picchi degli istogrammi (es-

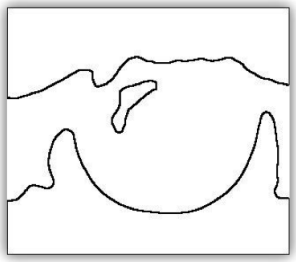

Fig. 3.11: Esempio di immagine ottenuta tramite therholding intelligente dell'input originale, usando filtri di Sobel e trasformata di Hough.

cludendo il picco più a sinistra che rappresenta la pupilla) e nella scelta di un valore di soglia tra questi punti. Anche se non sempre ottimale, questo

metodo crea generalmente una immagine con la quale ottere una corrispondenza approssimativa per l'iride, tramite la trasformata di Hough.

La trasformata di Hough è molto efficiente per trovare l'iride da un'immagine come questa, perché è resistente al rumore e si comporta bene anche quando una grande quantità del cerchio è nascosto. Per esempio, quando due palpebre coprono una larga porzione dell'iride, come in figura 3.12. Questo fornisce un cerchio che definisce l'iride, ma parte di questa è ancora oscurata dalle palpebre; ciò implica la necessità di definirle.

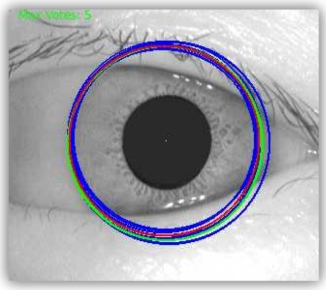

Fig. 3.12: Una dimostrazione dei risultati ottenuti dalla applicazione della trasformata di Hough su un'immagine come quella in figura 3.11. Il cerchio scelto è mostrato in rosso, che è una media di tutti i cerchi con il numero massimo di voti (in verde). I cerchi blu rappresentano quelli con il secondo maggior numero di voti.

Palpebre

Prendiamo atto che l'immagine di un occhio ha circa tre intensità, dal più scuro al più chiaro: pupilla, iride e sclera, e le palpebre. Quindi un metodo di thresholding idoneo dovrebbe essere in grado di separare le palpebre dal resto dell'immagine. Otsu offre un metodo in [66] per il thresholding di immagini in scala di grigi.

Assumendo di avere un immagine con L possibili intensità per ogni pixel e sia n_i il numero di pixel di intensità i. Inizialmente, normalizziamo l'istogramma in una distribuzione di probabilità, quindi $p_i = n_i / \sum_{i=1}^{L} n_i$. Supponiamo di dividere la soglia dell'immagine al livello k in due classi, C_0 che contiene livelli minori o uguali a k e C_1 che contiene il resto. Successivamente, definiamo:

$$\omega(k) := \sum_{i=1}^{k} p_i \quad e \quad \mu(k) := \sum_{i=1}^{k} i p_i,$$

che rappresentano rispettivamente la probabilità di prendere casualmente un pixel in C_0 e la relativa intensità attesa.

È chiaro che $1 - \omega(k)$ rappresenta la probabilità di pescare un pixel in C_1 e $\mu(L)$ è l'intensità attesa dell'intera immagine. Possiamo così calcolare:

$$\sigma^2 = \frac{[\mu(L)\omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]},$$

che rappresenta la varianza tra le classi C_0 e C_1 di Otsu in [66]. Calcoliamo questa varianza fra classi per ogni k e infine la soglia su k quando σ^2 è massimizzata.

Per ottenere tutte le informazioni utili dalla metodologia di Otsu, dobbiamo prima ignorare la pupilla per assicurare che il risultato non consista solamente di due classi (la pupilla ed ogni altra cosa). Una volta che abbiamo ignorato la pupilla nel calcolo delle classi, ricaviamo due classi dalla metodologia Otsu: una contenente l'iride, sclera e pupilla (in quanto è più bassa l'intensità), e l'altra classe che contiene le palpebre (figura 3.13). Adesso che abbiamo localizzato l'iride è necessario iniziare ad estrarre informazione da essa.

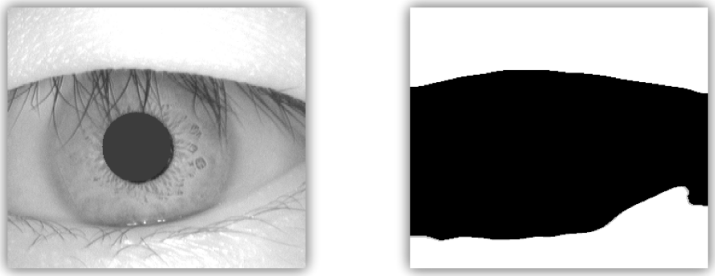

Fig. 3.13: Immagine prima e dopo l'applicazione di median filter e Otsu threshold.

Codifica dell'iride

L'iride è codificata da un unico insieme di 2048 bit, utilizzati per l'identificazione della persona. Questi codici possono essere memorizzati in un database e confrontati per l'identificazione univoca della persona. La dimensione di 2048 è sufficiente per memorizzare i dati di diversi filtri sulla maggior parte dell'iride, mentre è sufficientemente piccola per essere memorizzata in un database ed essere manipolata in fretta. La fase di estrazione dei dati dall'iride, non influenzata dalla deformazione della pupilla. Usiamo il filtro di Gabor per estrarre le informazioni in questa fase, come suggerito da Daugman [67].

Il filtro di Gabor è un'applicazione della Gabor wavelet (equazione (3.1)). Questo restituirà due bit a seconda del quadrante (figura 3.14) che conterrà il numero immaginario normalizzato. Questa equazione può essere semplificata per la computazione considerandola come combinazione di due filtri: un

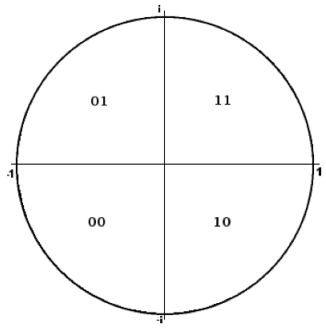

Fig. 3.14: I quattro quadranti e le corrispondenti sequenze di bit che li rappresentano.

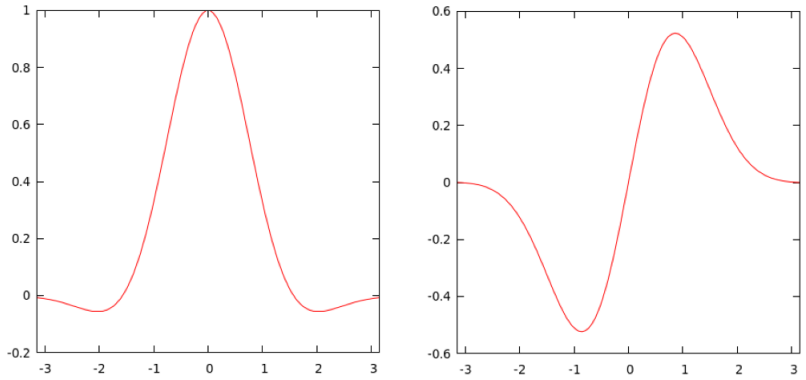

Fig. 3.15: A sinistra la parte reale e, a destra, la parte immaginaria della Gabor wavelets.

filtro rappresenta la parte reale dell'integrale e l'altro rappresenta la parte immaginaria. Ognuno di questi filtri consiste nella combinazione di un filtro Gaussiano e di uno sinusoidale (si veda la figura 3.15). I parametri α, β, e ω sono settati vincolando il filtro su un range di deviazione standard e il filtro sinusoidale nel range tra $-\pi$ e π .

$$h_{\{Re,Im\}} = sgn_{\{Re,Im\}} \int_{\rho}\int_{\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} e^{-\frac{(r_0-\rho)^2}{\alpha^2}} e^{-\frac{(\theta_0-\phi)^2}{\beta^2}} \rho d\rho d\phi \quad (3.1)$$

Filtro di Localizzazione e Dimensione

I filtri di Gabor devono essere posti in modo da tener conto di un grande range di informazioni, senza perdere dati relativi all'iride. Come per i due bit utilizzati per identificare univocamente ogni quadrante, abbiamo un ulteriore vincolo che consiste nell'utilizzare 1024 filtri, permettendo la gestione

di bit code di lunghezza 256B per ogni iride, che possono essere suddivise in lunghezze 4B. Per semplificare i calcoli coinvolti nell'applicazione dei filtri all'iride, possono essere considerati tutti posizionati ad una rotazione uniforme di 0.

Dal momento che vi è una quantità consistente di informazioni in punti diversi nella direzione angolare nell'iride, è anche una buona idea mettere i filtri in modo uniforme in questa direzione. In questa implementazione sono state considerate 256 direzioni angolari perché permette di dividere in maniera uniforme i filtri e comprende una grande diffusione di dati attraverso l'iride. Tuttavia, in molti casi, in direzione radiale ci sono maggiori informazioni osservando vicino alla pupilla.

Per includere i dati più importanti vicino alla pupilla, i filtri possono essere posizionati in modo che una percentuale maggiore di loro abbiano i centri in questo intervallo. Con le 256 direzioni angolari, ci sono 4 filtri in posizione radiale a sinistra. Questi possono essere collocati in modo uniforme tra il fondo dell'iride ed a metà strada, per consentire l'esplorazione dei dati intorno alla pupilla. Infine, per assicurare che i filtri siano sufficientemente grandi per rappresentare a sufficienza i dati, il centro del filtro non può essere posto sui tre pixel più vicini alle parti superiore e inferiore dell'immagine. Questo implica che l'uso di filtri di dimensioni 1 e 3 non è mai incluso in codifica dell'iride, ed eviteremo di includere frammenti di pupilla che possono essere inseriti nella sezione dell'iride. Il posizionamento finale dell'insieme di filtri per un'iride è rappresentato in figura 3.16. Una considerazione finale

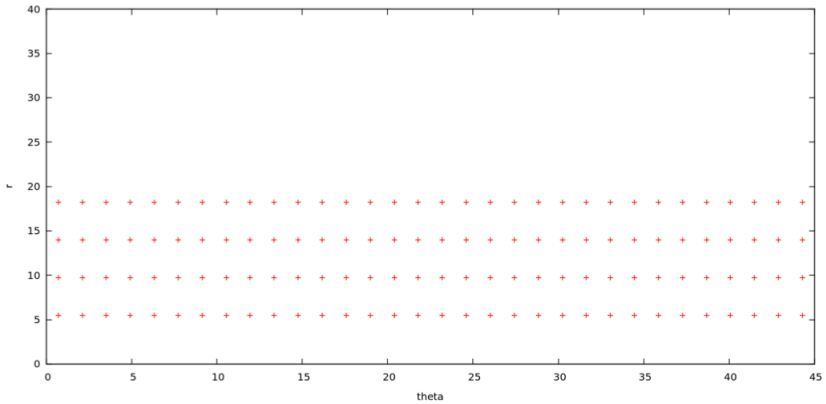

Fig. 3.16: Posizione dei filtri per un'iride di raggio 40 tra $\theta = 0$ e $\theta = 45$.

sui filtri può essere fatta sulle loro dimensioni. Due visioni schematiche sono rappresentate in figura 3.17. Il primo (a sinistra) mira a posizionare un filtro più grande possibile per ogni posizionamento. Questo significa soddisfare uno dei requisiti principali: acquisire più informazioni possibile. Questo sistema, però soffre di una problematica relativa alla scarsa rappresentazione dei dettagli al centro dell'iride, a causa del posizionamento dei centri dei

filtri in quelle zone. Il secondo schema (a destra), affronta questo problema modificando la posizione dei centri per i filtri a dimensione massima. Questa dimensione può essere impostata nell'implementazione per avere più bitcode distinti, per questo risulta preferibile ed è attualmente usata nel nostro algoritmo.

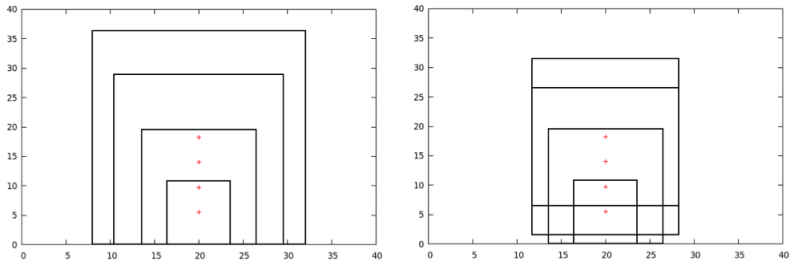

Fig. 3.17: Due schemi per le dimensioni ed il piazzamento dei filtri in punti fissati nella direzione angolare di un'iride di raggio 40. A sinistra, piazzamento di filtri a massima dimensione; a destra, filtri che coprono i centri con dimensione 25.

Confronti

Le codifiche dell'iride possono essere confrontate calcolando la distanza di Hamming fra esse. La distanza di Hamming tra due codifiche viene calcolata come mostrato in [67]. Assumendo di avere due codifiche, $codeA$ e $codeB$, con le corrispondenti bit mask, $maskA$ e $maskB$ (dove uno 0 nella bit mask corrisponde ad un bit errato in quella posizione nella codifica); calcoliamo la distanza di Hamming H come segue:

$$H = \frac{\| \, (codeA \otimes codeB) \cap maskA \cap maskB \, \|}{\| \, maskA \cap maskB \, \|},$$

dove $\otimes$ rappresenta l'operatore logico XOR e $\cap$ rappresenta l'operatore AND tra bit. Possiamo vedere che il numeratore sarà il numero di differenze tra mutui bit non-bad del $codeA$ e $codeB$, mentre il denominatore rappresenta il numero di mutui bit non-bad. Il lettore attento può osservare che questa misura può fallire se c'è un numero eccessivamente basso di mutui bit non-bad, perché porterebbe il numeratore a zero nonostante ci siano grandi differenze tra $codeA$ e $codeB$. Comunque, è un caso semplice da gestire e che si propone raramente nel confronto fra immagini di iride.

Il confronto viene eseguiti tramite vari shift nel codice, per controllare la rotazione dell'immagine, in quanto è molto probabile che l'occhio di un soggetto sia ruotato se la fotocamera o la loro testa è leggermente inclinata.

Il test di indipendenza statistica è la chiave per il riconoscimento dell'iride. Esso fornisce una sufficiente probabilità di fallire se gli iris bitcode di due

occhi diversi vengono confrontati, ed è superato unicamente quando un'iride è confrontata con sé stessa.

Iris Database

Dato che la codifica e la maschera per un'iride vengono generate direttamente, solo queste due sono da memorizzare nel database. L'algoritmo nella versione originale memorizzava tali contenuti in un semplice file di testo, mentre per i nostri scopi è stato necessario utilizzare una tabella dedicata, descritta nelle sezioni seguenti.

3.4.3 Database

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

Nella prima tabella creata 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	
output	varchar(40)	NO		NULL	
authFlag	int(11)	NO		NULL	

I parametri biometrici, relativi all'iride, vengono memorizzati in una tabella con la seguente struttura:

Field	Type	Null	Key	Default	Extra
irisID	int	NO	PRI	NULL	auto_increment
userId	char	NO		NULL	REF. users(idusers)
irisFeatures	varchar(5000)	NO		NULL	
fileName	char	NO		NULL	
acquisitionDate	date	NO		NULL	
acquisitionTime	time	NO		NULL	

3.4.4 Modulo di controllo della validità dello schema

L'attività di progettazione di qualsiasi sistema deve prevedere un'attività di verifica e validazione; la progettazione delle architetture dei sistemi di autenticazione multibiometrica non è esente da tale vincolo, perciò è stato necessario prevedere un modulo di validazione dell'architettura. Il modulo di controllo, fra gli altri compiti, provvede a:

- verificare la presenza di un blocco output;
- il blocco di input non può avere frecce entranti in esso;
- ogni item deve essere collegato a qualcosa, prima che l'architettura venga salvata;
- ogni architettura deve prevedere l'utilizzo dei blocchi di input, blocchi di fusione risultati e blocco di output.

L'utente non è comunque vincolato a seguire particolari architetture, ma ha assoluta libertà di scelta purché si disegnino sistemi validi.

Lo stralcio di codice che segue riporta la serie completa di condizioni (prendendo in considerazione il solo operatore AND) che deve avere un'architettura valida.

```
1  bool DiagramScene::validateDiagram() {
2      setLevel(); //set the level for every item in the scene
3    [ ... ]
4      foreach (QGraphicsItem *item, allItems) {
5          if (item->type() == InputItem::Type) { //if one input item
               exist
6              inputItem= qgraphicsitem_cast<InputItem *>(item);
7              if (inputItem->countArrow()==0) {
8                  isInput=false; //if every input item has one arrow
9              }
10             else timeline.append(inputItem->getTimelinePos());
11         }
12         if (item->type() == OutputItem::Type) { //if one output
               item exist
13             thisItem= qgraphicsitem_cast<OutputItem *>(item);
14             if (thisItem->countArrow()==0) { //if every output
                   item has one arrow
15                 isOutput=false;
16             }
17         }
18         if ((item->type() == AndItem::Type) || (item->type() ==
               OrItem::Type) || (item->type() == PriorityItem::Type)
               || (item->type() == VotingItem::Type)) { //if one and
               item exist
19             if (item->type() == AndItem::Type) {
20                 thisItem= qgraphicsitem_cast<AndItem *>(item);
21                 if (thisItem->countArrow()==0) { //if every and
                       item has one arrow
22                     isOperation=false;
23                 }
24                 [ ... ]
25     }
26     if (isInput && isOutput && isOperation) {
27         qSort(timeline);
28         bool isDistinct = true;
29         for (int i=0; i<timeline.size(); i++) {
30             if (timeline.at(i)!=i) {
31                 isDistinct = false;
32             }
33         }
34         if (isDistinct) isValid = true;
35     }
36     return isValid;
37 }
38 [ ... ]
39 void MainWindow::saveFile() {
40
41     if (!scene->validateDiagram()) {
42         writeInStatusBar("(E) INVALID Diagram! Impossible to save."
               , Qt::red);
43         return;
44     }
```

Si noti che nella riga 2 del precedente codice viene invocato il metodo setLevel: ad ogni item viene assegnato un livello di rappresentazione al fine di facilitare la verifica dello schema e la traduzione in gerarchia XML. Il salvataggio, che sarà comunque approfondito successivamente, è impossibile se l'architettura non risulti valida (vedasi riga 42 del precedente codice).

3.4.5 Soglie di sicurezza

L' autenticazione basata su caratteristiche (o biometrica) è legata agli attributi biometrici dell'utente, che possono essere di due tipi:

- caratteristiche fisiche, ad esempio impronte digitali o della retina;
- caratteristiche comportamentali, ad esempio firma o timbro di voce.

Questo tipo di tecnica richiede una fase iniziale chiamata enrollment phase, che consiste nella misurazione ripetuta della proprietà di interesse e nella definizione di un template. Terminata questa fase, il processo di autenticazione consiste nella misurazione della caratteristica e nel confronto con il template: se questi corrispondono entro un certo intervallo di tolleranza, all'utente viene dato l'accesso. Una soglia di tolleranza si rende necessaria perché non ci si può aspettare un uguaglianza perfetta, tuttavia la sua scelta deve essere estremamente studiata, in modo da massimizzare i successi e minimizzare gli insuccessi.

Nel software implementato, è stato tenuto conto di questo aspetto, consentendo al progettista di settare il parametro di soglia per ogni caratteristica biometrica, in maniera intuitiva e semplificata tramite un apposito slider (Figura 3.18).

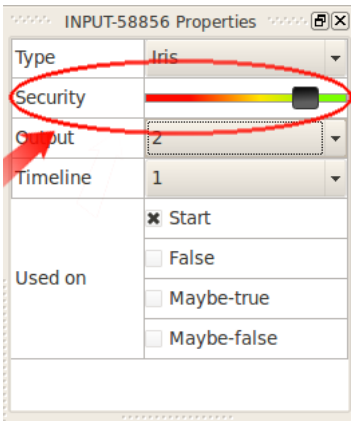

Fig. 3.18: In rosso, lo slider per settare la soglia di sicurezza

In particolare, nella nostra implementazione, per un input di tipo "face" il valore di soglia viene settato a 45 (valore intermedio su un intervallo [1, 90]); come si evince dallo stralcio di codice seguente:

```
1 InputItem::InputItem(DiagramItem::DiagramType diagramType, QMenu *
      contextMenu)
2     : DiagramItem(diagramType, contextMenu)
3 {
4     //Default value
5     inputType = "Face";
```

```
6       securityValue=45;
7       [ ... ]
8  void InputItem::securitySliderMove(int value){
9         securityValue=value;
10        securitySlider->setSliderPosition(value);
11 }
12      [ ... ]
13 int InputItem::getSecurity(){
14        return securityValue;
15 }
```

3.4.6 Configurazioni di output

Nei paragrafi precedenti, abbiamo visto che nella descrizione delle proprietà degli item era possibile dichiarare il loro numero di output (2, 3 o 5). Chiaramente, quando questi vengono collegati tramite un qualsiasi operatore sarà necessario combinare opportunamente le uscite e decidere il comportamento del sistema.

Una procedura di assegnazione semiautomatica degli output viene fatta dal software sviluppato, ma viene lasciato ampio spazio al progettista per le personalizzazioni del caso. Per accedere a tale possibilità l'utente, cliccando col pulsante destro sull'operatore di combinazione dei risultati, occorre selezionare la voce "Configure". Nell'esempio in figura 3.19, è possibile visualizzare una tabella *multivalued* generata dal sistema (dettagli in figura 3.20) per un operatore che ha in ingresso due input (es. Iris e Face), rispettivamente con 5 e 3 output. L'uscita di tale operatore dovrà considerare tutte le possibili configurazioni di output, ovvero 15 possibilità ($3 \times 5 = 15$); osservando la figura 3.20 si noti che, alcune configurazioni sono state già definite dal software tramite composizione automatica, altre invece risultano vuote e lasciano spazio alle specifiche del progettista. Anche le combinazioni predefinite sono modificabili. Possibili sviluppi futuri del sistema potranno prevedere un modulo di verifica delle configurazioni di output, perché attualmente rimane esclusiva responsabilità del progettista. Nella suite sviluppata, risulta particolarmente comoda la possibilità di modificare tabelle multivalued piuttosto ampie sfruttando i campi per la ricerca messi a disposizione (nella parte bassa, in figura 3.20).

L'implementazione di tale funzionalità ha reso necessario l'utilizzo della modalità di programmazione *Model View*. Qt 4 introduce un nuovo insieme di classi View che usa come architetture model/view per gestire la relazione tra dati e modi in cui vengono presentati all'utente. La separazione di funzionalità introdotta da questa architettura fornisce allo sviluppatore grande flessibilità per la personalizzazione della presentazione degli item, e fornisce un modello di interfaccia standard per consentire l'utilizzo di dati provenienti da viste definite. Model-View-Controller (MVC) consiste di tre tipi di oggetti: il Modello è l'oggetto dell'applicazione, la vista è la presentazione a schermo e il Controller definisce i modi con cui l'interfaccia utente reagisce

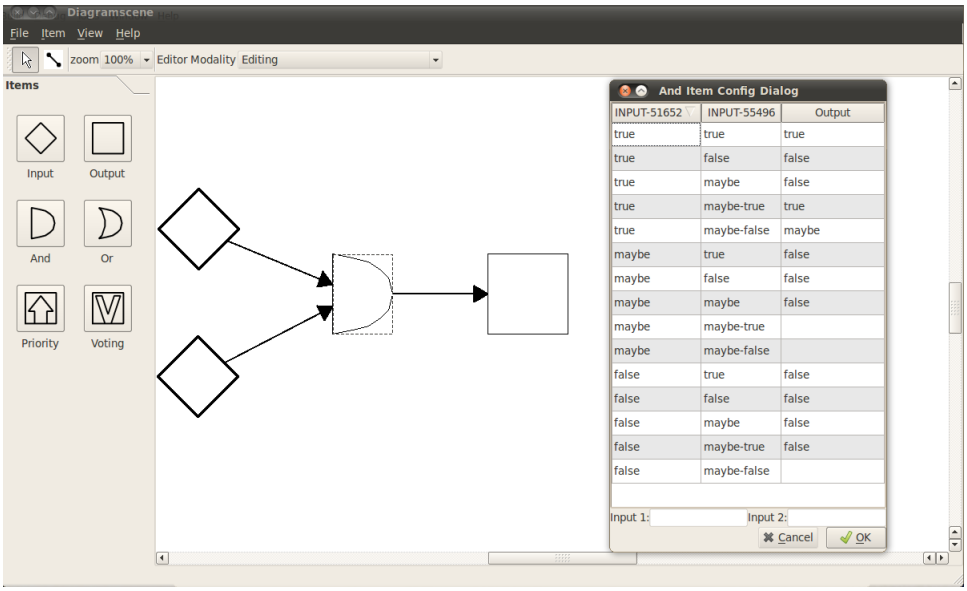

Fig. 3.19: Tabella di configurazione dell'uscita per l'AND item rappresentato.

agli input dell'utente. Prima di MVC, l'interfaccia utente tendeva a raggruppare questi tre elementi; il loro disaccoppiamento permette di aumentare la flessibilità e il loro riutilizzo.

Se gli oggetti View e Controller sono combinati, quello che si ottiene è l'architettura model/view. Quest'ultima, separa ancora il modo in cui i dati vengono memorizzati dal modo in cui vengono presentati all'utente, ma fornisce framework semplificato basato sugli stessi principi. La separazione permette di visualizzare gli stessi dati secondo diverse view, implementare nuove view, senza modificare le strutture dati sottostanti. Per consentire una manipolazione semplice dei dati, viene introdotto il concetto di *delegate*. Un delegate fornisce diverse possibilità per personalizzare, modificare e presentare i dati. Osservando la figura 3.21, il *modello* comunica con una sorgente di dati (tipicamente, la base di dati del sistema) fornendo una *interfaccia* per le altre componenti dell'architettura. La natura della comunicazione dipende dal tipo di dati sorgente e dai modi in cui il modello è stato implementato. La *view* ottiene il "modello indicizzato" dal modello; questi sono riferimenti ai dati. Avendo gli "indici modello", la view può recuperare i dati dalla sorgente. Nello standard delle view, il delegate presenta gli oggetti di informazione. Quando un oggetto viene modificato, il delegate comunica direttamente con il modello, utilizzando il modello indicizzato. In generale, l'architettura model/view può essere divisa nei 3 gruppi visti in precedenza: modello, viste e delegati. Ognuno di queste componenti è definito da classi astratte che forniscono interfacce comuni e, in qualche caso, implementazioni di caratteristiche di default. Modelli, viste e delegati comunicano sfruttando

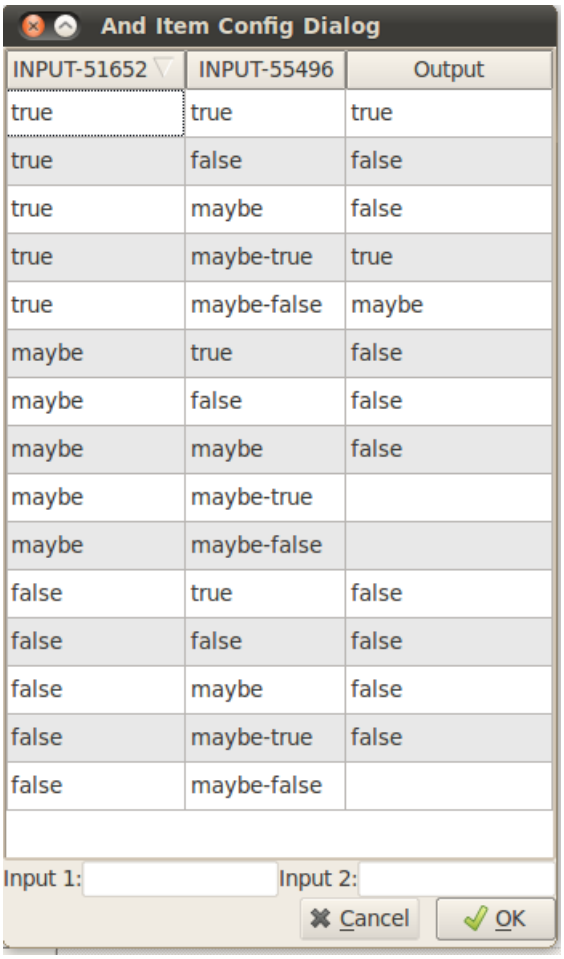

Fig. 3.20: Particolare ingrandito della tabella di configurazione dell'uscita per l'AND item rappresentato.

segnali e slot:

- segnali dal modello informano la view sul cambiamenti dei dati alla sorgente;
- segnali dalla view forniscono informazioni circa l'interazione dell'utente con gli oggetti che gli vengono presentati;
- segnali dal delegate sono usati durante le modifiche per comunicare, a modello e view, circa lo stato dell'editor.

Nel nostro caso, il modello è stato creato sfruttando la classe *QStandardItemModel* che fornisce strumenti per la gestione di strutture di oggetti generici, ognuno dei quali può contenere dati arbitrari. Le viste sono delle tabelle raffiguranti le configurazioni, come in figura 3.19, e sono state implementate sfruttando la classe *QTableView*. Gli oggetti visualizzati in una table

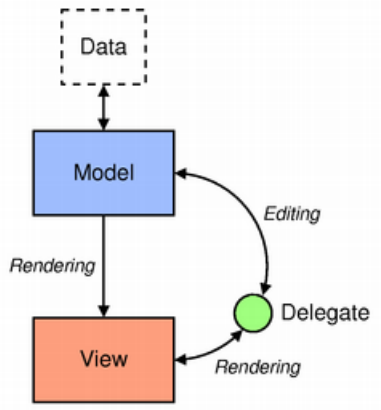

Fig. 3.21: Architettura *model/view*.

view, come per un generico item view, vengono presentati e modificati utilizzando delegati standard. Infatti, le regole predefinite delle configurazioni possono essere modificate sfruttando la classe *Delegate*; per quest'ultima, assumono particolare importanza i metodi setEditorData(QWidget *editor, **const** QModelIndex &index) e setModelData(QWidget *editor, QAbstractItemModel *model, **const** QModelIndex &index): quando si desidera cambiare le regole predefinite per l'output, l'apertura della lista nella tabella mostra le varie possibilità e crea un oggetto di tipo QComboBox: il metodo setModelData(...) lo mette dentro al modello e le modifiche dell'utente vengono memorizzate in una lista dedicata (customList) per non sovrascrivere le configurazioni predefinite.

La possibilità di effettuare ricerche all'interno delle tabelle di configurazione viene resa possibile dalla classe *MySorterFilterProxyModel* che, implementando un proxy tra modello e vista, filtra (usando le espressioni regolari e le primitive della classe QRegExp) gli item nella dialog box di configurazione per gli item. Anche se è possibile fare operazioni di ordinamento e filtraggio con le funzioni interne delle View, questo approccio non consente di condividere fra più viste i risultati di operazioni potenzialmente costose. Per risolvere questo problema, si è deciso di adottare il modello proxy previsto dal framework model/view per gestire le informazioni fornite tra i singoli modelli e le viste. I modelli proxy hanno lo stesso comportamento dei modelli normali, dalla prospettiva di una View e per l'accesso ai dati dai modelli sorgente. I segnali e gli slot utilizzati dal framework model/view assicurano che ogni View sia aggiornata opportunamente, senza tenere conto di come i proxy diano posti tra la view e il modello sorgente.

3.4.7 Salvataggio e interpretazione dell'architettura

Il salvataggio dell'architettura è uno degli aspetti cruciali del software sviluppato poiché implica le seguenti problematiche:

- registrazione di un'architettura valida;
- lettura/interpretazione e trasportabilità efficiente del file prodotto per il software di deployment a valle.

Il primo punto, viene risolto invocando scene->validateDiagram() secondo le politiche decritte nei paragrafi precedenti. Il problema della registrazione dell'architettura su un file consiste nella sua interpretabilità: infatti, per motivi tecnici, risulta molto più efficiente e preciso un sistema che si basa sulla lettura e interpretazione delle stringhe di caratteri, piuttosto che sui disegni. Per questa ragione, il disegno dell'architettura deve essere convertito in qualcosa di facilmente interpretabile e processabile dal software 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 tramite una semplice 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 perché 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 3.22:

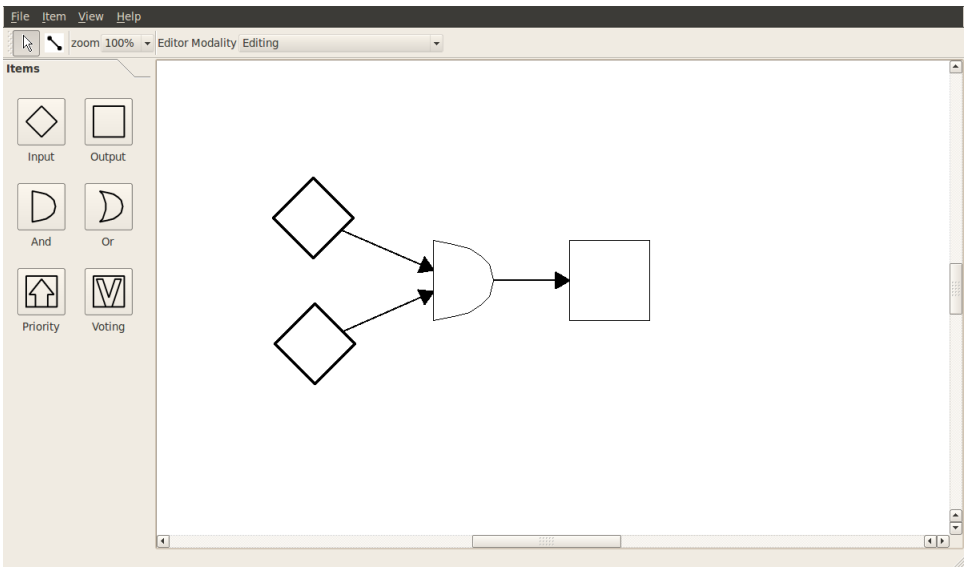

Fig. 3.22: Screenshot con il disegno di un'architettura esemplificativa

```
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>
```

3.5 Conclusioni

L'obiettivo principale del progetto di cui fa parte questa tesi era sviluppare 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, in particolare sviluppa gli obiettivi riguardanti la realizzazione degli strumenti software a supporto dell'attività di progettazione dell'architettura di un sistema di autenticazione multibiometrico. Le criticità emergenti nella progettazione, vengono semplificate dalla suite progettata; in particolare, l'interfaccia user-frendly consente di:

- scegliere facilmente le biometrie da utilizzare nel sistema e gli algoritmi da sfruttare per uno stesso parametro biometrico;
- disegnare l'architettura di sistema mediante l'inserimento di blocchi elementari o modificarne una esistente;
- scegliere delle misure di affidabilità per ogni parametro biometrico, in funzione dei requisiti progettuali;
- configurare in maniera automatica, o semi-automatica, gli output dei blocchi di sistema;
- costruire architetture flessibili, le cui prestazioni possano essere agevolmente verificate;
- salvare progetti di architetture valide e trasmetterle in esecuzione in modo efficiente.

La realizzazione del progetto ha portato all'esecuzione di una dimostrazione [64] che permette di verificarne le potenzialità, utilizzando dati reali (sono stati utilizzati un rilevatore dell'iride e una videocamera per la rilevazione del volto). Tenendo conto di alcune problematiche tecniche (dispositivo di rilevamento dell'iride non si adatta particolarmente bene alle caratteristiche dell'algoritmo di riconoscimento) e ambientali (scarsa illuminazione), il funzionamento del sistema non ha fatto rilevare particolari problemi: l'interpretazione dell'architettura, il deployment, i tempi di trasmissione dei dati e la loro sicurezza sono accettabili.

Sviluppi futuri, potranno interessare l'implementazione di nuove funzionalità di fusione dei risultati, il testing distribuito del sistema con parametri biometrici diversi da quelli descritti in precedenza e l'utilizzo di algoritmi di riconoscimento più sofisticati.

Bibliography

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

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

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

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

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

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

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

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

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

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

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

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

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

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

[23] C. Cerisara. *Contribution de l'Approache Multi-Bande à la Reconnaissance Automatic de la Parole.* Ph.D. thesis, Institute Nationale Polytéchnique de Lorraine, Nancy, France, 1999.

[24] S. Dupont. *Etude et Dévelopment de Nouveaux Paradigmes pour la Reconnaissance Robuste de la Parole.* Ph.D. thesis, Laboratoire TCTS, Universite de Mons, Belgium, 2000.

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

[26] 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.

[27] 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.

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

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

[30] N. Poh. *Multi-system Biometric Authentication: Optimal Fusion and User-Specific Information.* Ph.D. thesis, Swiss Federal Institute of Technology in Lausanne (Ecole Polytechnique Fédérale de Lausanne), 2006.

[31] K. Kryszczuk. *Classification with Class-independent Quality Information for Biometric Verification.* Ph.D. thesis, Swiss Federal Institute of Technology in Lausanne (Ecole Polytechnique Fédérale de Lausanne), 2007.

[32] J. Richiardi. *Probabilistic Models for Multi-Classifier Biometric Authentication Using Quality Measures.* Ph.D. thesis, Swiss Federal Institute of Technology in Lausanne (Ecole Polytechnique Fédérale de Lausanne), 2007.

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

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] Nicolò Paganin. *Sistemi di Autenticazione Multibiometrica: Strumenti per il Deployment Automatico in Ambito Distribuito.* Master's thesis, Padova University, December 2010.

[65] S. Perreault and P. Hebert. *Median Filtering in Constant Time.* IEEE Transactions on Image Processing, 16(9):2389-2394, 2007.

[66] N. Otsu. *A threshold selection method from gray-level histograms.* Automatica, 11:285-296, 1975.

[67] J. Daugman. *How iris recognition works.* IEEE Transactions on circuits and systems for video technology, 14(1): 21-30, 2004.

Appendix A

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 (acquistata 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. 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 utilizzando 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 proprie 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.

Appendix B

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.

B.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.

B.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 };
```

B.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.

B.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.

Appendix C

Requisti software e Risoluzione dei problemi

In questa 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 installata. Di seguito una lista con il software necessario:

- Qt;
- Librerie Intel openCv;
- openssl-client;
- openssl-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;

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 l'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 "mysql.so". Per risolvere questo problema seguire la procedura:
  1. se non è stato già fatto, passare a Qt in versione OpenSource, cliccando sul pulsante in basso a sinistra sopra la freccia verde che permette la compilazione;
  2. copiare dalla directory plugin nella cartella di installazione di Qt il file "myslq.so" facilmente recuperabile in rete.