ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea in Ingegneria Elettronica, Informatica e Telecomunicazioni

Acquisizione e Riconoscimento di Impronte Digitali Mediante Smartphone

Elaborato in Fondamenti di Elaborazione delle Immagini

Relatore:
Chiar.mo Prof. RAFFAELE CAPPELLI

Presentata da:
FRANCESCO RICCI

Co-Relatore:
Dott. MATTEO FERRARA

I Sessione
Anno Accademico 2013/2014

Indice

Introduzione

Oggigiorno il riconoscimento di impronte digitali ha un ruolo fondamentale per quanto riguarda aspetti di sicurezza, come l'accesso a determinate zone tramite porte regolate da serrature biometriche di qualunque tipologia, e aspetti forensi come può essere il riconoscimento delle impronte digitali su una scena del crimine o l'acquisizione di impronte di pregiudicati.

L'importanza delle impronte digitali è insita nel fatto che sono una caratteristica biometrica permanente (non cambia nel tempo) e capace di identificare ogni individuo in modo univoco.

Esistono diverse metodologie per acquisire le impronte digitali (inchiostro, scanner, ecc.) che portano a risultati più o meno accurati. Nel presente studio è stato affrontato il problema di effettuare l'acquisizione mediante la fotocamera di uno Smartphone. Si tratta di un'applicazione potenzialmente molto interessante in quanto l'utilizzo di uno Smartphone renderebbe molto più semplice l'acquisizione delle impronte, non essendo necessari dispositivi specifici come, ad esempio, scanner d'impronte digitali. D'altra parte, l'utilizzo di una fotocamera per l'acquisizione delle impronte introduce diverse problematiche, fra cui individuare l'area del dito corrispondente all'impronta, valutare la qualità di un'immagine e determinare quali minuzie estratte corrispondano effettivamente a quelle di interesse.

Questo lavoro di tesi è stato svolto a Cesena presso il laboratorio di Sistemi Biometrici del Dipartimento di Informatica dell'*Alma Mater Studiorum – Università di Bologna.* Il *Biometric System Laboratory* è uno dei più noti centri di ricerca, a livello nazionale e internazionale, sui sistemi biometrici. Le principali attività di ricerca ivi svolte riguardano l'acquisizione e il riconoscimento di impronte digitali, il riconoscimento facciale, la valutazione delle prestazioni di

sistemi biometrici e l'integrazione in progetti large-scale (passaporti, ID card…).

In questo lavoro è stato effettuato uno studio di fattibilità di tale modalità di acquisizione delle impronte, in particolare sono state svolte le seguenti attività:

- Implementazione di un sistema ad-hoc per l'acquisizione di un database di impronte
- Acquisizione delle impronte da un gruppo di volontari
- Prima analisi di qualità delle foto acquisite
- Estrazione delle minuzie
- Test dei risultati ottenuti

L'idea di base è quella di realizzare un'applicazione mobile in grado di scattare fotografie ad alta risoluzione, che possa mettere a fuoco l'area del dito e che scatti foto in modo automatico quando ritiene che la qualità dell'immagine sia sufficientemente buona. Una volta scattata una foto, l'applicazione deve inviarla automaticamente, tramite Wi-Fi, ad un software in esecuzione su un computer. Quest'ultimo si occupa della creazione del database e della visualizzazione dei dati ottenuti.

L'acquisizione del database è un operazione fondamentale per valutare le performance del sistema, è quindi necessario che sia costruito utilizzando specifici criteri in modo da realizzare una base di dati coerente. Le impronte digitali sono state dunque acquisite nelle medesime condizioni (luminosità, luogo, sfondo…) ed è stato definito un apposito protocollo di acquisizione che prevede l'acquisizione di 20 foto per ogni utente (10 per l'indice sinistro e 10 per l'indice destro), in due sessioni diverse separate da almeno una settimana di tempo. Al fine di effettuare dei test sulle performance del sistema, in particolare per valutarne l'efficienza e la correttezza, è stato anche necessario acquisire le impronte dei soggetti tramite uno scanner per impronte digitali.

Una volta acquisito il database è stato necessario sviluppare un software che implementasse un algoritmo di localizzazione dell'area del dito all'interno della foto e un algoritmo di estrazione delle minuzie nella zona precedentemente trovata. Per fare questo è stata sviluppata un'applicazione che implementa i due algoritmi e che visualizza graficamente tutti i loro passi intermedi, in modo da poter stabilire l'efficienza di ogni singolo passo e riscontrare rapidamente eventuali errori.

Infine sono stati eseguiti test sulle performance del sistema per valutare l'errore commesso da una simile metodologia di acquisizione; questo è avvenuto confrontando le minuzie estratte dalle immagini acquisite tramite scanner con quelle acquisite tramite la fotocamera dello Smartphone; per garantire la correttezza del calcolo dell'errore sono stati effettuati dei confronti "*genuine*" e dei confronti "*impostor*". In questa fase è stato necessario affidarsi ad un procedimento automatico che permetta di discernere le immagini a fuoco da quelle sfuocate e concentrare il calcolo delle performance sulle sole immagini giudicate di buona qualità.

Tutti i passi sopra enunciati sono trattati e descritti approfonditamente nei capitoli seguenti.

Capitolo 1.
Le impronte digitali

1.1 I sistemi biometrici

Un sistema di riconoscimento biometrico è un sistema informatico che riconosce una persona determinando la corrispondenza di uno specifico aspetto fisico o comportamentale posseduto da tale individuo (Maltoni, D; Maio, D; Jain, A. K. e Prabhakar, S. *Handbook of Fingerprint Recognition*).
I sistemi biometrici assumono nomi diversi a seconda del contesto applicativo in cui operano:

- *Sistema biometrico di verifica*: effettua un confronto tra la caratteristica biometrica di una persona con quella del suo modello già memorizzato nel sistema. Viene effettuato un confronto "uno a uno" per verificare che l'identità che l'individuo dichiara di possedere sia effettivamente corretta.
- *Sistema biometrico di identificazione*: effettua un riconoscimento di un soggetto cercando fra tutti i modelli dell'intero database, all'interno del quale viene condotta una ricerca "uno a molti" per stabilire, se nota, l'identità dell'individuo.

È dunque possibile dividere un sistema biometrico in due moduli logici:

- *Modulo di Enrollment*: durante questa fase la caratteristica biometrica viene acquisita da un lettore biometrico che produce una rappresentazione digitale della stessa. Viene effettuato un controllo per verificare la qualità della rappresentazione prima che venga processata ulteriormente

da un estrattore di caratteristiche (“*feature extractor”*) che genera un *template* da memorizzare nel database di sistema.

A seconda del compito che il sistema biometrico deve assolvere, la seconda unità logica può dunque essere:

- *Modulo di verifica*: durante questa fase l'utente dichiara la sua identità (attraverso username o PIN) prima che il lettore biometrico acquisisca i dati. Viene prodotta la rappresentazione digitale che viene passata in input al *feature matcher* che lo confronta con il singolo *template*, estratto dal database in base all'identità dichiarata o fornito direttamente dall’utente (ad esempio tramite una smart-card). Il sistema fornisce un riscontro positivo qualora l’identità dell'utente corrisponda effettivamente a quella dichiarata, in caso contrario un messaggio di errore.
- *Modulo di identificazione*: l'utente non deve fornire alcun PIN o username in quanto il sistema confronta i dati biometrici acquisiti con tutti i *template* presenti nel database. Il sistema può riconoscere l'identità dell'utente oppure non riconoscerla, segnalando un errore.

In base al dominio applicativo, un sistema biometrico può operare in due modalità:

- *Modalità online:* richiede un'immediata risposta al riconoscimento. Le caratteristiche biometriche sono catturante usando dei live scanner e sia la fase di *enrollment* che la fase di decisione sono del tutto automatiche.
- *Modalità offline*: non richiede un tempo di risposta istantaneo. È propria di sistemi semi-automatici che richiedono l'intervento umano nelle varie fasi (ad esempio un operatore che scansiona manualmente un'impronta precedentemente acquisita).

1.2 Le caratteristiche biometriche

Qualsiasi caratteristica fisica o comportamentale può essere utilizzata come identificatore biometrico, purché soddisfi i seguenti requisiti:

- *Universalità*: tutte le persone possiedono tale caratteristica biometrica.
- *Unicità*: la caratteristica di ogni persona è sufficientemente diversa da quelle delle altre persone.
- *Permanenza*: la misura biometrica deve rimanere sostanzialmente invariata nel tempo.
- *Misurabilità*: la caratteristica biometrica deve essere misurabile quantitativamente.

Durante la progettazione di un sistema biometrico è necessario considerare altre questioni quali velocità, robustezza e precisione di riconoscimento. È inoltre opportuno esaminare il fattore di elusione come indicatore della facilità con la quale è possibile ingannare il sistema con metodi fraudolenti.

Le caratteristiche di un essere umano che rispettano i requisiti citati precedentemente possono essere divise in due categorie a seconda della loro natura fisiologica o comportamentale. Le principali caratteristiche biometriche fisiologiche sono:

- *Impronte digitali*: con la loro provata unicità e stabilità nel tempo, sono la caratteristica biometrica più utilizzata nei sistemi automatici di riconoscimento dell'identità di un individuo.
- *DNA*: è il codice che contiene le informazioni genetiche di una persona. Molto usato in applicazioni forensi in quanto ogni individuo possiede un codice genetico diverso (ad eccezione dei gemelli monozigoti). L'utilizzo del DNA in sistemi online è ostacolato dalla sua difficoltà di trattamento. Numerosi processi chimici e personale altamente qualificato sono necessari per estrarre le informazioni desiderate.

- *Orecchio*: la forma e la struttura dell'orecchio e della cartilagine sono elementi caratterizzanti di ogni individuo anche se queste caratteristiche non sono uniche per ogni individuo.
- *Volto*: è una delle caratteristiche biometriche più accettate perché è il metodo più comune che utilizzano gli esseri umani per riconoscersi. Tuttavia è molto impegnativo sviluppare tecniche che possano tollerare l'effetto dell'invecchiamento, del cambiamento delle espressioni facciali e variazioni di posa rispetto alla telecamera.
- *Geometria della mano*: alcune caratteristiche della mano umana (come ad esempio la lunghezza delle dita) sono relativamente invarianti e specifiche (ma non uniche) di ciascun individuo.
- *Iride*: la tessitura dell'iride di un occhio umano è assunta ormai come una caratteristica distintiva di ogni individuo e di ogni occhio. I sistemi basati su questa caratteristica sono estremamente accurati e veloci.
- *Retina*: la vascolatura della retina viene considerata come essere una caratteristica unica di ogni individuo. È considerata molto sicura in quanto difficile da cambiare o replicare, tuttavia ne è molto complicata l'acquisizione.

Le principali caratteristiche biometriche comportamentali sono:

- *Firma*: il modo con il quale una persona firma è una caratteristica distintiva di un individuo. Oggi è la forma più accettata come metodo di verifica dell'identità nelle transazioni commerciali. Presenta il grosso svantaggio di poter cambiare in un periodo di tempo anche breve essendo influenzata dalle condizioni fisiche ed emotive della persona stessa.
- *Voce*: il timbro vocale di una persona è una caratteristica biometrica distintiva. Non è sufficientemente peculiare da permettere l'identificazione di un individuo in grandi database, inoltre il segnale perde qualità durante le fasi di acquisizione (tramite microfono) e digitalizzazione. Anche questa caratteristica comportamentale viene condizionata dallo stato fisico ed emotivo di una persona.

L’uso di caratteristiche biometriche ai fini del riconoscimento presenta numerosi vantaggi rispetto ai tradizionali sistemi d’autenticazione basati sull’uso di PIN e password, che possono facilmente essere dimenticati dai legittimi proprietari, ceduti ad altri, o rubati da utenti non autorizzati. Proprio per questi motivi i sistemi biometrici sono ritenuti molto più affidabili dei sistemi tradizionali, vista la difficoltà di falsificazione di molte caratteristiche biometriche.

1.3 Storia delle impronte digitali

Le rappresentazioni di impronte digitali trovate in numerosi reperti archeologici hanno evidenziato come gli antichi fossero già consapevoli dell'unicità delle impronte digitali pur senza alcuna base scientifica apparente. Sono state scoperte tavolette babilonesi risalenti al 500 a.C. riguardanti transazioni commerciali che recavano impresse sulla loro superficie un'impronta digitale come firma personale.

Lo studio delle impronte digitali che va sotto il nome di *dattiloscopia* ha però origini più recenti, solo nel XVII secolo la scienza si interessò a questo argomento con il botanico Nehemiah Grew e l'anatomista Marcello Malpighi che, quasi contemporaneamente, pubblicarono i loro studi sulla struttura delle creste e dei pori. Nel 1823 John Evangelist Purkinje, professore di anatomia all'Università di Breslavia, introdusse il primo schema di classificazione delle impronte in nove differenti classi basate sulla struttura generale delle creste.

Verso la fine del XIX secolo, Sir Francis Galton condusse numerosi studi introducendo il concetto di minuzie come base per la verifica dell'uguaglianza tra due impronte. Nello stesso periodo, Edward Henry introdusse un sistema di classificazione delle impronte in grado di velocizzare il processo di identificazione (a quel tempo svolto manualmente da esperti) che portò già all'inizio del XX secolo al loro utilizzo nei tribunali di diversi stati.

Negli anni '60 l'FBI sviluppò il primo sistema automatico di identificazione delle impronte: AFIS (*Automated Fingerprint Identification System*).

1.4 Utilizzo delle impronte digitali

I sistemi di riconoscimento biometrico basati su impronte digitali vengono utilizzati in diversi tipi di applicazioni, sia in ambito governativo (militare, sanità, giustizia, enti e istituzioni pubbliche), sia in quello commerciale (turismo, trasporti, banche, assicurazioni, hi-tech, telecomunicazioni, industria), per assicurare una maggiore sicurezza ai sistemi, alle transazioni e alla tutela dei dati (Maltoni, D; Maio, D; Jain, A. K. e Prabhakar, S. *Handbook of Fingerprint Recognition*). Le applicazioni maggiormente in uso sono:

- Autenticazione degli accessi fisici in locali protetti
- Sicurezza nelle transazioni finanziarie
- Prevenzione delle frodi
- Protezione di transazioni bancarie via internet
- Identificazione di soggetti
- Sicurezza negli aeroporti
- Attività investigative e forensi

Questa tecnologia è in continua crescita e si presume che nel futuro questi sistemi avranno una profonda influenza nella nostra vita quotidiana.

1.5 Anatomia delle impronte digitali

Le impronte digitali si formano definitivamente nel feto a circa 7 mesi di gestazione e fanno parte del fenotipo di un individuo, ossia tutte le caratteristiche fisiche di una persona. Lo sviluppo del fenotipo è ritenuto essere univocamente determinato dalla combinazione di uno specifico codice genetico con uno spe-

cifico ambiente (Maltoni, D; Maio, D; Jain, A. K. e Prabhakar, S. *Handbook of Fingerprint Recognition*). La micro-diversità delle condizioni ambientali nelle immediate vicinanze di ciascun polpastrello caratterizza la formazione dei più minuti dettagli della loro superficie. Anche due gemelli monozigoti, pur possedendo lo stesso codice genetico, hanno impronte digitali diverse. Analizzando la struttura delle impronte digitali è facile osservare che sono costituite da un flusso orientato di creste ("*ridge lines"*) e valli ("*valleys"*), che formano un disegno complesso chiamato *ridge pattern*, come esemplificato nella figura 1.1 .

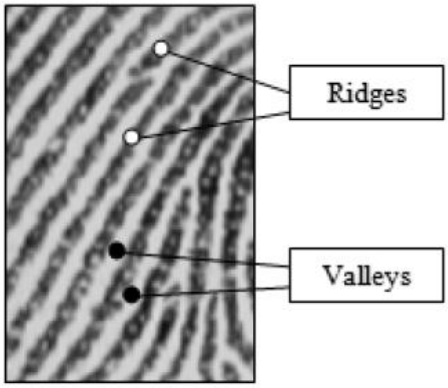

Figura 1.1 Elementi strutturali di un impronta digitale: creste e valli

L'andamento delle *ridge line* può essere efficacemente descritto dall'immagine direzionale: una matrice i cui elementi denotano l'orientazione della tangente alle *ridge line* in corrispondenza dei blocchi di una griglia sovrapposta all'immagine dell'impronta.

Studiando la struttura di un'impronta ad un livello globale si può notare come le creste assumano forme particolari caratterizzate da elevate curvature. Queste regioni, chiamate singolarità o punti singolari, possono essere classificate in tre tipi: cicli ("*loop"*), delta e spirali (*"whorl"*). Queste regioni sono caratterizzate da forme specifiche, rispettivamente simili a ∩, Δ e O. Si definisce inoltre il punto di *core* come il punto più a nord della *ridge line* più interna. La figura 1.2 ne mostra alcuni esempi.

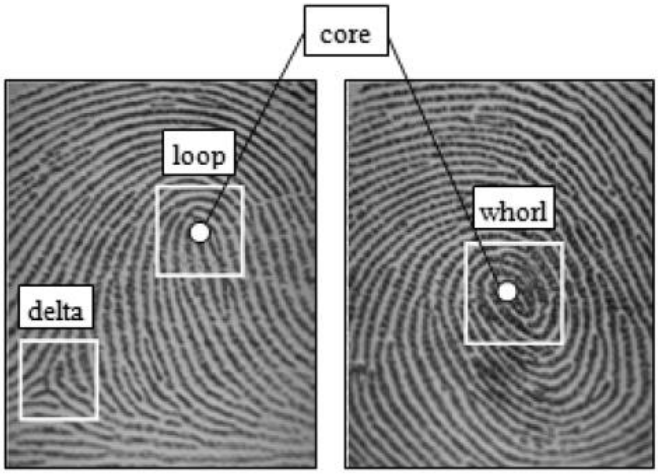

Figura 1.2 Elementi strutturali di un impronta digitale: delta, cicli, spirali

Ad un livello locale vi sono altre importanti caratteristiche chiamate minuzie ("*minutiae"*), esse rappresentano discontinuità della struttura creste/valli. Nel 1892 Sir Francis Galton classificò le minuzie ed osservò che esse rimangono identiche per tutta la vita di un individuo.

Sono state individuate ben sette principali tipologie di minuzie, ma in applicazioni pratiche è piuttosto difficile capire la differenza tra un tipo e un altro. Per questo motivo l'FBI attualmente considera solo due classi di minuzie: le terminazioni e le biforcazioni. Si ha un punto di terminazione quando una *ridge line* si interrompe bruscamente, mentre si ha un punto di biforcazione quando una *ridge line* diverge in due rami (figura 1.3).

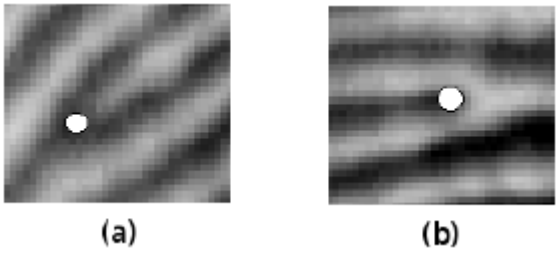

Figura 1.3 Minuzie. Biforcazione (a) e terminazione (b).

Il numero di minuzie, la loro posizione e il loro orientamento rappresentano una caratteristica altamente distintiva che consente di distinguere un'impronta da un'altra.

Se l'immagine di un impronta viene acquisita ad alta risoluzione (1000 dpi) è possibile identificare i pori della pelle che appaiono come piccoli puntini in ogni cresta (figura 1.4). Il numero, la posizione e la forma di questi pori sono caratteristiche altamente distintive e alcuni sistemi di riconoscimento sono basati proprio sul confronto di queste peculiarità.

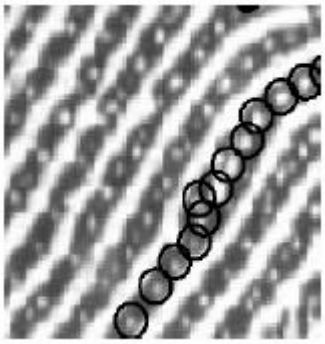

Figura 1.4 Struttura dei porti all'interno delle creste.

1.6 Classificazione delle impronte digitali

L'enorme mole di confronti che devono essere effettuati per l'identificazione di un'impronta su un ampio database, può comportare un tempo di risposta inaccettabile. Per questo motivo la comunità scientifica ha volto l'attenzione verso la ricerca di metodi di classificazione per le impronte, comparando solo impronte della stessa classe.

Nel 1900 Edward Henry raffinò un precedente studio di Galton, introducendo uno schema di classificazione che tuttora viene adottato in tutto il mondo. Sulla base del numero e della posizione dei punti singolari, lo schema di Galton-Henry divide le impronte in cinque classi, delle quali è possibile vedere un esempio nella figura 1.5 :

- *Arch*: impronte ad arco semplice, in cui le creste entrano da un lato, crescono verso il centro e scendono per poi uscire dal lato opposto.
- *Tended Arch*: impronte ad arco triangolare, ossia impronte che hanno lo stesso andamento di quelle ad arco semplice, ma in cui le creste formano un angolo o una piega al centro con la presenza di un delta.
- *Right Loop*: impronte con ansa a destra in cui una o più creste entrano dal lato destro, si ripiegano, superano la linea immaginaria determinata dal core ed escono dallo stesso lato.
- *Left Loop*: impronta con ansa a sinistra. Come le precedenti, ma piegate dal lato opposto.
- *Whorl*: impronte con almeno due delta e una figura chiusa (circolare, ellittica o a spirale) centrale.

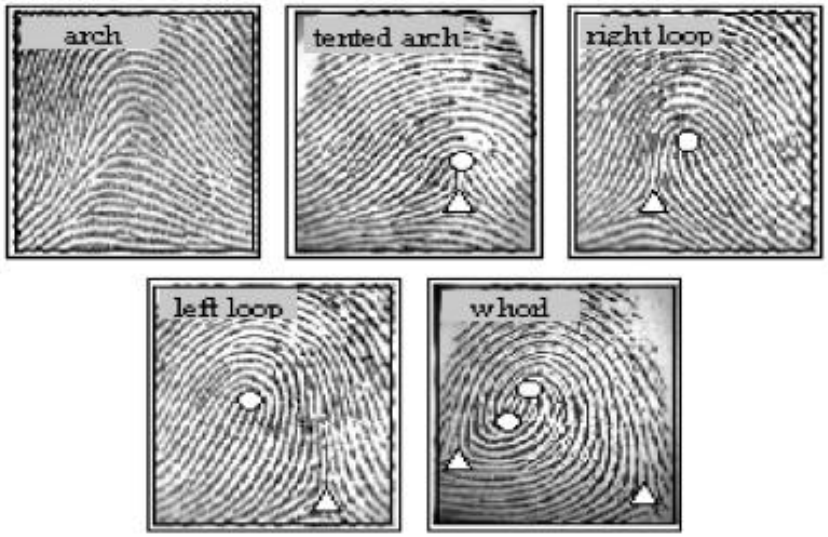

Figura 1.5 Classificazione delle impronte digitali.

Capitolo 2.
Gli Smartphone

2.1 Storia e sviluppo

Il telefono fu inventato da Alexander Graham Bell, che nel 1878 effettuò la prima telefonata della storia. Uno dei primi cellulari venduti al pubblico fu fabbricato da Motorola negli anni ’80, ma questo era ben diverso dai telefoni odierni: costava 4000$ e pesava circa 1 kg. Con il passare del tempo cresce l’esigenza di disporre di cellulari che non siano solamente in grado di telefonare, ma che possano svolgere diverse funzioni.

Uno Smartphone è un telefono cellulare che include al suo interno dei software e dispositivi in grado di fornire all’utente diverse funzionalità: navigare su internet, accedere all’e-mail, scattare foto, geo localizzazione tramite GPS etc. La diffusione di questi dispositivi ha completamente rivoluzionato lo stile di vita delle persone, connettendole perennemente tra di loro in tempo reale tramite internet, social network e App; ma ha influenzato anche lo sviluppo del software, promuovendo la mobilità e l’integrazione dei sistemi.

Il primo Smartphone fu sviluppato da IBM e BellSouth nel 1993 con l’idea di creare un dispositivo in grado di trasferire voce e dati; “Simon” era dotato di un Touchscreen e poteva inviare e-mail e fax.

Nel 1996 Nokia presenta il “Nokia 9000 Communicator”, dotato di una tastiera e di uno schermo in scala di grigi. Internamente era costituito da un processore AMD 24 MHz, da 8 MB di memoria e adottava un sistema operativo GEOS 3.0. Nel 1998 ne viene presentata una nuova versione, il “Nokia 9110”, dove

veniva leggermente incrementata la potenza computazionale del dispositivo e ne veniva diminuito il peso.

Nel 2002 viene prodotto il "BlackBerry 5810", un cellulare in grado di accedere alla mail e di navigare su internet. Nel 2003 la compagnia Palm produce il "Palm Treo 600", uno Smartphone in standard GPRS basato su Palm OS dotato di tastiera, schermo a colori Touchscreen, fotocamera da 0.3 megapixel, 32 MB di RAM e processore da 144 Mhz.

Il 2007 è un anno decisivo per lo sviluppo degli Smartphone in quanto vede la nascita dell'iPhone di casa Apple che avrà un grandissimo successo aprendo la strada alla diffusione degli Smartphone. Basato su iOS, questo dispositivo è dotato di fotocamera da 2.0 megapixel, display multi touch, memoria interna da 4/8/16 GB, connettività Quad-band GSM/GPRS, Wi-Fi, Bluetooth e USB. Nello stesso anno Google Inc. inizia a sviluppare il sistema operativo mobile Android.

Figura 2.1 Figura illustrativa di alcuni Smartphone. Da sinistra: IBM Simon, Nokia 9000 Communicator, Nokia 9110, BlackBerry 5810, Palm Treo 600, Apple iPhone

Nel luglio 2008 Apple pubblica l'iOS SDK (Software Development Kit) per consentire agli sviluppatori di terze parti di creare delle App (sviluppate in linguaggio Objective C) e viene lanciato l'App Store, un servizio che da un lato consente agli utenti di comprare applicazioni in modo da personalizzare ed estendere le funzionalità del dispositivo, dall'altro permette agli sviluppatori, previa registrazione e pagamento di una quota annua, di pubblicare le proprie

App nello store e ricavarne in modo molto rapido dei profitti. L'App Store si rivela anche molto redditizio per l'azienda di Steve Jobs che riceve il 30% di commissione su ogni applicazione acquistata.

Nell'ottobre dello stesso anno anche Google seguirà la scia lasciata da Apple introducendo l'Android Market che in seguito verrà rinominato Google Play Store, un negozio virtuale dedicato a dispositivi Android.

In modo analogo nel 2010 Microsoft lancia il proprio store, denominato Windows Phone Store, per gli omonimi dispositivi. (Reed, B. *A Brief History of Smartphones*; *History of the Smartphone*).

2.2 Le tre grandi famiglie

La storia degli Smartphone ha visto il susseguirsi di diversi sistemi operativi mobile ma solamente tre di questi sono riusciti a conquistare il mercato monopolizzandolo completamente: iOS, Android e Windows Phone (Figura 2.2). Le caratteristiche che hanno portato questi sistemi al successo sono la facilità e l'immediatezza di utilizzo, le accattivanti interfacce grafiche e la possibilità di scaricare le più svariate applicazioni da un apposito App Store (Stark, J. *Sviluppare applicazioni per iPhone;* Iacubino, A. *Creare applicazioni di successo per iPhone e iPad).*

Figura 2.2 Loghi delle tre grandi famiglie di Smartphone: Apple iPhone, Google Android e Microsoft Windows Phone

iOS è un sistema operativo mobile per iPhone, iPod touch e iPad, sviluppato da Apple nel 2007. Come Mac OS X è una derivazione di Unix e usa un micro-kernel XNU Mach basato su Darwin OS. Porta per la prima volta su dispositivi mobile il paradigma dell'Application Store, consentendo all'utente di estendere le funzionalità del dispositivo, scaricando e installando delle "App" in modo da personalizzarlo in base alle proprie necessità. Inizialmente il prodotto non forniva alcun supporto per applicazioni di terze parti poiché Steve Jobs pensava che le funzionalità fornite dalle web application fossero abbastanza per la maggior parte degli utenti, ma gli sviluppatori di software iniziarono ad effettuare il "jailbreak" sul dispositivo in modo da poter scrivere applicazioni di terze parti. Così nel 2008 viene pubblicato l'iOS SDK e viene lanciato l'App Store.

Android è un sistema operativo mobile sviluppato da Google Inc. a partire dal 2007. Le sue caratteristiche principali sono il fatto di essere completamente open source, chiunque infatti può modificarne il codice sorgente creando una propria distribuzione del sistema operativo, e di essere basato su Kernel Linux. Le App sono fruibili dal Google Play Store e sono sviluppate in linguaggio Java secondo la politica "write once run anywhere" tramite un apposito Android SDK ed eseguite sulla Dalvik Virtual Machine interna al sistema. Attualmente viene largamente utilizzato da tutti gli Smartphone di casa Samsung.

L'ultima grande famiglia di sistemi operativi per dispositivi mobili è rappresentata da Windows Phone; la sua trattazione viene affrontata in dettaglio nel paragrafo 2.3.

2.3 Windows Phone

Windows Phone è un sistema operativo mobile prodotto da Microsoft nel 2010. A differenza del suo predecessore Windows Mobile, questo sistema operativo è pensato per il mercato consumer ed appare completamente ridisegnato nella sua interfaccia grafica. Caratteristica principale che lo differenzia da Android e

iOS è proprio l'interfaccia utente "Metro", questa è caratterizzata da una schermata Start composta da "Live Tiles" (piastrelle), ognuna delle quali rappresenta un collegamento ad un'applicazione del telefono (figura 2.3); in seguito l'interfaccia Metro verrà utilizzata da tutti i sistemi Microsoft, come Windows 8 e X-Box.

Per questo sistema è disponibile uno store, il "Windows Phone Store", dove è possibile acquistare applicazioni e giochi. Nel 2011 Microsoft e Nokia stipulano un accordo di esclusiva per la piattaforma Windows Phone sui cellulari "Nokia Lumia" fino a quando nel settembre 2013 Microsoft acquista la divisione device e servizi Nokia per un totale di 7,17 miliardi di euro (L. Salvioli, Il Sole 24 Ore, Microsoft acquista i cellulari di Nokia per 7,17 miliardi di dollari)

Figura 2.3 Interfacce grafiche di iOS, Android e Windows Phone

2.3.1 Sviluppare App:

Per sviluppare App con Windows Phone è necessario innanzitutto scaricare il Windows Phone SDK dal sito della Microsoft. Dopo aver eseguito tutte le installazioni dei componenti richiesti è possibile sviluppare applicazioni direttamente dal software Microsoft Visual Studio (eventualmente è sufficiente anche Visual Studio 2010 Express for Windows Phone), creando tramite gli appositi

menù un “Applicazione Windows Phone Visual C#”. Dal punto di vista della programmazione vengono utilizzati due linguaggi: XAML per definire la GUI (“*Graphic User Interface”*) dell’applicazione e C# per stabilirne il comportamento in risposta agli eventi che vengono generati dall’utente. Microsoft Visual Studio fornisce un tool per “disegnare” in modo molto rapido l’interfaccia grafica tramite un meccanismo di “drag and drop” generando in automatico tutto il codice XAML. Una volta compilato il codice è possibile effettuare il debugging tramite un apposito simulatore, oppure si può effettuare il deploy dell’app sul telefono; per fare questo è necessario sbloccare il telefono tramite il Windows Phone Registration Tool, presente all’interno del SDK, e installare il software Microsoft Zune, quest’ultimo permette di sincronizzare telefono e computer abilitando gli scambi di dati. Una volta installati tutti i componenti necessari, basterà cliccare sul menù “deploy” per trasferire l’applicazione sul telefono. Tutti i software sopracitati sono fruibili gratuitamente dal sito Microsoft (Brunetti, R. *Introduzione a Windows Phone 7.5*).

Capitolo 3.
Acquisizione del database

3.1 Protocollo di acquisizione

Per affrontare in modo efficiente uno studio come quello trattato è necessario disporre di una base di dati sufficientemente ampia, in modo da garantire la generalità dei risultati ottenuti in fase di test del sistema. Per acquisire un database in modo coerente è stato stabilito il seguente protocollo di acquisizione:

- Per ogni soggetto di cui vengono raccolte le impronte digitali sono necessarie due sessioni di acquisizione.
- Fra le due sessioni deve intercorrere almeno una settimana di tempo.
- In ogni sessione vengono scattate 10 foto dell'indice sinistro e 10 foto dell'indice destro.
- Durante la prima sessione è necessario acquisire le impronte del soggetto anche tramite uno scanner, in modo da poter verificare (in fase di test) il corretto funzionamento del sistema, confrontando le minuzie estratte dalle immagini acquisite tramite scanner con quelle delle immagini acquisite dallo Smartphone.
- Tutte le acquisizioni devono essere effettuate nello stesso luogo e nelle medesime condizioni (sfondo, luminosità, etc.). In particolare è stato scelto come luogo il *Biometric System Laboratory*, e tutte le foto sono scattate dall'alto verso il basso con il dito da fotografare appoggiato su di un tavolo di colore grigio.

3.2 Struttura del database

Prima di acquisire il database, è necessario formalizzarne la struttura; è stato quindi assegnato ad ogni soggetto un numero identificativo di tre cifre in modo da creare una base di dati che sia strutturata come esemplificato dalla figura 3.1.

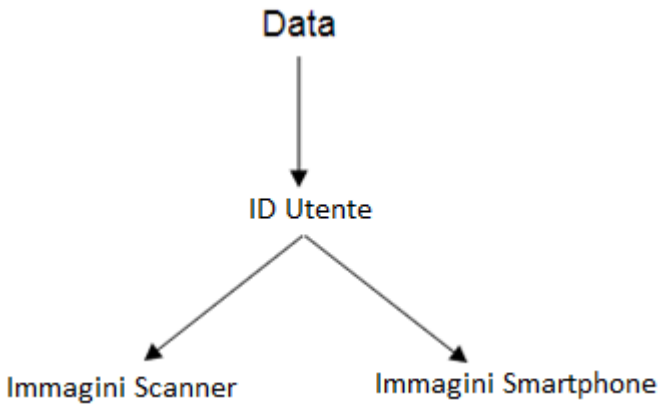

Figura 3.1 Struttura del database

Al fine di facilitare il conseguente utilizzo del database, il nome di ogni immagine memorizzata segue una codifica ben definita:

- Scanner Images: *{XXX}_{YY}_{ZZ}P.png*
  XXX = User ID (numero di 3 cifre 1-based)
  YY = Finger ID (07 indice sinistro, 02 indice destro)
  ZZ = Session ID (01 per le immagini acquisite da scanner)

- Smartphone Images: *{XXX}_{YY}_{ZZ}_{WW}S.png*
  XXX = User ID (numero di 3 cifre 1-based)
  YY = Finger ID (07 indice sinistro, 02 indice destro)
  ZZ = Session ID (01 oppure 02)

WW = Frame ID (da 00 a 09)

Nella figura 3.2 è possibile osservare nello specifico la struttura del database acquisito: è presente una cartella per ogni utente, nominata con il suo ID, all'interno di questa sono presenti due sotto-cartella, una denominata "Plain" contenente le impronte acquisite attraverso scanner, ed un'altra denominata "Smart" che contiene tutte le immagini acquisite tramite Smartphone.

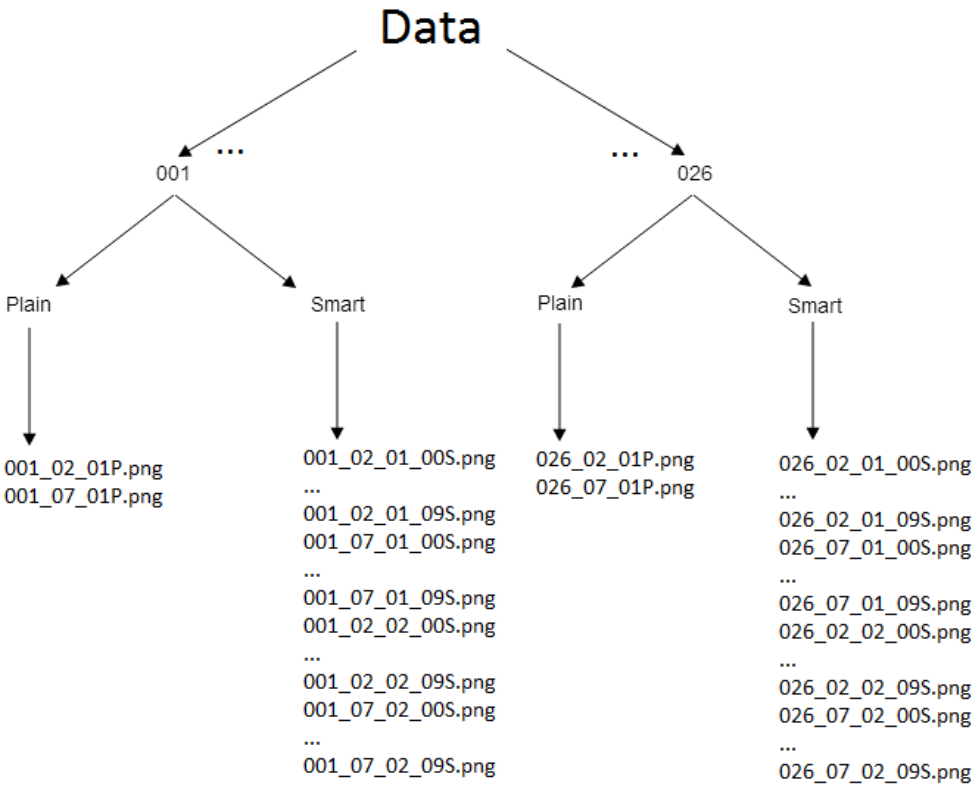

Figura 3.2 Struttura ad albero del database

A titolo esemplificativo, l'immagine "001_02_01_00S.png" rappresenta il frame numero "0" dell'indice destro dell'utente "001" acquisito nella prima sessione.

3.3 Problematiche riguardanti l'acquisizione

Aspetto fondamentale dell'attività di acquisizione, è quello di ottenere fotografie di buona qualità ad alta risoluzione. Acquisire immagini ad alta risoluzione risulta essere piuttosto semplice: è sufficiente essere in possesso di uno Smartphone dotato di una fotocamera con un numero di pixel adeguato (la maggior parte degli odierni dispositivi soddisfano questo requisito, una fotocamera da 5 Mpx è più che sufficiente). Acquisire immagini di buona qualità invece, ovvero dotate di un ragionevole grado di messa a fuoco, può essere piuttosto problematico. Sebbene sia stato sviluppato un software apposito che permette alla fotocamera di mettere a fuoco esattamente sull'area del dito, è molto difficile acquisire solamente immagini non sfuocate; questo è un problema insito nella particolare modalità di acquisizione: il movimento inevitabile della mano dell'utilizzatore dello Smartphone potrebbe comportare un degrado della qualità dell'immagine acquisita.

Per far fronte a questa problematica è stata dunque fissata una soglia: per ogni utente almeno il 50% delle immagini devono essere di buona qualità, in caso contrario è necessario ripetere l'intera sessione di acquisizione per quel determinato dito. Un'immagine viene giudicata di buona qualità se, effettuando uno zoom, è possibile distinguere in modo abbastanza chiaro le *ridge line*; è necessario disporre dunque di un software che faciliti questo procedimento mostrando rapidamente tutte le immagini acquisite (vedere paragrafo 3.4).

Altra problematica riguarda la comunicazione tra Smartphone e PC: il sistema per funzionare necessita di una connessione Wi-Fi, qualora questa venga a mancare, non è possibile effettuare l'acquisizione.

3.4 Software di acquisizione

Per acquisire il database è stato realizzato un sistema software formato da un'applicazione PC che funge da server (da questo momento in poi ci si riferirà a tale applicazione con l'appellativo di "server"), che memorizza i dati ricevuti dallo Smartphone e li mostra sullo schermo, e da un'applicazione mobile che si occupa dell'acquisizione delle impronte e le invia, tramite Wi-Fi, al server. All'interno del sistema è inoltre presente uno scanner di impronte digitali collegato tramite porta USB al PC Server. L'interazione tra i vari componenti del sistema è esemplificata dalla figura 3.3.

Per quanto riguarda lo Smartphone utilizzato, si tratta di un Nokia Lumia 520 che dispone di una fotocamera da 5Mpx.

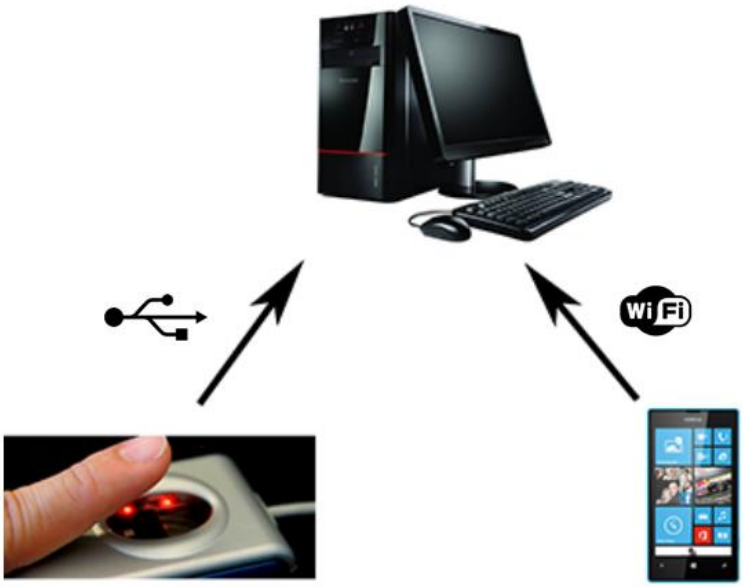

Figura 3.3 Interazione tra i componenti del sistema

Dato che il *Byometric System Laboratory* ha sviluppato numerose librerie per l'analisi di impronte digitali in linguaggio C#, si è scelto di utilizzare il framework *.NET*, e quindi di realizzare un applicazione Windows Form per il server e un'applicazione Windows Phone per quanto riguarda lo Smartphone.

3.4.1 Applicazione Windows Form:

Il server è un applicazione Windows Form che essenzialmente svolge i seguenti compiti (chiariti dal behavioral state diagram di figura 3.4):

- Iniziare e terminare le sessioni di acquisizione.
- Ricevere le foto dallo Smartphone e salvarle nella corretta posizione del database e con il corretto formato.
- Acquisire le impronte digitali tramite scanner di impronte e calcolarne l'indice di qualità *NFIQ*.
- Visualizzare in modo opportuno, per ogni utente, le impronte acquisite tramite scanner e le impronte acquisite tramite Smartphone.
- Aprire velocemente un'immagine acquisita per valutarne la qualità e mostrare la *bounding box* (vedi sezione 3.4.2) ricevuta dallo Smartphone.
- Eliminare in modo rapido tutte le foto ricevute relative ad un determinato dito e ad una determinata sessione.

A tal proposito sono stati inseriti i controlli grafici adatti a queste funzionalità e ne sono stati creati alcuni nuovi in modo da permettere un miglior funzionamento del programma. La schermata dell'applicazione (figura 3.5) è suddivisa in due parti: la parte sinistra è relativa alle impronte acquisite tramite scanner, mentre la parte destra è relativa alle impronte acquisite tramite Smartphone; è presente inoltre un numeric-up-down per selezionare l'utente corrente in modo da iniziare l'acquisizione o visualizzare le relative immagini acquisite.

Per quanto riguarda la parte dedicata allo scanner, sono presenti due anteprime delle immagini acquisite: una per l'indice destro e una per l'indice sinistro. In seguito al click su una delle immagini, questa viene ingrandita. Sono presenti due pulsanti, uno per mostrare l'immagine catturata dallo scanner e l'altro per terminare l'acquisizione.

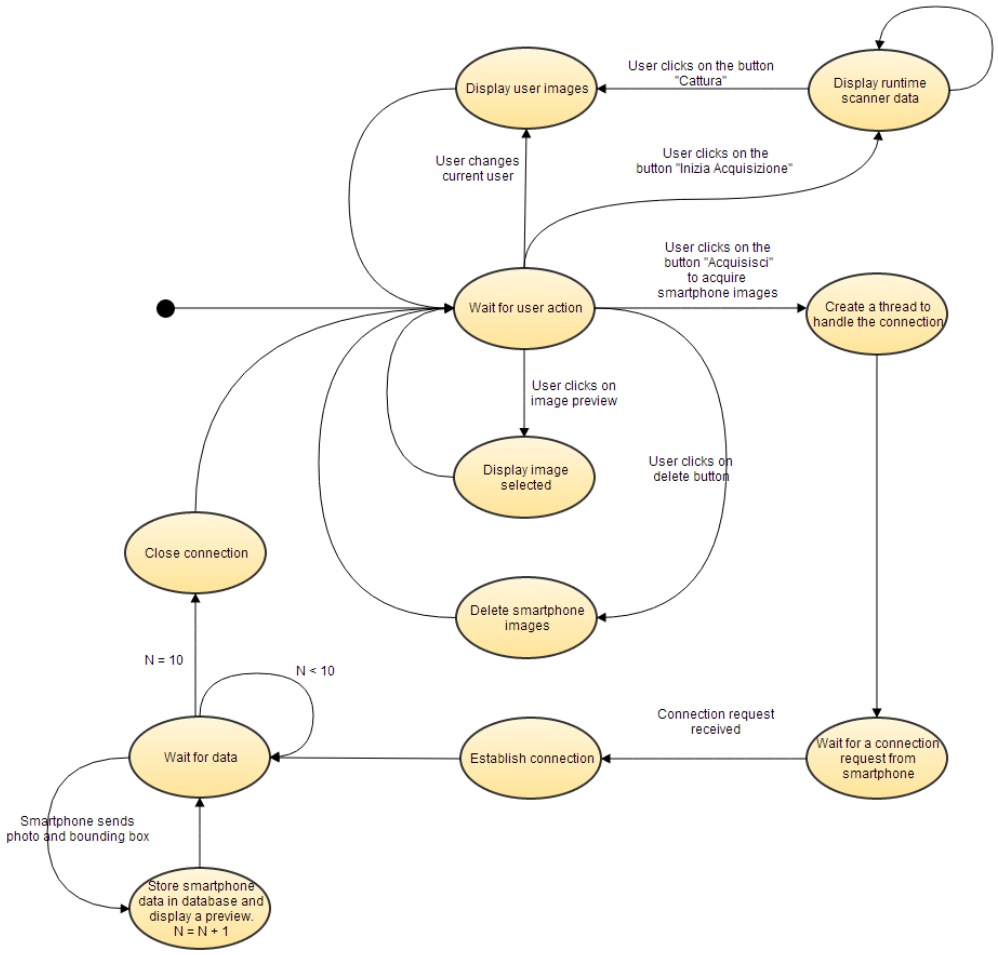

Figura 3.4 Behavioral state diagram dell'applicazione di acquisizione lato "server"

Nella parte dedicata allo Smartphone sono presenti tutte le anteprime delle immagini ricevute, organizzate per sessione e dito. Per ogni sessione e dito sono presenti due pulsanti: uno per iniziare l'acquisizione (quindi per portare il server in stato di attesa dei dati) ed un altro per eliminare tutte le foto relative a quella determinata sessione e a quel determinato dito. In seguito al click su un immagine di anteprima si apre un'altra schermata, che mostra l'immagine nella sua risoluzione originale e la sua corrispondente *bounding box* (vedi sezione seguente) ricevuta dallo Smartphone.

Una problematica rilevante per quanto riguarda l'applicazione Windows Form è quella delle performance. Quando si vogliono visionare i dati relativi ad un

determinato utente, l'applicazione deve caricare sul Form tutti i suoi dati, ovvero 42 immagini (1 scanner indice sinistro, 1 scanner indice destro, 10+10 foto dell'indice sinistro, 10+10 foto dell'indice destro). In tale contesto, in software sviluppato proprio per velocizzare l'acquisizione, una esagerata latenza è totalmente inaccettabile. Per questo motivo si è fatto largo uso della programmazione multi-threading, in particolare le immagini vengono lette dalla memoria in modo asincrono, utilizzando la classe *Parallel* del framework *.NET*, al fine di non bloccare l'interfaccia utente durante il caricamento.

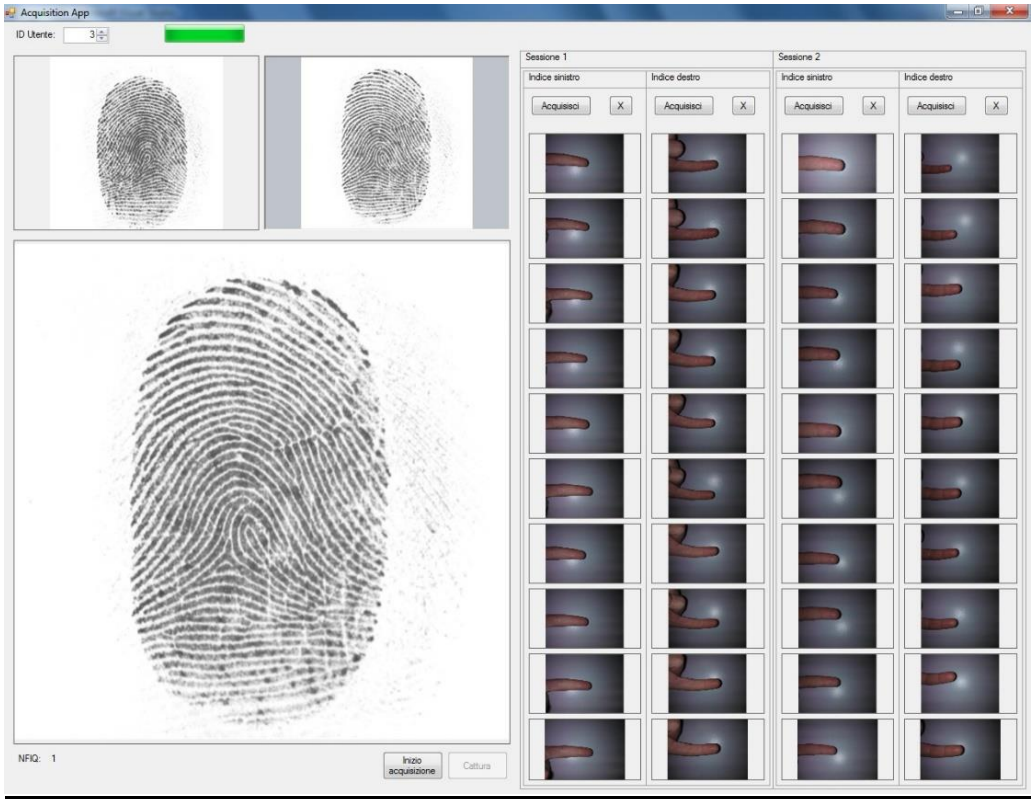

Figura 3.5 Schermata dell'applicazione di acquisizione lato "server"

3.4.2 Applicazione Windows Phone:

L'applicazione mobile è una Windows Phone Application che svolge i seguenti compiti:

- Effettuare una richiesta di connessione all'applicazione "server".

- Valutare in modo continuativo la messa a fuoco su immagini acquisite a bassa risoluzione.
- Scattare foto in automatico (ad alta risoluzione) quando l'immagine in corrispondenza del dito è a fuoco.
- Calcolare la *bounding box*, ovvero il rettangolo la cui area interna contiene esclusivamente la falange distale.
- Inviare automaticamente la foto appena scattata e la sua corrispondente *bounding box* al "server".
- Terminare l'acquisizione quando il server chiude la connessione.

In particolare l'app si comporta nel seguente modo: alla pressione del pulsante "start", il software inizia a valutare la messa a fuoco in modo continuativo. Mentre il software cerca di mettere a fuoco l'immagine in corrispondenza della zona del dito viene mostrato all'utente un messaggio indicante la qualità del frame corrente. Comparirà la scritta "*IF"* ovvero "Invalid Frame" qualora il software non riesca a mettere a fuoco il dito, suggerendo così all'utilizzatore di provare a variare la distanza tra quest'ultimo e il dispositivo. Quando sono presenti 3 "*AF"*, ovvero "Available Frame", la foto viene scattata ed inviata al server insieme alla sua *bounding box*.

Per motivi di performance, la *bounding box* è calcolata sulle immagini acquisite a bassa risoluzione. Per calcolarla è innanzi tutto necessario ottenere una *skin map* della foto acquisita, ovvero un immagine a sfondo nero con il solo dito di colore bianco, tramite un analisi del colore. Una volta calcolata la *skin map* si calcola il baricentro del dito e quindi, scorrendo l'asse y alla coordinata x del baricentro si calcola un approssimazione della sua larghezza. La lunghezza della falange distale (o falangetta) viene approssimata come due volte la larghezza del dito. Infine, siccome la *bouding box* verrà visualizzata dall'applicazione "server" su immagini ad alta risoluzione, è necessario effettuare un'apposita operazione di scalatura.

Un procedimento del tutto analogo verrà in seguito eseguito dall'applicazione "server" che andrà a ricalcolare la *bounding box*, il tutto sarà discusso nel capitolo 4.

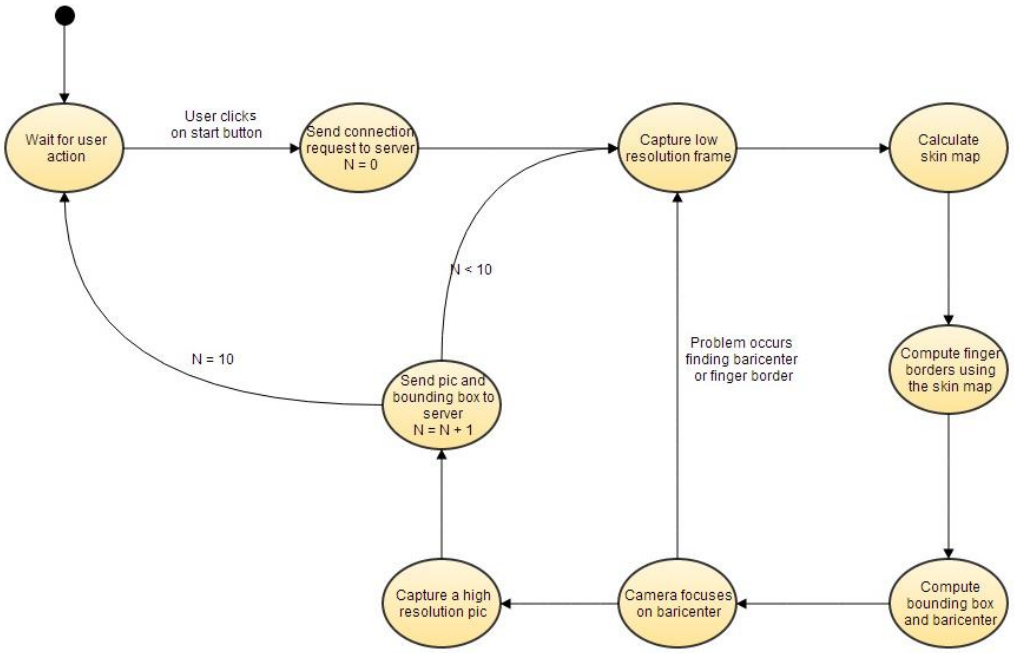

Figura 3.6 Behavioral state diagram dell'applicazione di acquisizione mobile

3.5 Informazioni statistiche sui dati acquisiti

Nella tabella sottostante vengono riportate alcune informazioni statistiche sui dati acquisiti:

Numero di soggetti:	26
Uomini:	21
Donne:	5
Età:	21-40

Tabella 3.1 Informazioni statistiche sui dati acquisiti

I soggetti sono costituiti da studenti, professori, ricercatori e personale tecnico dell'*Alma Mater Studiorum* Università di Bologna – Sede di Cesena. In parti-

colare sono presenti 11 studenti, di età compresa tra i 21 e i 23 anni, e altri 15 soggetti di età compresa tra i 25 e i 40 anni.

Per alcuni soggetti, causa qualità troppo bassa delle immagini ottenute, è stato necessario ripetere l'acquisizione diverse volte, questo si è verificato circa nel 30% dei casi.

La prima sessione di acquisizione è stata svolta tra il 21 e il 23 ottobre 2013, mentre la seconda sessione tra il 28 ottobre e il 18 novembre 2013.

Capitolo 4.
Individuazione del dito all'interno della foto

4.1 L'importanza della *bounding box*

È evidente che un'immagine acquisita tramite una fotocamera, per quanto possa essere di buona qualità, ha ben poco a che vedere con un acquisizione effettuata tramite scanner di impronte, come mostrato dalla figura 4.1 .

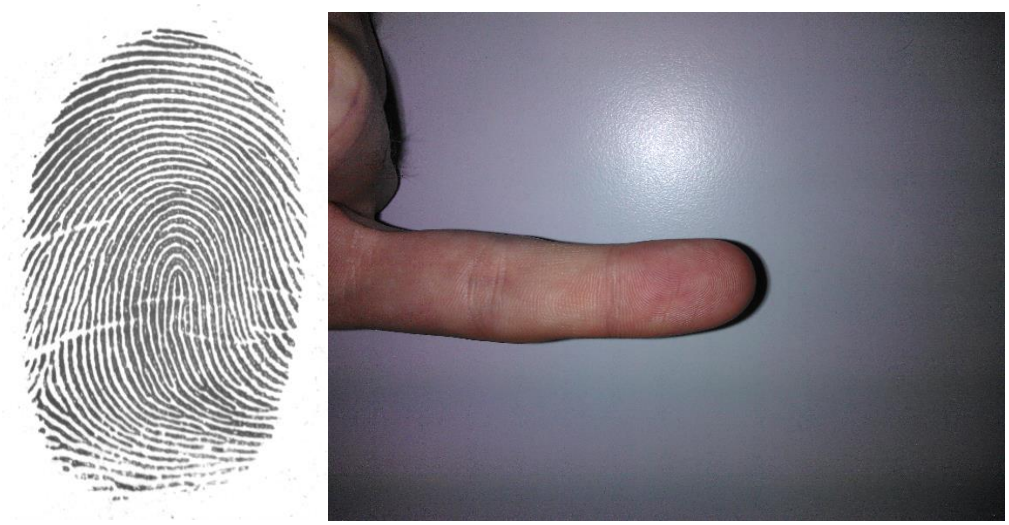

Figura 4.1 A sinistra, immagine acquisita tramite scanner per impronte digitali, a destra immagine acquisita tramite la fotocamera di uno Smartphone

Risulta dunque necessario effettuare opportune elaborazioni intermedie che permettano di ricondurre, per quanto possibile, la fotografia di un dito ad un'immagine acquisita tramite scanner.

A questo scopo è stato sviluppato un software ad hoc che utilizza un algoritmo di individuazione del dito e algoritmi di estrazione delle minuzie; tale software permette di visualizzare tutte le elaborazioni intermedie che è necessario operare sulle immagini.

In particolare, implementare un efficiente algoritmo di individuazione del dito risulta essere fondamentale per garantire la correttezza delle elaborazioni eseguite dall'intero sistema.

Viene definita bounding box il rettangolo all'interno del quale si trova la porzione di immagine di interesse; in questo lavoro essa corrisponde al rettangolo all'interno del quale si trova la falange distale del dito.

Una *bounding box* non esattamente posizionata e dimensionata può peggiorare notevolmente le performance del sistema compromettendone i risultati. Ad esempio, se essa comprende solamente parte del dito interessato, si andranno a perdere delle minuzie; viceversa se la *bounding box* oltre a comprendere l'area della falange distale comprende anche parte della falange media, si andranno ad estrarre anche minuzie che non sono presenti nell'impronta acquisita tramite scanner, causando così potenziali problemi in fase di *matching* (ovvero di confronto delle minuzie estratte).

Emerge quindi l'esigenza di realizzare un algoritmo che sia il più accurato possibile per non commettere, durante l'estrazione delle minuzie, errori che andrebbero a compromettere la correttezza dei risultati.

4.2 Primi tentativi e relative problematiche

Prima di riuscire ad implementare l'algoritmo definitivo in grado di calcolare la *bounding box*, sono stati effettuati diversi tentativi utilizzando varie strategie. Di seguito vengono elencate e spiegate in tutte le loro parti, in modo da

fornire al lettore una miglior comprensione del problema e del lavoro effettuato.

In una prima analisi, si è pensato fosse conveniente far eseguire il calcolo della *bounding box* direttamente dallo Smartphone. Questa veniva calcolata sull'immagine catturata a bassa risoluzione per problemi di performance, e poi ingrandita di un fattore di scala per applicarla all'immagine ad alta risoluzione. Tale metodologia, tuttavia, è risultata essere un approccio semplicistico al problema, è stato riscontrato operativamente la sua mancanza di precisione e di correttezza. In moltissimi casi l'algoritmo non riusciva ad identificare il dito, andando a creare *bounding box* sovradimensionate (figura 4.2) e mal posizionate (figura 4.3); questo fatto era causato proprio dal fatto che l'algoritmo veniva applicato sulle immagini a bassa risoluzione.

Per ovviare a questo si è deciso di effettuare il calcolo della *bounding box* sulle immagini ad alta risoluzione; questo implica il fatto che l'algoritmo di individuazione del dito non possa essere eseguito sullo Smartphone; se così fosse, infatti, l'algoritmo verrebbe eseguito ogni qual volta si scatta una foto e, a causa dei tempi di computazione troppo elevati per elaborare un'immagine a 5Mpx su una CPU Snapdragon S4 Dual Core 1 Ghz, si avrebbe un'eccessiva latenza tra un acquisizione e l'altra rendendo il sistema inutilizzabile. L'unica possibilità, dunque, è quella di eseguire l'individuazione della *bounding box* sul "server".

Questo procedimento consente di ottenere *bounding box* molto più corrette e precise; nelle figure 4.2 e 4.3 è possibile visualizzare un esempio di *bounding box* calcolata dallo Smartphone (rosso) e di *bounding box* calcolata lato server (giallo).

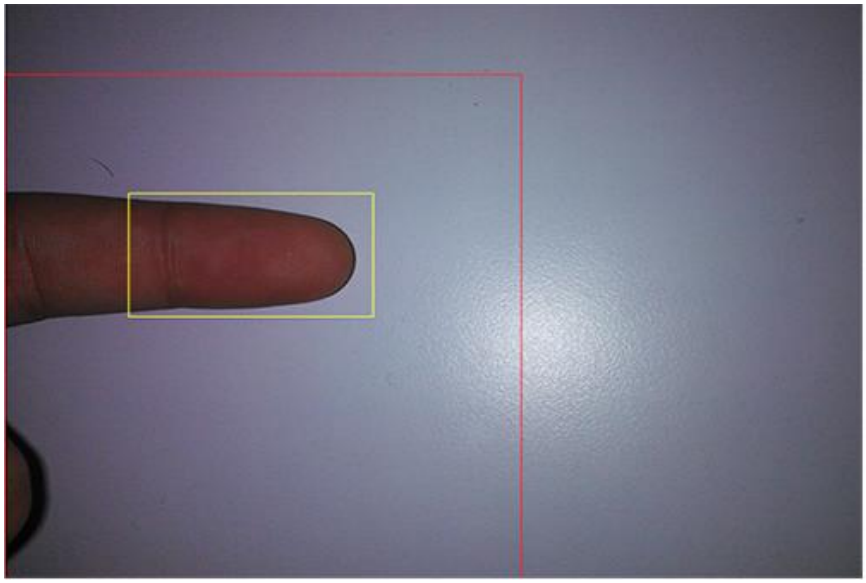

Figura 4.2 Caso estremo: *bounding box* calcolata dallo Smartphone (in rosso) e calcolata lato server (in giallo)

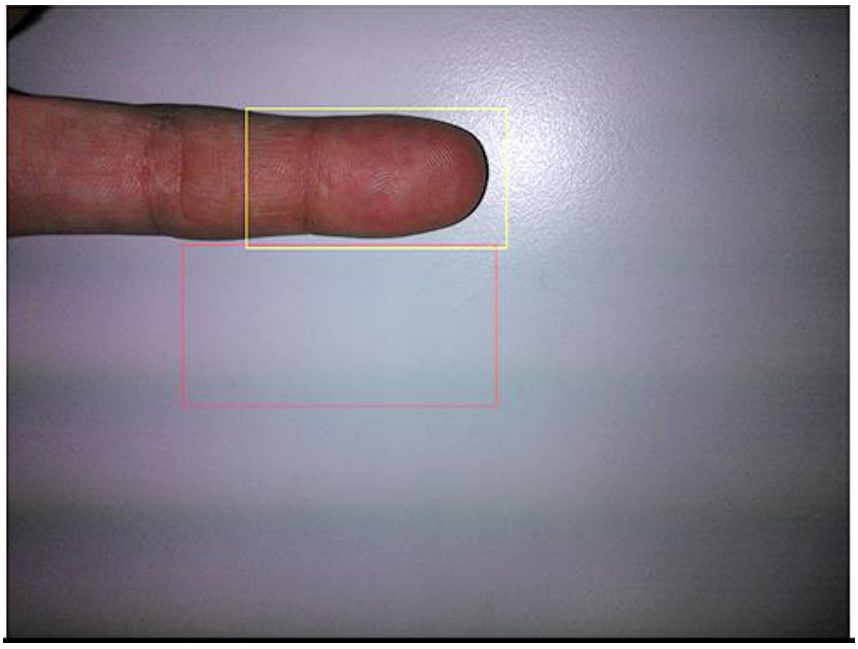

Figura 4.3 Caso estremo: *bounding box* calcolata dallo Smartphone (in rosso) e calcolata lato server (in giallo)

Al fine di migliorare ulteriormente i risultati, oltre ad eseguire sul server l'algoritmo su immagini ad alta risoluzione, lo stesso è stato modificato per aumentarne la robustezza, come descritto nella sezione seguente. Questo ha

consentito di ridurre ulteriormente il numero di errori, evitando anche errori quando all'interno delle immagini non è presente solamente il dito, ma anche parte della mano (per un esempio di tale situazione si veda la figura 4.4).

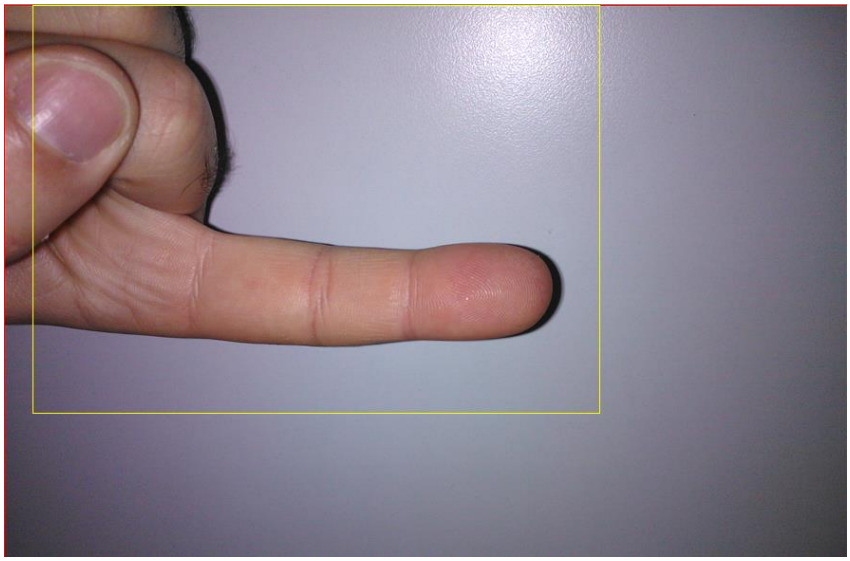

Figura 4.4 Caso in cui l'immagine presenti parte della mano del soggetto

4.3 Algoritmo di individuazione del dito

Il primo passo consiste nel calcolare la *skin map*, quest'ultima è una rappresentazione dell'immagine originale in due soli colori: nero e bianco. La *skin map* è di colore nero nelle zone esterne al dito e di colore bianco nelle zone interne (figura 4.5); il tutto viene effettuato tramite un'analisi del colore utilizzando i tre canali YCbCr. La *skin map* risulta essere fondamentale per il calcolo della *bounding box*, questo perché essa consente di individuare la posizione del dito all'interno dell'immagine in modo relativamente semplice, sulla base del colore atteso della pelle.

Un simile approccio, tuttavia, non è esente da errori e numerosi altri pixel possono essere selezionati per via del rumore nell'immagine. Un esempio è mostrato nella figura 4.5; l'immagine a sinistra mostra un caso in cui la *bounding box* risulta completamente errata (cerchiata in giallo per maggior chiarezza).

L’immagine a destra, mostra come la causa di questo comportamento sia dovuta alla presenza di vari pixel erroneamente bianchi nella *skin map* (cerchiati in rosso) all’esterno dell’area del dito che vanno ad alterare, come sarà chiarito in seguito, il calcolo della *bounding box*.

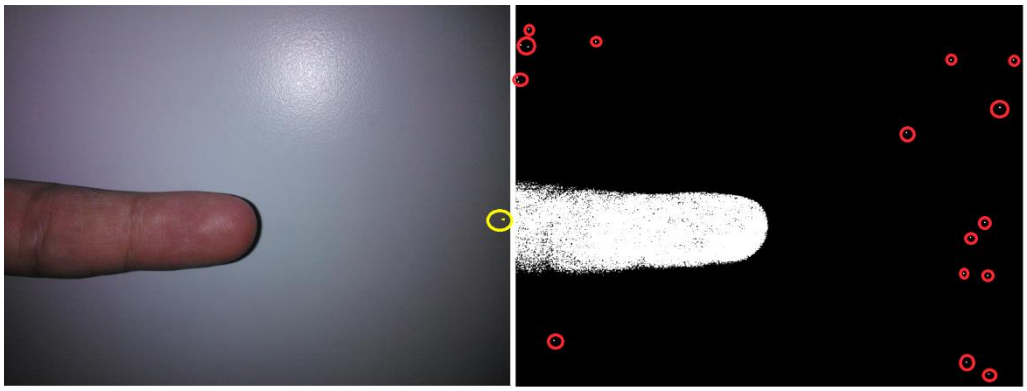

Figura 4.5 A sinistra l’immagine originale, a destra la sua relativa *skin map*

In questa fase tali errori possono essere evitati eseguendo un’operazione di morfologia matematica di apertura avente come elemento strutturante un cerchio di dimensione 9 pixel: questa consente di eliminare le zone debolmente connesse andando così a cancellare tutti i pixel bianchi esterni all’area del dito. Nella figura 4.6 è illustrata la *skin map* della stessa immagine dopo un operazione di questo tipo: si noti che in questo caso la *bounding box* risulta essere corretta.

Figura 4.6 A sinistra l’immagine originale, a destra la sua relativa *skin map* in seguito ad un operazione di apertura

Successivamente nell'algoritmo viene determinata la lunghezza del dito, questo viene effettuato scorrendo tutte le colonne della *skin map* partendo da destra; la componente x del primo pixel bianco trovato rappresenta la lunghezza totale del dito, nonché la coordinata x dove terminerà la *bounding box*. La sua lunghezza è stimata come due volte la larghezza.

Sebbene sia relativamente semplice individuare la lunghezza della *bounding box*, non è altrettanto semplice calcolarne la larghezza; infatti non è possibile sapere a priori dove si trovi il dito all'interno dell'immagine e se sia presente o meno parte della mano all'interno della foto, non è possibile utilizzare un algoritmo similare a quello utilizzato per calcolare la lunghezza del dito.

Una prima strategia che è stata sperimentata in questo lavoro, è quella di calcolare il baricentro del dito e, partendo da questo punto, valutarne la larghezza. Questa viene calcolata scorrendo la colonna della *skin map* corrispondente al baricentro partendo da quest'ultimo. L'ultimo pixel bianco trovato scorrendo verso l'alto rappresenta l'estremo superiore della *bounding box*, mentre l'ultimo pixel bianco trovato scorrendo verso il basso l'estremo inferiore. I risultati così ottenuti sono mostrati nella figura 4.7 .

Questo metodo è abbastanza efficacie in immagini contenenti il solo dito, dove il baricentro si trova nella zona interna ad esso, ma risulta essere totalmente errato in immagini che contengono parte della mano del soggetto. Questo perché la presenza della mano (e quindi di molti pixel bianchi aggiuntivi nella *skin map*) vanno ad alterare il calcolo del baricentro che risulta quindi essere spostato, andando a ricadere in zone periferiche o addirittura esterne al dito, compromettendo quindi la determinazione della sua larghezza.

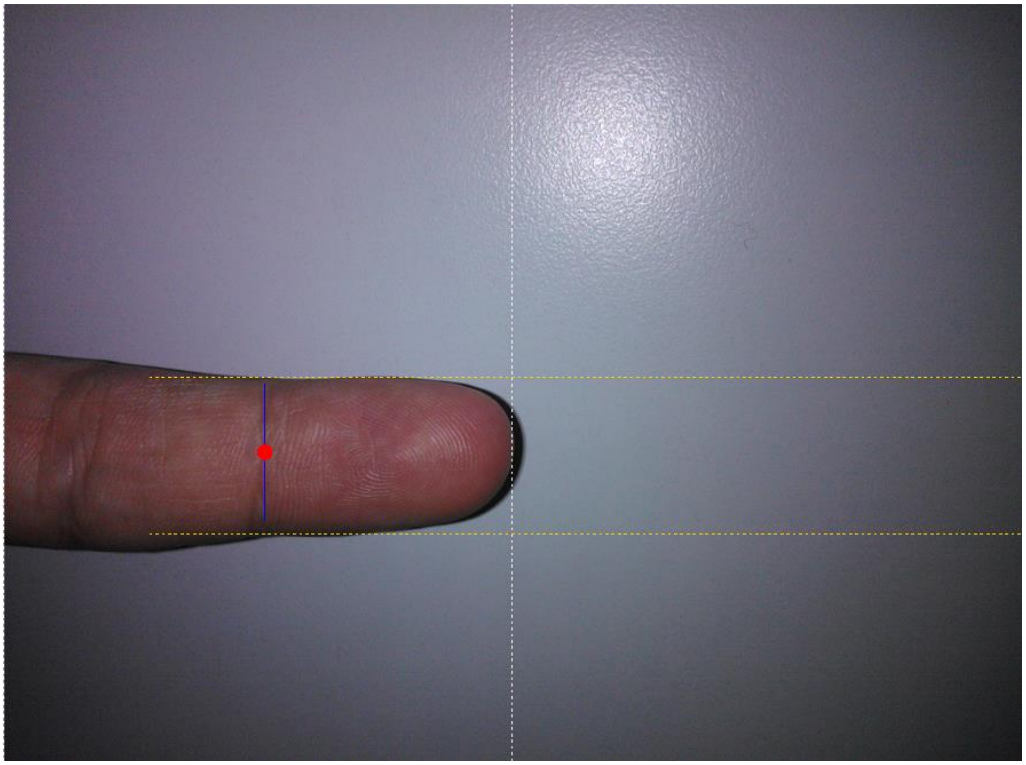

Figura 4.7 Baricentro (in rosso), coordinata x dove terminerà la *bounding box* e rispettiva larghezza

Per facilitare la comprensione di questa problematica, nel seguito vengono riportati alcuni esempi; la figura 4.8 rappresenta un caso di immagine contenente parte della mano del soggetto: si può notare come questo fattore vada a inficiare sul calcolo del baricentro che risulta essere spostato verso sinistra e verso l'alto. In questo caso, fortuito, il baricentro si trova però ancora in una porzione dell'immagine dove è presente il solo dito, quindi in generale la *bounding box* risulterà essere abbastanza corretta ma imprecisa. Si noti infatti che essendo presente una leggera curvatura nel dito, determinarne la larghezza in questo modo comporta una perdita di alcuni pixel nella porzione di immagine di interesse.

Nella figura 4.9 è rappresentato un caso estremo (ma non il peggiore rilevato) di immagine contenente parte della mano del soggetto. La presenza della mano altera notevolmente il calcolo del baricentro, che ricade in una zona dove è presente la mano stessa. È evidente, quindi, che quando si andrà a calcolare la larghezza del dito, questa sarà totalmente errata, come si può notare nella figura.

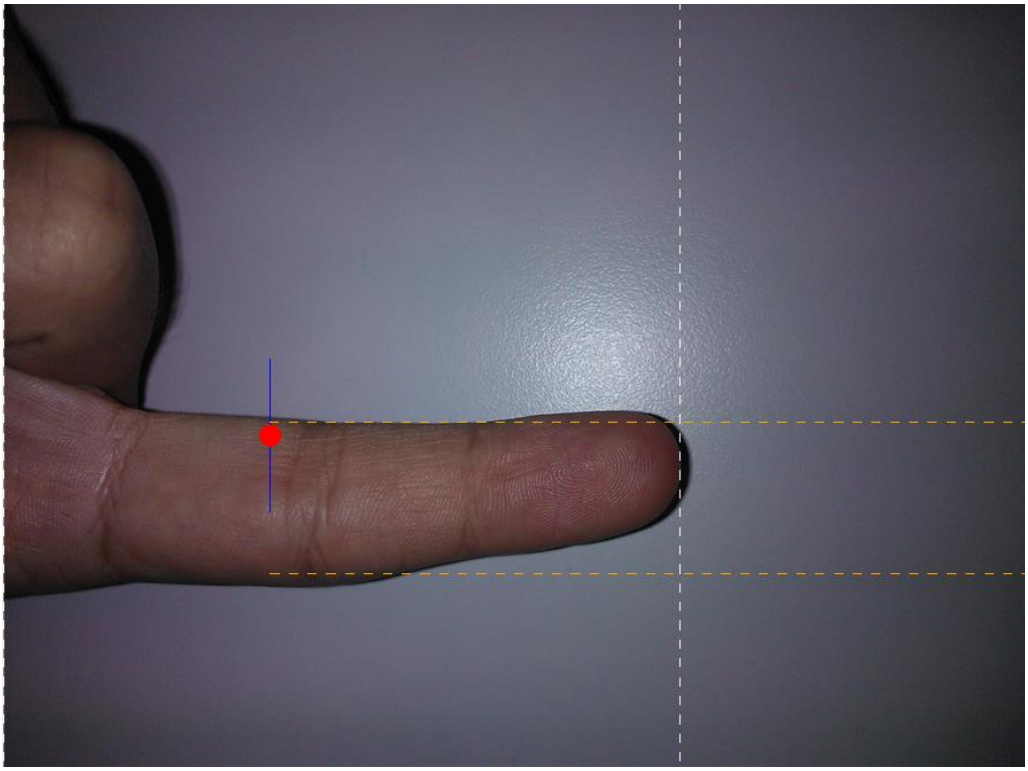

Figura 4.8 Immagine con baricentro spostato, il calcolo della larghezza del dito è impreciso. Vengono persi dei pixel in corrispondenza della falange distale

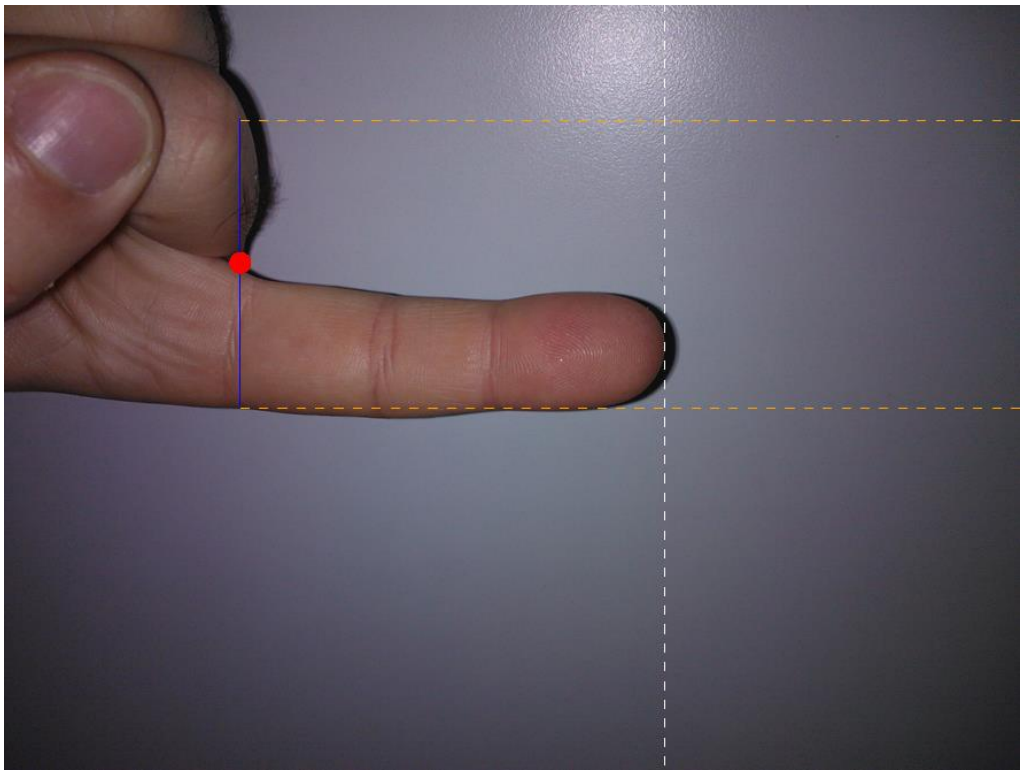

Figura 4.9 Immagine con baricentro spostato, il calcolo della larghezza del dito è totalmente errato

Una possibile soluzione potrebbe essere quella di scartare una certo numero dei primi pixel dell'immagine a partire da sinistra, ciò permetterebbe di non considerare la zona relativa alla mano del soggetto, se presente, e di calcolare in modo corretto il baricentro.

Ma anche in questo caso, non sapendo nulla a priori sulla posizione del dito, non è possibile stabilire staticamente il numero di pixel da scartare; è stato implementato quindi un algoritmo ad hoc che svolge questa funzione in modo dinamico. L'idea è quella di partire dalla punta del dito e, scorrendo verso sinistra, di calcolare ad intervalli regolari (ogni *dx* pixel) le variazioni di larghezza: quando la variazione di larghezza è superiore ad una determinata soglia significa che si è incorsi in una porzione di immagine contenente parte della mano del soggetto. La componente *x* di questo punto, con l'aggiunta di una costante, rappresenterà il numero di pixel da scartare a sinistra.

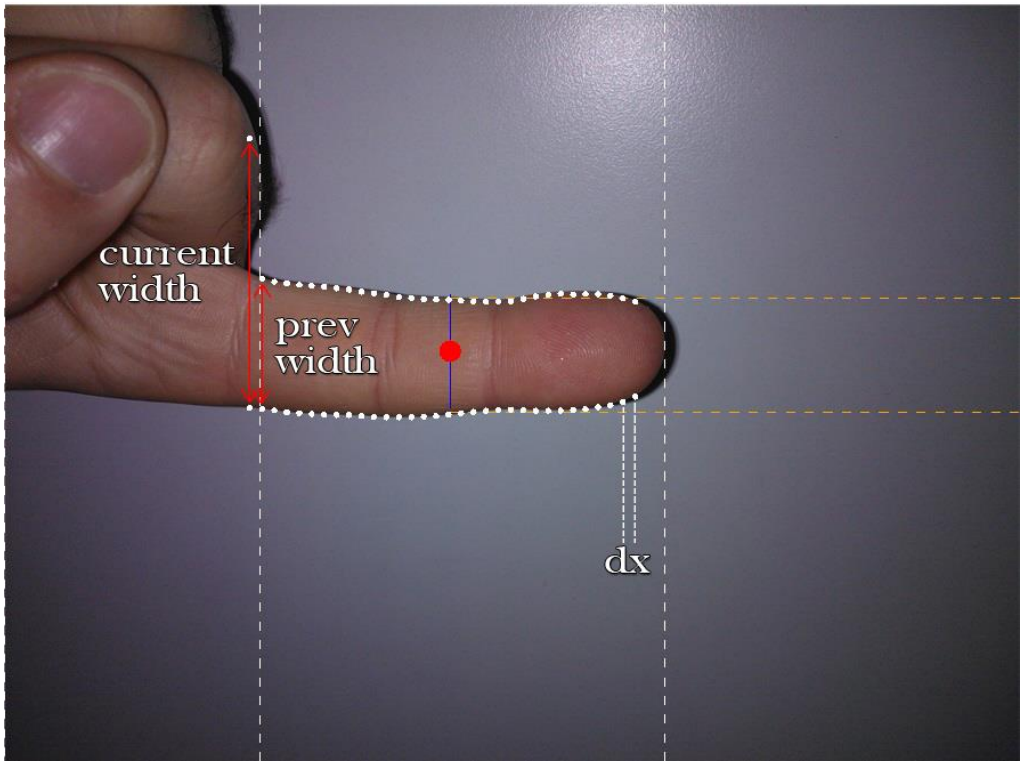

Figura 4.10 Implementazione del nuovo algoritmo. La larghezza del dito viene determinata correttamente

Un'esemplificazione grafica è mostrata dalla figura 4.10; si noti che quando si raggiunge la mano, la larghezza aumenta considerevolmente causando il termine dell'algoritmo e quindi un corretto calcolo della larghezza del dito. Il baricentro, ovviamente, è in una posizione differente da quanto visto nella figura 4.9 .

La punta del dito, a causa della sua curvatura, presenta variazioni di larghezza apprezzabili e potrebbe causare la terminazione prematura dell'algoritmo; per ovviare a questo vengono scartati i primi 100 pixel a partire dalla punta del dito.

Per completezza di informazione si riporta al lettore che nella figura 4.10 è stato utilizzato un valore di "*dx"* pari a 40 (in modo da poter meglio visualizzare i punti nei quali avviene il calcolo della larghezza del dito), mentre è stato verificato sperimentalmente che un risultati ottimali si ottengono con valori di *"dx"* pari a 10. Per quanto riguarda la variazione di larghezza, questa non deve essere superiore ad un fattore di 1.2; la *bounding box* così trovata risulta essere correttamente posizionata e dimensionata.

Utilizzare valori diversi per i parametri sopra citati, causa il verificarsi di errori o imprecisioni. Se la variazione di larghezza consentita fosse troppo elevata, l'algoritmo non si accorgerebbe dell'inizio della mano del soggetto e quindi si incorrerebbe nel problema esemplificato dalla figura 4.9; viceversa, se questa fosse troppo piccola l'algoritmo terminerebbe prematuramente, andando a scartare un numero troppo elevato di pixel. Avere un fattore *dx* non troppo piccolo consente di avere differenze di larghezza più dissimili fra loro e di velocizzare l'algoritmo.

In modo da includere con maggior certezza tutta la parte di immagine interessata, anche a fronte di piccole imprecisioni nelle misure, la *bounding box* viene ingrandita secondo un fattore di scala, come esemplificato dalla figura 4.11 .

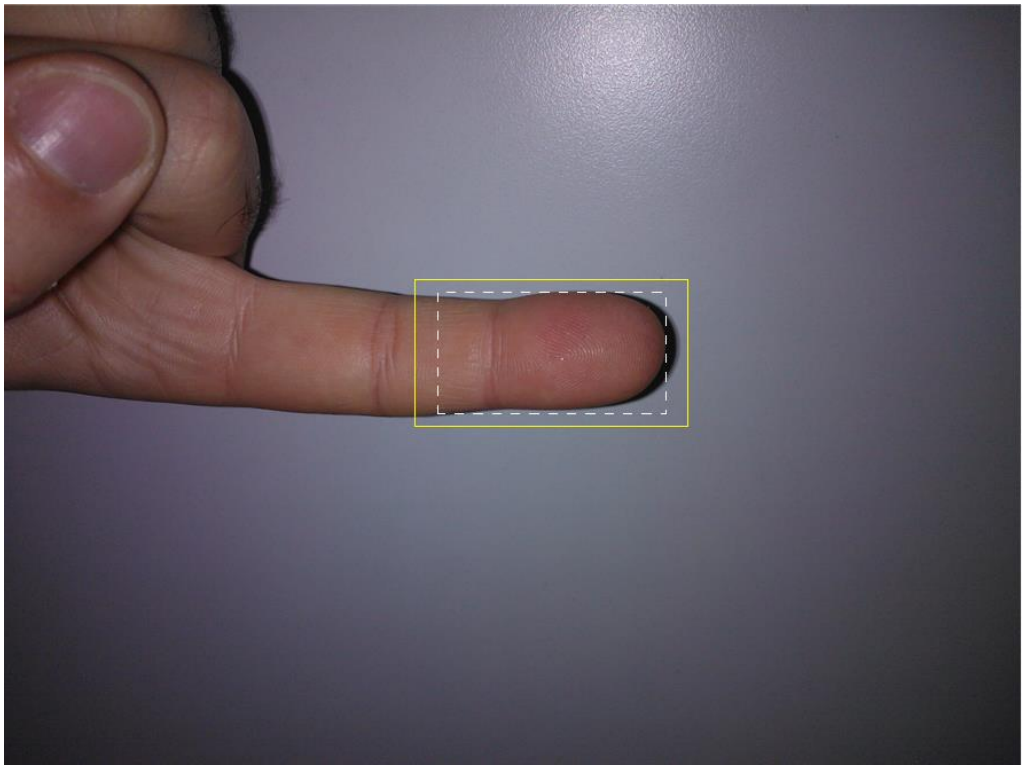

Figura 4.11 *Bounding box* originale (bianco tratteggiato) e bounding box ingrandita di un fattore di scala (giallo)

Ora che è stata trovata l'area corrispondente al dito, si ritaglia l'immagine, la si converte in scala di grigi e la si ruota in modo che il dito sia in posizione verticale, ovvero nella stessa posizione di un dito acquisito tramite scanner per impronte.

L'immagine così ottenuta tuttavia, come mostrato in figura 4.12a, contiene un alto numero di pixel di sfondo; questi potrebbero andare a peggiorare i risultati ottenuti al momento dell'estrazione delle minuzie. Si è quindi deciso di operare un ulteriore raffinamento della soluzione nel modo descritto in seguito.

In questa fase è stato riscontrato durante gli esperimenti, analizzando le *skin map* delle immagini, che molte di queste hanno bordi molto frastagliati e contengono ancora dei pixel bianchi esterni all'area del dito (figura 4.12b). Questi non portavano problemi al precedente calcolo della *bounding box* poiché sono molto vicini al dito; in questa fase, però, dove si sta cercando di calcolare la

bounding box in modo più preciso, la presenza di pixel bianchi esterni all'area del dito e i bordi frastagliati portano inevitabilmente a delle inesattezze e in alcuni casi al malfunzionamento del sistema creando *bounding box* errate.

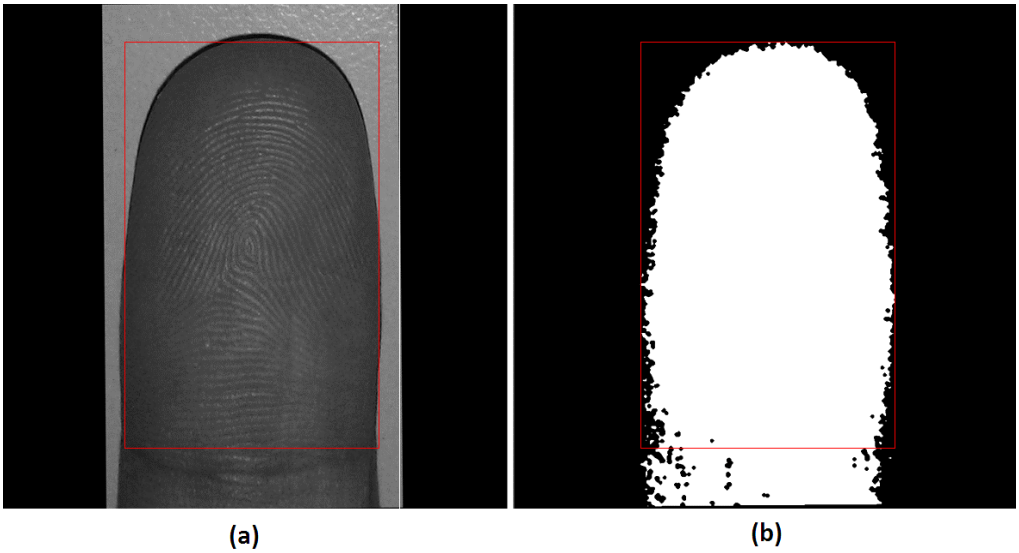

Figura 4.12 A sinistra l'immagine ritagliata e ruotata, a destra la rispettiva *skin map*

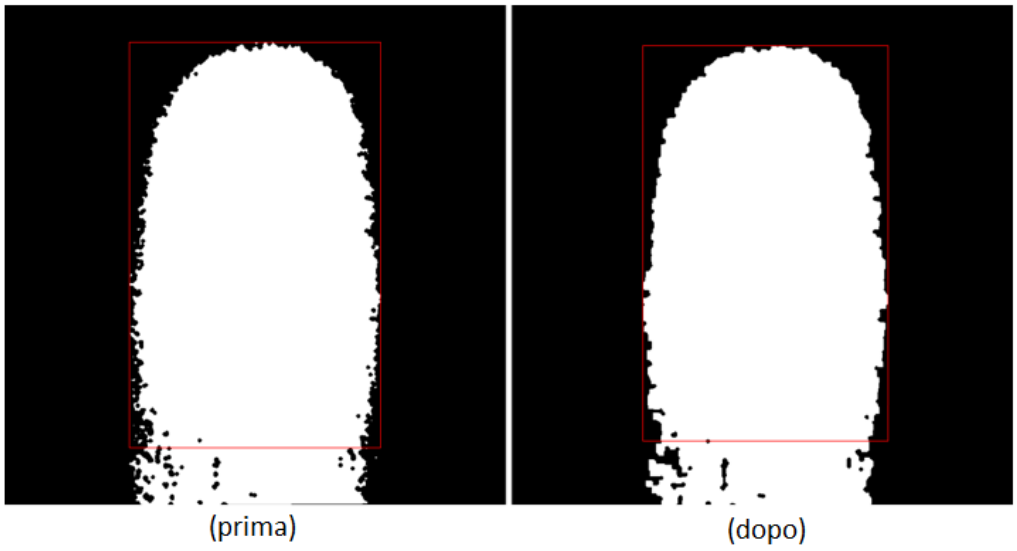

Figura 4.13 A sinistra *skin map* prima dell'operazione morfologica di apertura e dell'eliminazione delle componenti connesse, a destra *skin map* in seguito a tali operazioni. In rosso le *bouding box* delle immagini

Per ovviare a questo, viene innanzi tutto eseguita un'operazione di morfologia matematica di apertura seguita da un operazione di eliminazione delle componenti connesse; in particolare viene calcolata l'area di tutte le componenti connesse di colore bianco, quindi vengono eliminate tutte le componenti connesse eccetto quella di area massima (che corrisponde all'area del dito). Un esempio del risultato ottenuto in seguito a queste operazioni è mostrato in figura 4.13 .

In questo contesto, l'algoritmo per il calcolo della *bounding box* è del tutto similare a quanto visto nelle pagine precedenti. Innanzitutto è necessario trovare la lunghezza del dito, in modo da ottenere la coordinata y massima della *bounding box*, questo avviene scorrendo tutte le righe della *skin map* a partire dall'alto fino a quando non si trova un pixel di colore bianco. In questo caso calcolare la larghezza è molto semplice: si scorrono tutte le righe della *skin map* partendo dalla y massima precedentemente trovata e si memorizzano di volta in volta il primo e l'ultimo pixel di colore bianco di una determinata riga. La larghezza sarà data dalla seguente formula:

$$width = x_{max} - x_{min}$$

In questa fase la sua lunghezza è calcolata tramite un fattore di scala rispetto alla larghezza del dito; in particolare sono stati utilizzati i dati raccolti da Jin Chu Wu, che nel suo studio "*Statistical analysis of widths and heights of fingerprint images in terms of ages from segmentation data*", ha raccolto la lunghezza e la larghezza della falange distale di 70801 persone. Da questi dati è stata calcolata la larghezza e la lunghezza media, il loro rapporto rappresenta il fattore di scala impiegato nell'algoritmo:

$$fingerSizeRatio = \frac{avgFingerHeight}{avgFingerWidth} = \frac{2{,}31\ [cm]}{1{,}44\ [cm]} = 1{,}6041\bar{6}$$

La *bounding box* così trovata è mostrata dal riquadro rosso della figura 4.13 .

Infine viene ritagliata l'immagine, viene ribaltata orizzontalmente e viene portata, per motivazioni che saranno illustrate in seguito, alla risoluzione di 500 dpi. L'output finale dell'algoritmo è mostrato dalla figura 4.14 .

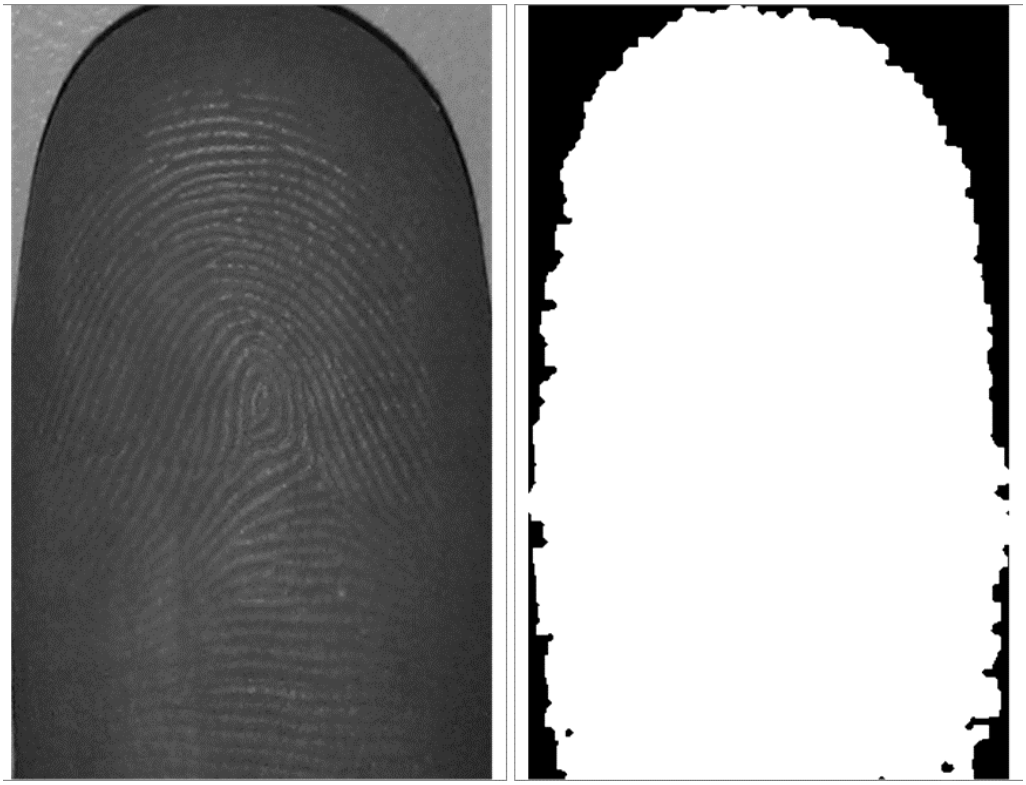

Figura 4.14 Output finale dell'algoritmo di individuazione del dito: a sinistra l'immagine della falange distale del dito a 500Dpi, a destra la sua *skin map* a 500 Dpi

Capitolo 5.
Il sistema di riconoscimento

5.1 Algoritmo base di estrazione delle minuzie

L'estrazione delle minuzie da immagini di impronte digitali è un argomento molto studiato nella lettura scientifica (Maltoni, D; Maio, D; Jain, A. K. e Prabhakar, S. *Handbook of Fingerprint Recognition*). In questo lavoro di tesi si è partiti da un algoritmo esistente, sviluppato per immagini "tradizionali" di impronte digitali (R. Cappelli; M. Ferrara; D. Maio, *A Fast and Accurate Palmprint Recognition System Based on Minutiae*) e si è cercato di estenderlo ed adattarlo a un contesto molto più complesso qual' è l'estrazione di minuzie da una fotografia di un dito. Tale algoritmo ha come input l'immagine, la sua corrispondente *skin map*, oltre a vari altri parametri che ne controllano il comportamento. L'algoritmo fornisce come output diverse informazioni, tra cui:

- Immagine delle orientazioni
- Immagine delle frequenze
- Risultato dell'enhancement
- Impronta binarizzata
- Mappa della qualità locale
- Minuzie estratte

Nel seguito sono descritti i passi principali di tale algoritmo e definiti alcuni concetti necessari per illustrarne il funzionamento (Cappelli, R; Ferrara, M; Maio, D. *A Fast and Accurate Palmprint Recognition System Based on Minutiae*).

Come prima cosa viene calcolata l'immagine delle orientazioni: l'orientazione locale in un punto [x, y] è l'angolo θ_{xy} che le *ridge line* che attraversano un intorno piccolo a piacere formano con l'asse orizzontale. Per calcolare le orientazioni viene eseguito un pre-processing dell'immagine effettuando uno smoothing Gaussiano con una maschera 3x3 seguito da un filtraggio mediano con una finestra 3x3. Le orientazioni sono stimate, utilizzando il gradiente, ad ogni nodo di una matrice quadrata con un passo di quattro pixel; ogni suo elemento viene determinato attraverso una finestra di dimensione 23x23 e consiste in un angolo $\theta_{xy} \in [0, \pi[$ e in un valore $s_{xy} \in [0,1]$ che denota la consistenza della misura dell'orientazione (ossia il suo grado di affidabilità). Infine ad ogni elemento è applicato uno smoothing attraverso una finestra 5x5. La figura 5.1b mostra le orientazioni ottenute dall'immagine della figura 5.1a .

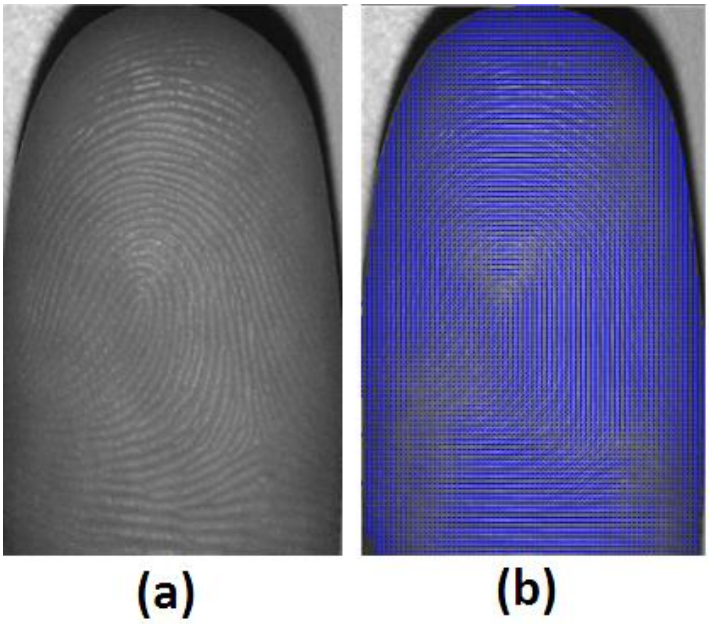

Figura 5.1 La figura (a) rappresenta l'immagine originale, la figura (b) l'immagine delle orientazioni

In seguito viene calcolata l'immagine delle frequenze: la frequenza locale nel punto [x, y] è data dal numero di *ridge* per unità di lunghezza lungo un ipotetico segmento centrato in [x, y] e ortogonale all'orientazione locale θ_{xy}. Ogni frequenza è stimata contando il numero medio di pixel tra due picchi di livello

di grigio consecutivi lungo la direzione normale all'orientazione locale della ridge. I picchi di grigio sono individuati attraverso la *x-signature*, che è ottenuta accumulando, per ogni colonna *x*, i livelli di grigio dei pixel corrispondenti nella finestra orientata. In particolare, inizialmente viene effettuato un preprocessing analogo a quanto visto nel calcolo delle orientazioni, le frequenze vengono quindi stimate in ogni elemento della matrice con un passo di 16 pixel. Per ogni pixel dove la frequenza deve essere stimata viene calcolata la *x-signature* attraverso una finestra orientata 48x16, viene effettuato uno smoothing della *x-signature* con una media locale attraverso una finestra 3x3, viene trovata la distanza media tra due picchi consecutivi e quindi stimato il periodo delle *ridge line* come media di valori non troppo lontani dalla distanza media:

$$\bar{T} = avg\left\{d_i \,|\left|d_i - \bar{d}\right| \leq 5,\ \ i = 1, \dots, n\right\}$$

dove d_i rappresenta la distanza tra due picchi consecutivi e $\bar{d}$ la distanza media. Infine si calcola la frequenza come inverso del periodo e si esegue un post-processing dove ogni frequenza che non ha potuto essere calcolata viene sostituita dalla frequenza media e dove viene effettuato due volte uno smoothing Gaussiano con finestra 3x3. La figura 5.12 mostra l'immagine delle frequenze ottenuta dalla figura 5.2a .

L'operazione di enhancement ha come obiettivo il miglioramento della qualità delle impronte, questa è attuata eseguendo un'operazione di convoluzione con filtri di Gabor. L'enhancement è guidato dalle orientazioni locali e dalle frequenze, e produce immagine quasi binaria (figura 5.3b) che in seguito è binarizzata con una soglia prefissata (figura 5.3c).

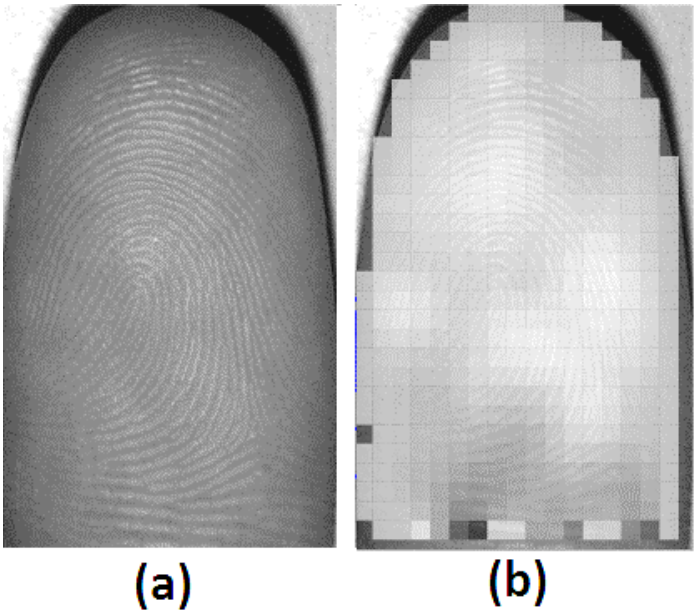

Figura 5.2 La figura (a) rappresenta l'immagine originale, la figura (b) l'immagine delle frequenze: un grigio più chiaro indica frequenze più alte.

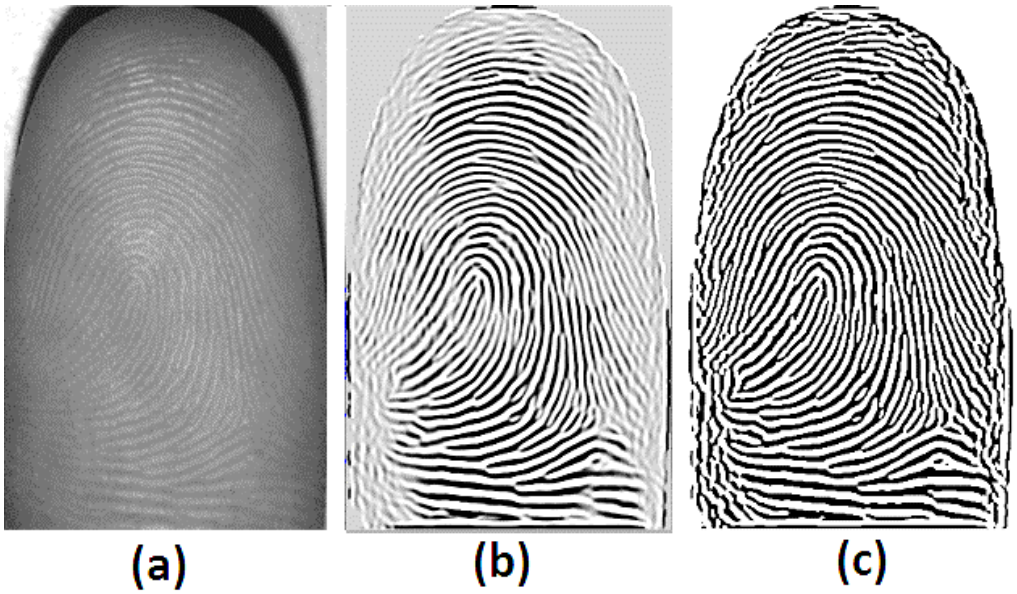

Figura 5.3 La figura (a) rappresenta l'immagine originale, la figura (b) l'immagine a seguito di un operazione di enhancement, la figura (c) l'immagine a seguito di un operazione di enhancement e di binarizzazione

Quindi viene determinata una mappa di qualità locale: essa è calcolata, in base alla consistenza delle orientazioni locali, applicando per due volte un operazio-

ne di media locale. Risultato di questo è un'immagine in bianco e nero dove ogni pixel ha un valore più o meno intenso in base alla qualità dell'immagine (figura 5.4b).

Combinando la mappa di qualità locale con e la *skin map* a blocchi passata come parametro in ingresso all'algoritmo, si determina la mappa delle regioni di bassa qualità. Essa è formata da blocchi di colore bianco in corrispondenza di zone di cattiva qualità (dove i pixel della mappa di qualità locale non superano una certa soglia) e da blocchi di colore nero in corrispondenza di zone di buona qualità (figura 5.4c).

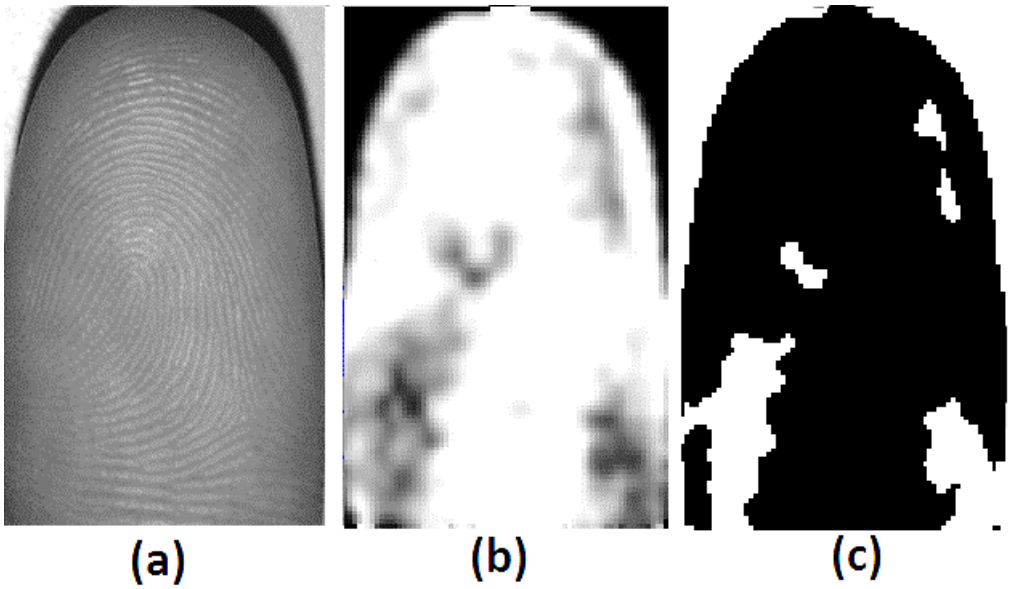

Figura 5.4 La figura (a) rappresenta l'immagine originale, la figura (b) la mappa di qualità locale, la figura (c) la mappa di qualità a blocchi binaria, si noti come le porzioni di immagine in bianco rappresentano le zone di scarsa qualità

Una volta eseguite tutte queste operazioni è possibile estrarre le minuzie. L'algoritmo prevede due possibilità: estrarre tutte le minuzie (come mostrato nella figura 5.5a) oppure estrarre solo quelle che ricadono in zone giudicate di buona qualità (come mostrato nella figura 5.5b).

Per estrarre le minuzie viene effettuato un thinning dell'immagine binaria dell'enhancement (figura 5.3c) in modo da ottenere il *ridge line skeleton*. Lo stesso algoritmo è utilizzato per ottenere il *valley skeleton*. Quindi le minuzie

sono estratte da entrambi gli *skeleton*; solo le minuzie estratte dal *ridge line skeleton* che hanno una controparte (o un tipo complementare) nel *valley skeleton* sono considerate valide.

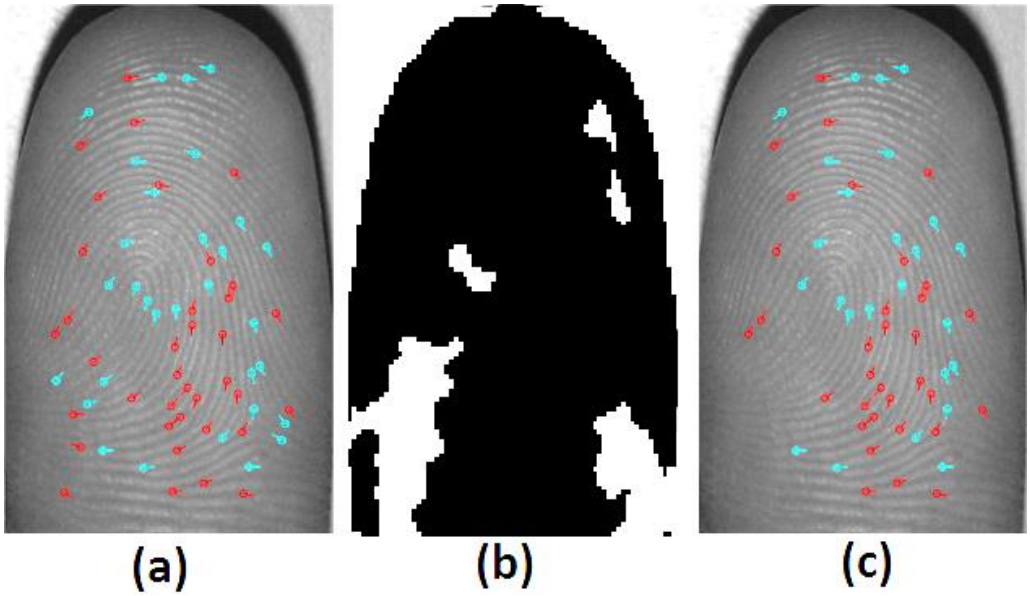

Figura 5.5 Estrazione di tutte le minuzie (figura a), mappa di qualità a blocchi binaria (figura b), estrazione delle sole minuzie nelle zone di buona qualità (figura c). Si noti come le minuzie che si trovano nelle zone di cattiva qualità (bianco) vengono eliminate

5.2 Algoritmo base di confronto impronte

Per confrontare due impronte digitali viene utilizzato l'algoritmo *MCC*: *Minutia Cylinder Code*. Tale algoritmo rappresenta le minuzie utilizzando particolari strutture locali (chiamate *cilindri*), calcola il grado di similarità tra due cilindri ed infine determina un punteggio di match globale che indica il grado di somiglianza tra i due insiemi di minuzie.

Nella sezione seguente viene descritto brevemente l'algoritmo *MCC* (Cappelli, R; Ferrara, M; Maltoni, D. *MCC: a baseline algorithm for fingerprint verification in FVC-onGoing*, *Minutia Cylinder-Code: a new representation and matching technique for fingerprint recognition*).

5.2.1 Strutture locali

Sia T un *template* di minuzie, in cui ogni minuzia m è definita come $m = \{x_m, y_m, \theta_m\}$, dove x_m e y_m sono la posizione della minuzia, mentre θ_m la sua direzione (nell'intervallo $[-\pi, \pi]$). La rappresentazione *MCC* associa una struttura locale ad ogni minuzia: questa struttura codifica le relazioni spaziali e direzionali tra la minuzia considerata e le minuzie vicine, e può essere convenzionalmente rappresentata come un cilindro di raggio R e altezza 2π centrato nella posizione della minuzia, come mostrato dalla figura 5.6 .

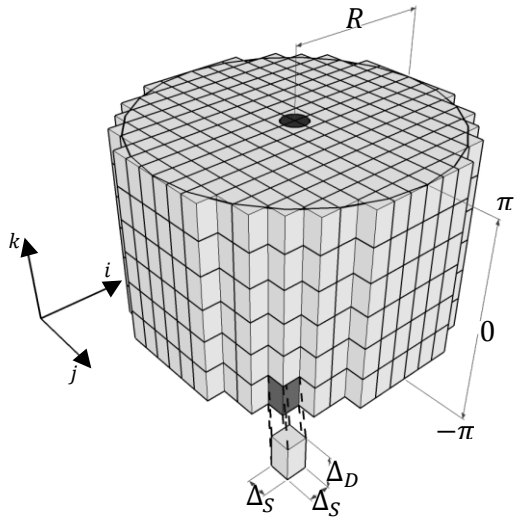

Figura 5.6 Una rappresentazione grafica della struttura dati associata ad una minuzia nella rappresentazione MCC. Il cilindro è formato da celle di base $\Delta_S \times \Delta_S$ e altezza Δ_D.

Il cilindro è diviso in un certo numero di sezioni, ognuna delle quali corrisponde a un intervallo di differenze di direzione in $[-\pi, \pi]$; le sezioni sono discretizzate in celle. Per ogni cella valida, viene calcolato un valore numerico accumulando contributi dalle minuzie vicine m_t sulla proiezione del centro della cella sulla base del cilindro. Il contributo di ogni minuzia m_t su una determinata cella (del cilindro corrispondente ad una data minuzia m) dipende da:

- Quanto la minuzia m_t è vicina al centro della cella.

- Quanto la differenza di direzione tra la minuzia m_t e la minuzia m è simile alla differenza di direzione associata alla sezione dove la cella risiede.

Il valore della cella rappresenta dunque la probabilità di trovare minuzie che gli siano vicine la cui differenza di direzione rispetto ad m è simile a un certo valore. Una cella è considerata valida se e solo se è contenuta nell'intersezione della base del cilindro con l'inviluppo convesso determinato da tutte le minuzie in *T*. La figura 5.7 mostra il cilindro associato ad una minuzia avente cinque minuzie nel suo intorno.

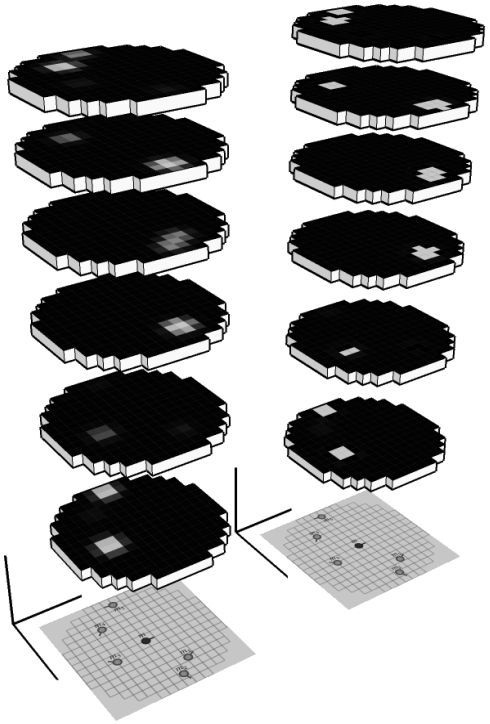

Figura 5.7 A sinistra una rappresentazione grafica di un cilindro: le zone chiare rappresentano alti valori; la minuzia corrispondente e le minuzie nel suo intorno sono mostrate sotto la base del cilindro. A destra lo stesso cilindro utilizzando una rappresentazione a bit (nero = 0, bianco = 1)

Dato un *template T*, l'algoritmo crea un insieme di cilindri che contiene i cilindri associati a tutte le minuzie in *T* con un sufficiente numero di minuzie vicine. Ogni cilindro è una struttura locale che:

- È invariante per traslazione e rotazione, dato che codifica solamente distanze e differenze di direzione tra minuzie e la sua base è ruotata in base alla sua corrispondente minuzia.
- È robusta contro la distorsione della pelle e contro piccoli errori di estrazione di caratteristiche.
- Ha una lunghezza fissa, data dal numero totale di celle nel cilindro.

5.2.2 Implementazione binaria e similarità locale

Un vantaggio della rappresentazione *MCC* è che ogni cilindro può essere memorizzato come vettore di bit, con una perdita di precisione trascurabile (figura 5.7). Ogni elemento del cilindro $C_m(i,j,k)$, corrispondente ad una data minuzia m, può assumere i valori 0, 1 o invalido. Esso viene considerato come una coppia di vettori binari $\boldsymbol{c}_m$, $\hat{\boldsymbol{c}}_m \in \{0,1\}^n$, ottenuto linearizzando i valori delle celle:

$$\boldsymbol{c}_m[lin(i,j,k)] = \begin{cases} 1 & \Leftrightarrow\ C_m(i,j,k) = 1 \\ 0 & altrimenti \end{cases}$$

$$\hat{\boldsymbol{c}}_m[lin(i,j,k)] = \begin{cases} 1 & \Leftrightarrow\ C_m(i,j,k) \neq invalido \\ 0 & altrimenti \end{cases}$$

in pratica il vettore $\hat{\boldsymbol{c}}_m$ è utilizzato come una maschera di bit per selezionare i bit validi nel vettore $\boldsymbol{c}_m$. Vengono definiti, quindi, i seguenti vettori:

$$\boldsymbol{c}_{a|b} = \boldsymbol{c}_a\ AND\ \hat{\boldsymbol{c}}_{ab}$$

$$\boldsymbol{c}_{b|a} = \boldsymbol{c}_b\ AND\ \hat{\boldsymbol{c}}_{ab}$$

dove $\hat{\boldsymbol{c}}_{ab} = \hat{\boldsymbol{c}}_a\ AND\ \hat{\boldsymbol{c}}_b$ rappresenta l'intersezione tra le due maschere. Infine il grado di similarità tra due vettori è dato da:

$$\gamma_{Bit}(a,b) = \begin{cases} 1 - \dfrac{\|\boldsymbol{c}_{ab}\ XOR\ \boldsymbol{c}_{ba}\|}{\|\boldsymbol{c}_{a|b}\| + \|\boldsymbol{c}_{b|a}\|} \Leftrightarrow \boldsymbol{c}_a\ e\ \boldsymbol{c}_b\ sono\ abbinabili \\ 0 \quad altrimenti \end{cases}$$

Utilizzare dei vettori di bit consente di implementare l'algoritmo in modo molto efficiente.

5.2.3 Punteggio globale (*match score*)

Per confrontare due impronte digitali viene determinato un punteggio globale che indica la loro similarità complessiva. Benché tale punteggio possa essere determinato combinando solamente similarità locali, risultati più accurati si ottengono implementando una fase di consolidamento per ottenere un punteggio globale che riflette in quale misura le relazioni locali fra minuzie valgono anche su scala globale. In questo lavoro di tesi è stato utilizzato una tecnica di questo tipo per calcolare i match score, al fine di ottenere un riconoscimento biometrico il più preciso possibile.

Sebbene l'algoritmo di estrazione descritto nella sezione precedente e l'algoritmo di confronto descritto in questa sezione siano in grado di operare anche su immagini di qualità medio/bassa, durante i test sono stati riscontrate numerose problematiche legate al particolare tipo di immagini utilizzate in questo lavoro (per ulteriori informazioni vedere il capitolo 6). È stato quindi necessario operare su fronti diversi:

1. Trovare un metodo automatico che permetta di discernere immagini a fuoco da immagini sfuocate, in modo da concentrare l'estrazione delle minuzie sulle sole immagini a fuoco.
2. Migliorare le prestazioni dell'algoritmo cercando di portare l'immagine acquisita con Smartphone alla stessa risoluzione di quella acquisita con scanner.
3. Scartare minuzie troppo vicine al bordo del dito.

Le sezioni seguenti descrivono il lavoro che è stato svolto per raggiungere tali obiettivi.

5.3 Selezione automatica dei frame

Durante alcuni test preliminari è stato riscontrato un elevato tasso di errore, causato principalmente dalla presenza di immagini con qualità estremamente bassa. In un contesto applicativo non è possibile sapere a priori se un'immagine è di buona qualità o meno: è quindi necessario studiare un metodo che permetta di discernere in modo automatico le fotografie a fuoco da quelle sfuocate. A tale fine, durante questo lavoro di tesi sono state studiate e sperimentate varie tecniche, brevemente descritte nel seguito.

Un primo tentativo ha visto il calcolo della convoluzione dell'immagine con un filtro di sharpening, al fine di stimare il livello di messa a fuoco come media del valore assoluto di tutti i pixel. Tale procedimento, tuttavia, anche effettuando elaborazioni intermedie sulle immagini, quali l'equalizzazione dell'istogramma, non ha portato a risultati apprezzabili. In seguito, sono stati effettuati altri tentativi basati sul *TSI ("Top Sharpening Index")*, tale indice è una misura proposta per valutare la messa a fuoco di un impronta digitale basandosi sulla ripidezza delle transizioni *ridge/valley* (Ferrara, M; Franco, A; Maltoni, D. *Fingerprint scanner focusing estimation by Top Sharpening Index*), e sulla segmentazione, ma nuovamente con scarso successo.

Si è scelto quindi di sfruttare l'algoritmo utilizzato per scartare le minuzie corrispondenti alle zone di immagine di cattiva qualità. Tale algoritmo divide l'immagine in blocchi, calcola la qualità di ogni blocco, e lo marca come di buona o di cattiva qualità. L'indice di qualità è stato calcolato contando il numero di blocchi di buona qualità e dividendo tale valore per il numero totale di blocchi. Tale metodo ha portato a un netto miglioramento nei risultati. Nella figura 5.8 è possibile notare come l'indice così ottenuto abbia un valore basso

in un immagine di bassa qualità e un valore più alto in un'immagine di media qualità.

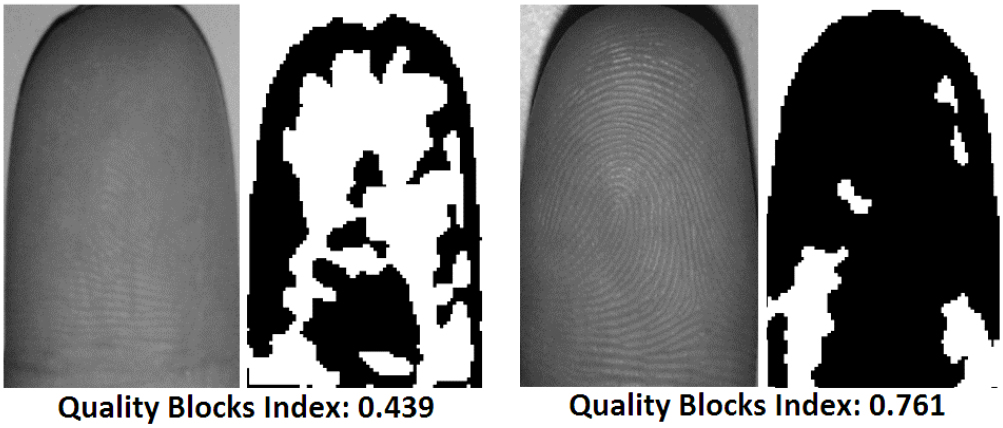

Figura 5.8 A sinistra, immagine di cattiva qualità con relativa quality skin map blocks; A destra immagine di buona qualità. Si noti la differenza tra i due indici di qualità.

È stato sperimentato infine anche un altro indice di qualità delle impronte sviluppato all'interno al *Biometric System Laboratory*, basato sulla consistenza delle orientazioni e altre informazioni locali; tale indice è risultato essere il più performante. In questo caso la qualità viene rappresentata da un valore tra 0 e 100; in particolare, facendo riferimento alle immagini della figura 5.8, queste hanno rispettivamente un indice di qualità pari a "10" e "59". Tale indice, benché in grado di individuare molte immagini di bassa qualità, non riesce a discernere alcune immagini sfuocate da immagini a fuoco. Tali immagini, identificate in precedenza manualmente (circa il 27%), sono state scartate durante i test, delegando ad eventuali sviluppi futuri lo studio di un algoritmo che sia in grado di scartarle in modo automatico.

Per ogni immagine non sfuocata viene dunque determinata la qualità: durante i test del sistema, per ogni dito sono stati selezionati gli *"n"* frame di miglior qualità, dove *"n"* è un parametro del sistema.

5.4 Ridimensionamento automatico delle immagini

Durante alcuni test preliminari, controllando i punteggi di match ottenuti da impronte dello stesso dito, è stato notato che questi hanno valori molto bassi, anche in immagini di buona qualità (la figura 5.9 riporta un esempio).

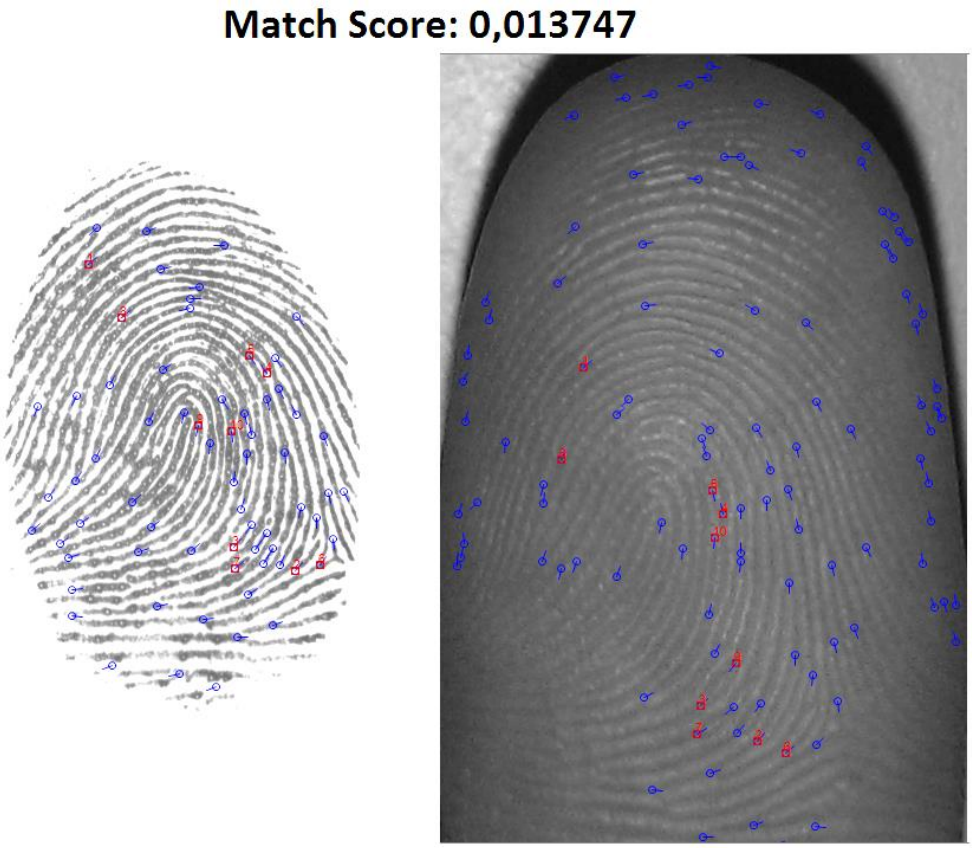

Figura 5.9 Estrazione delle minuzie (nelle sole zone di buona qualità) e punteggio di match relativo.

Analizzando il problema ci si è resi conto che un fattore che va a influire negativamente sui risultati ottenuti è la diversa risoluzione delle immagini. In particolare lo scanner per impronte digitali ha una risoluzione nota, ovvero 500 dpi, mentre le immagini, in seguito al loro ritaglio in corrispondenza della *bounding box*, possono avere una diversa risoluzione, non nota a priori.

Una prima soluzione a tale problema è stata sviluppata mediante ridimensionamento dell'immagine di un fattore dato dalla seguente formula:

$$resizeFactor = \frac{(500/2.54) \cdot avgFingerWidth}{Image\ Width}$$

dove "*avgFingerWidth*" è la larghezza media della falange distale calcolata secondo i dati di Jin Chu Wu menzionati precedentemente.

In seguito a questa modifica dell'algoritmo si ottiene un netto miglioramento; nell'immagine riportata come esempio si ha un incremento del punteggio di match pari al 217%. I risultati sono riportati nella figura 5.10; si noti come anche la posizione delle minuzie risulta essere diversa rispetto alla figura 5.9 .

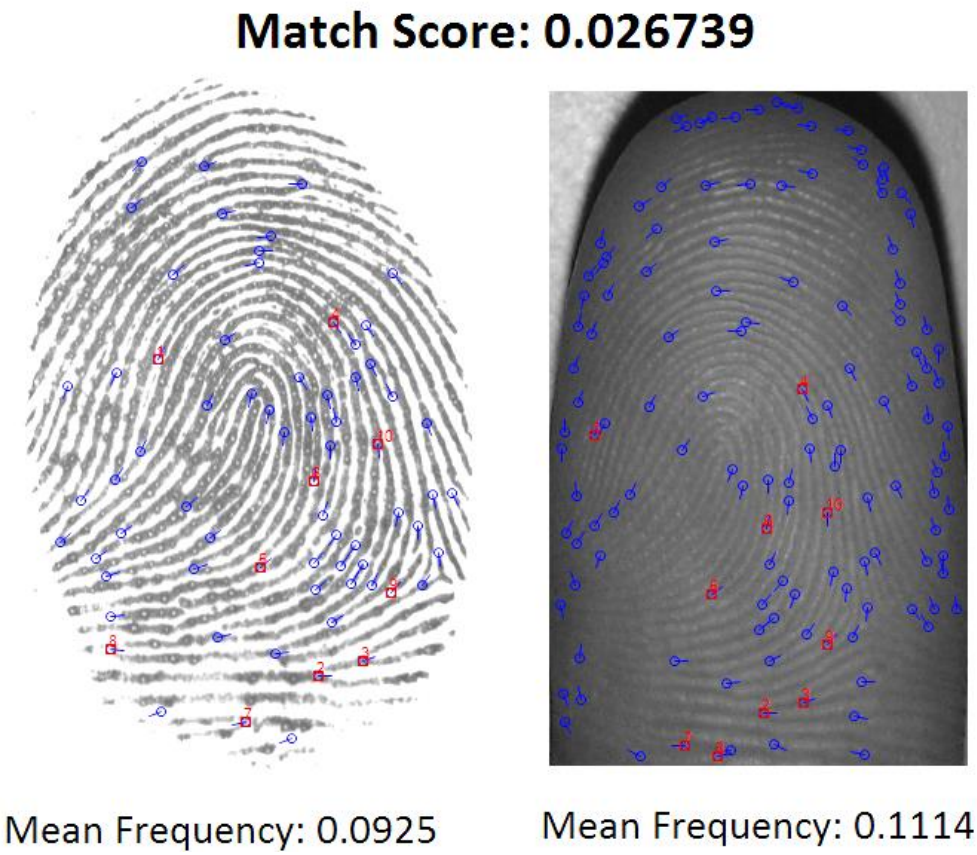

Figura 5.10 Estrazione delle minuzie dopo il ridimensionamento a 500 dpi

Una ulteriore è più dettagliata analisi ha evidenziato come la frequenza media delle *ridge line* dell'immagine di un'impronta acquisita tramite scanner è, no-

nostante il ridimensionamento applicato, in generale abbastanza diversa da quella delle immagini dello stesso dito acquisite tramite Smartphone. Si è studiato quindi un metodo per cercare di portare la frequenza media dell'immagine acquisita tramite Smartphone ad un valore il più prossimo possibile a quella dell'immagine acquisita tramite scanner.

A tal fine, dopo varie sperimentazioni, si è deciso di utilizzare un procedimento iterativo: si ridimensiona l'immagine finché la differenza tra la sua frequenza media e quella dell'immagine acquisita tramite scanner non è minore di una determinata soglia. Sebbene questo procedimento funzioni molto bene per alcune immagini, vi sono numerosi casi (specialmente nelle immagini di scarsa qualità) dove si ha un peggioramento del punteggio di match. Nel seguito vengono illustrati alcuni esempi.

Si consideri il caso in cui si voglia che la frequenza delle immagini acquisite tramite Smartphone tenda a quella delle immagini acquisite tramite Smartphone; si sceglie quindi un valore della soglia relativa alla differenza di frequenza molto basso. A titolo di esempio, si sceglie un immagine di scarsa qualità.

Max Error:	0,0001
Plain Freq:	0,108878178

Tabella 5.1 Parametri di Input all'algoritmo

I risultati ottenuti, mostrati nella tabella 5.2, mettono in luce diversi aspetti:

- Non è detto che aumentando le iterazioni il punteggio di match debba per forza aumentare; infatti, nel seguente esempio inizialmente il punteggio di match aumenta ma alla quarta iterazione cala improvvisamente.
- Non è detto che aumentando le iterazioni l'errore debba per forza diminuire (notare iterazioni quattro, sei, otto).

Numero Iterazioni	Smartphone Freq	Errore	Punteggio di Match
1	0,12181	0,01293	0,022934
2	0,11446	0,00558	0,034213
3	0,11132	0,00244	0,038472
4	0,11163	**0,00275**	**0,019679**
5	0,10868	0,00020	0,019679
6	0,10998	0,00110	0,019679
7	0,10917	0,00029	0,019679
8	0,10832	0,00055	0,019679
9	0,10851	0,00037	0,021372
10	0,10883	0,00005	0,021372

Tabella 5.2 Risultati ottenuti

Per risolvere queste problematiche viene fissato un numero massimo di iterazioni da far eseguire all'algoritmo, in modo da non causare l'abbassamento del punteggio di match. Per evitare un eccessivo ridimensionamento delle immagini, inoltre, viene anche fissato un valore massimo di ridimensionamento. L'algoritmo è esemplificato dal diagramma di flusso della figura 5.11 .

Un aspetto implementativo da considerare, inoltre, è il fatto che i continui ridimensionamenti a cui vengono sottoposte le immagini causano un progressivo degrado della qualità delle immagini, e quindi un conseguente peggioramento dei punteggi di match. Per ovviare a questo vengono utilizzate delle liste che andranno a contenere immagine, *skin map* ed errore relativi ad ogni iterazione dell'algoritmo, per ogni nuova immagine prodotta viene quindi effettuato un solo ridimensionamento. Al termine dell'algoritmo viene selezionata l'immagine affetta da errore minore, e su questa vengono estratte le minuzie e calcolato il punteggio di match.

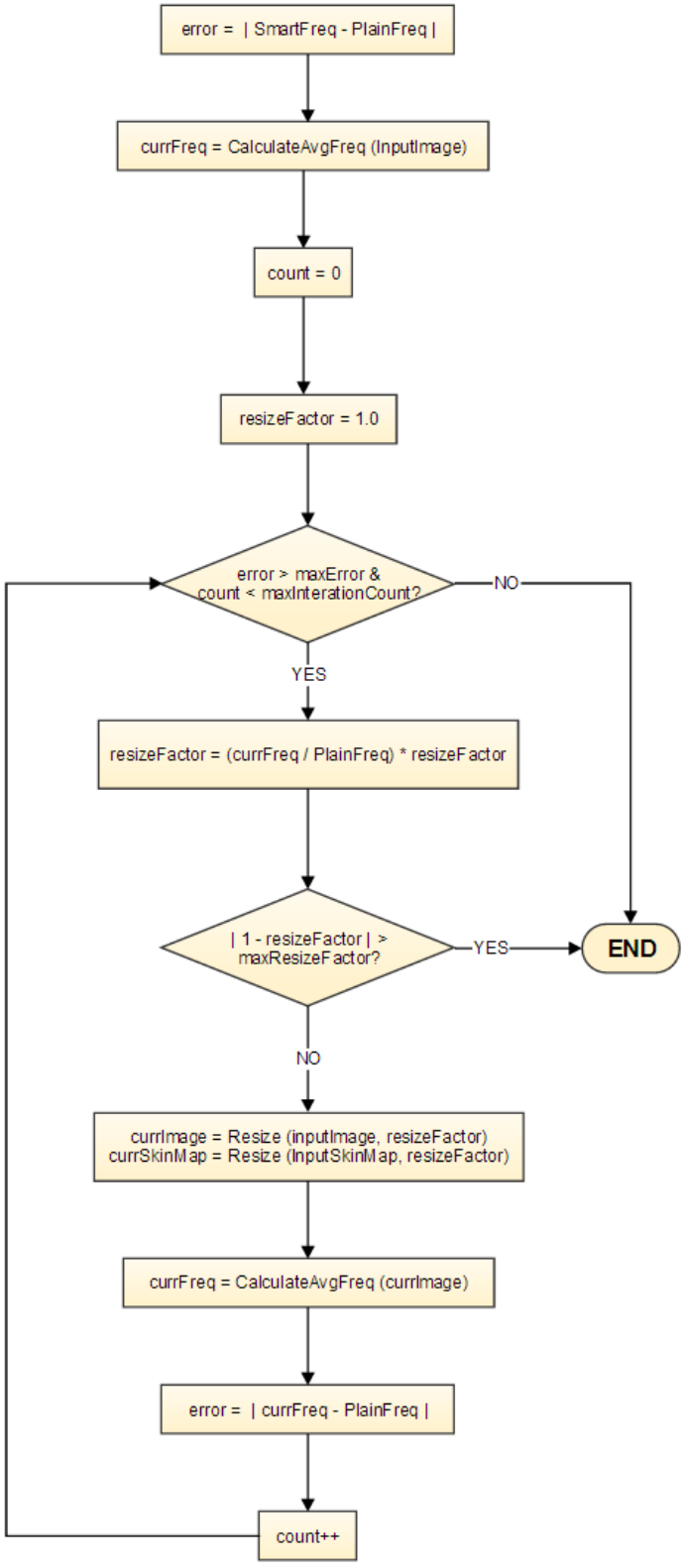

Figura 5.11 Diagramma di flusso dell'algoritmo di ridimensionamento delle immagini in base alla frequenza media

5.5 Eliminazione delle minuzie di bordo

In ultima analisi è stato notato che, mentre nelle immagini acquisite tramite scanner il dito viene "schiacciato" su di esso (quindi con una perfetta proiezione sul piano di acquisizione di tutte le minuzie), nelle immagini acquisite tramite Smartphone, le zone periferiche del dito presentano un elevato raggio di curvatura che va ad alterare la qualità dell'immagine modificando considerevolmente la posizione delle minuzie. Per ovviare a questa problematica, in questo lavoro vengono semplicemente scartati un certo numero di pixel da ogni bordo del dito, andando ad estrarre le minuzie solamente sulla parte restante dell'immagine; questo porta ad un miglioramento del punteggio di match, come mostrato nella figura 5.12 .

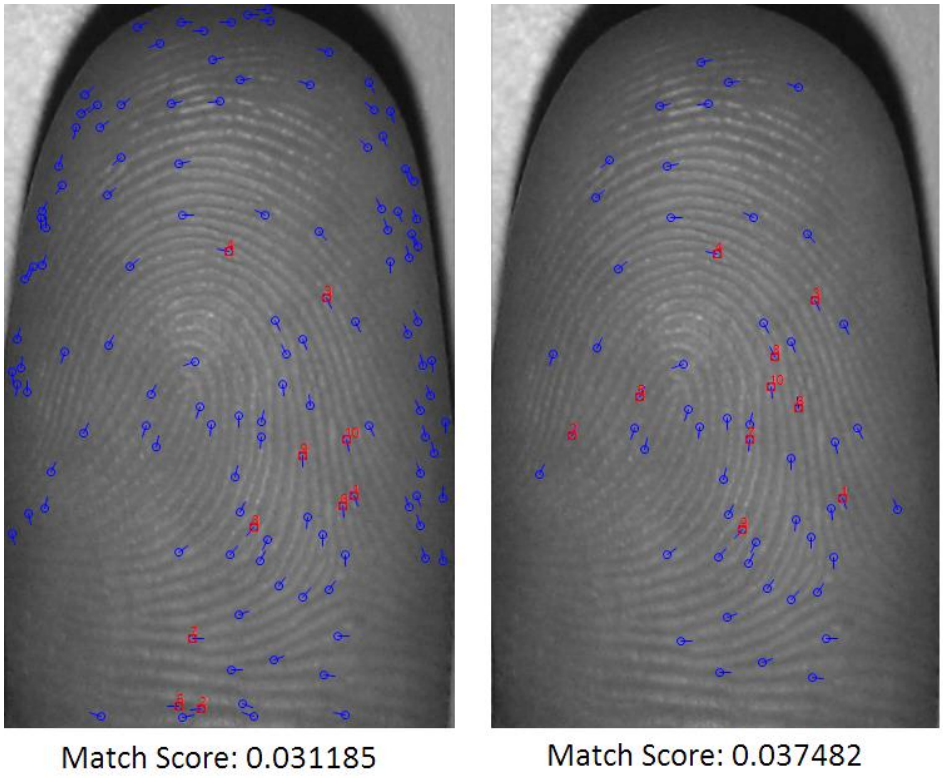

Figura 5.12 A sinistra immagine con tutte le minuzie estratte, a destra immagine senza le minuzie in prossimità dei bordi

La figura 5.13, mostra il risultato finale ottenuto dall'algoritmo. Si noti come la differenza tra le due frequenze diminuisce notevolmente (ma non è infinitesima) e che il punteggio di match migliora considerevolmente.

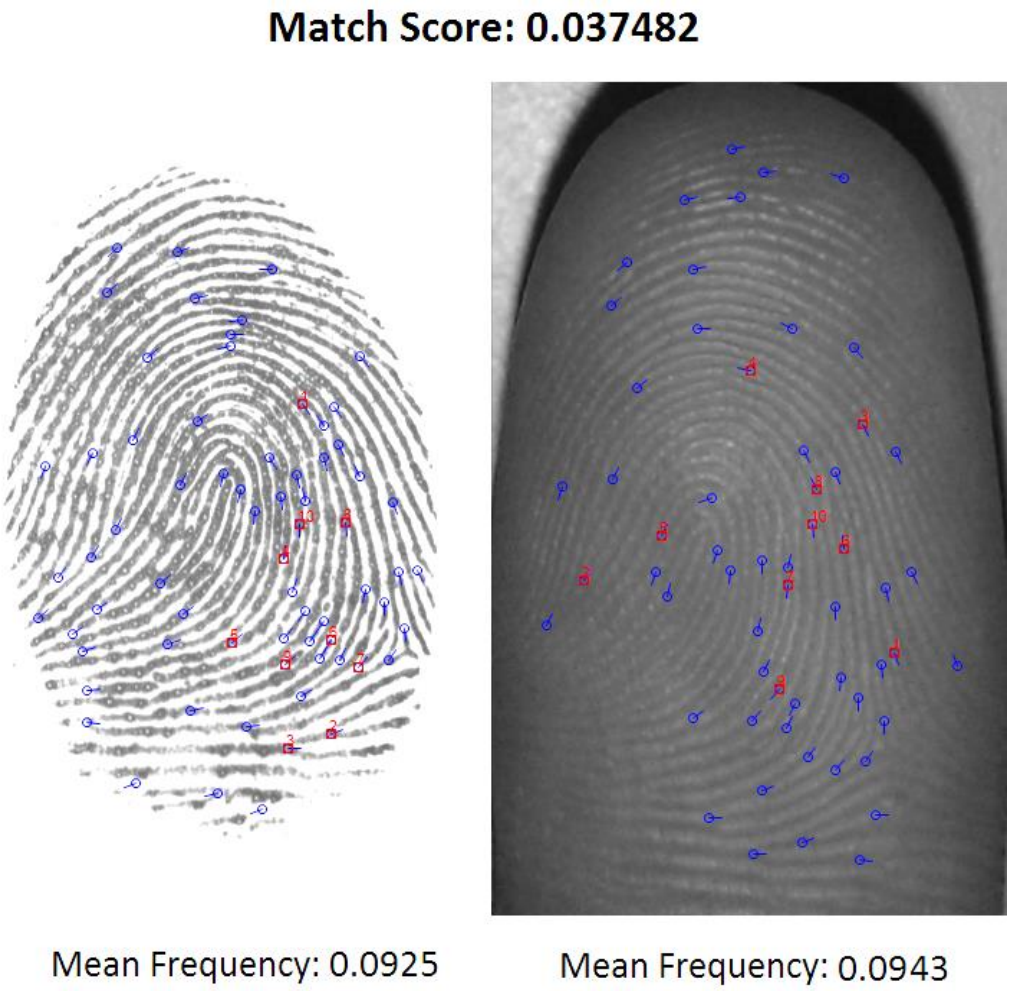

Figura 5.13 A sinistra, minuzie estratte dall'immagine acquisita tramite scanner per impronte digitali, a destra minuzie estratte dall'immagine acquista tramite Smartphone ridimensionata in base alla frequenza media dell'immagine acquisita tramite scanner.

Capitolo 6.
Risultati sperimentali

6.1 Programma di test del sistema

Per valutare le performance del sistema è necessario misurare l'errore da esso compiuto in seguito ad un certo numero di confronti. Questo al fine di rispondere alle seguenti domande: qual è la probabilità che il sistema non riconosca l'impronta digitale di un soggetto presente all'interno del database? E qual è la probabilità che un individuo le cui impronte non sono presenti all'interno del database venga erroneamente riconosciuto?

Per rispondere a queste domande è necessario effettuare dei confronti "genuine" ed "impostor" e valutare il corrispondente punteggio di match. I confronti "*genuine*" vengono effettuati tra due impronte dello stesso dito: in questo lavoro di tesi quindi tra un'immagine acquisita con Smartphone e quella (relativa allo stesso dito) acquisita tramite scanner. I confronti "*impostor*", invece, vengono effettuati tra impronte di dita diverse.

È stato necessario sviluppare un apposito programma di test del sistema che esegua queste operazioni in maniera automatica, esegua tutti i confronti previsti e calcoli l'errore commesso dal sistema. A tale fine è stata utilizzata la libreria *MCC SDK v1.3*, che consente di sviluppare applicazioni di verifica di impronte digitali utilizzando l'algoritmo *MCC* ("*Minutia Cylinder Code"*), descritto nel paragrafo 5.2 .

Nel programma sviluppato, vista l'enorme mole di calcoli che il software di test dovrà effettuare, viene utilizzato ampiamente il multithreading.

Per ogni immagine selezionata viene determinata la *bounding box*, vengono estratte le minuzie e vengono effettuati tutti i confronti "*genuine*" ed "*impostor*" determinandone così i punteggi di match. Per ogni dito (avente *"n"* frame selezionati in base alla qualità), vengono scelti gli *"nScore"* punteggi migliori, dove *"nScore"* è un parametro del sistema.

A partire dai confronti "*genuine*" e "*impostor*", si misurano i seguenti due tipi di errori:

- *FMR*: la percentuale di impronte di dita diverse che sono erroneamente considerate dello stesso dito.
- *FNMR*: la percentuale di impronte dello stesso dito che sono erroneamente considerate di dita diverse.

Tali errori variano a seconda della soglia di sicurezza del sistema. Infine, si calcolano gli indicatori riassuntivi *EER* (il valore *FMR* = *FNMR*) e *FMR100* (il valore di *FNMR* alla soglia di sicurezza per cui *FMR* = 1%).

Il programma di test è stato di utilità fondamentale, non solo per valutare l'efficienza del sistema, ma anche per individuare gli errori presenti negli algoritmi. In particolare, per ogni test eseguito, il programma salva su file tutti gli score "*genuine*" e tutti gli score "*impostor*", consentendo una successiva analisi dei risultati e degli errori.

6.2 Parametri del sistema

Si elencano di seguito i parametri che influenzano maggiormente i risultati ottenuti:

Nome	Descrizione
validSkinMapBlockPerc	Utilizzato per ottenere la *skin map* a blocchi. Rappresenta la percentuale di pixel appartenenti alla zona del dito

	(bianca) minima necessaria che deve avere un blocco affinché esso sia considerato come interno al dito.
maxFrequencyError	Differenza massima tollerata tra la frequenza media delle immagini acquisite tramite Smartphone e quella delle immagini acquisite tramite scanner
MinutiaeMinimumBorderDistance	Numero di pixel che vengono scartati dal bordo del dito durante la fase di estrazione delle minuzie
MaxFrequencyIterationCount	Numero di interazioni massime dell'algoritmo di ridimensionamento basato sulla frequenza
n	Numero di immagini (ordinate in base alla qualità) acquisite tramite Smartphone da considerare per ogni utente
nScore	Numero di score (ordinati in ordine decrescente) da considerare
scoreCombinationTipe	Utilizzato per il calcolo dello score totale di un determinato dito, può assumere i valori *MAX, AVG, PRODUCT*

Tabella 6.1 Parametri del sistema

Le impronte acquisite durante la prima sessione sono state utilizzate per una serie di test che hanno consentito di calibrare i vari parametri, con i quali è stato eseguito poi un test finale sulle impronte della seconda sessione. La tabella 6.2 mostra i valori finali assegnati ai parametri.

n	3
nScore	1
Minutiae Min Border Distance	30
Valid SkinMapBlock Perc	0,1
Max Freq Interation Count	5
Freq Error	0,0025
Max Resize Factor	0,5
Combination Type	MAX

Tabella 6.2 Valori finali assegnati ai parametri del sistema

6.3 Prestazioni del sistema

Utilizzando immagini di alta qualità, acquisite tramite appositi scanner, l'ordine di grandezza degli errori dei sistemi di riconoscimento è dell'ordine dello 0.01% o inferiore. La situazione cambia completamente quando si utilizzano immagini come quelle considerate in questo lavoro di tesi: con tali tipologie di immagini, secondo l'attuale letteratura scientifica, errori inferiori al 5% sono accettabili.

Si consideri che un'applicazione diretta degli algoritmi di base, utilizzando una *bounding box* non perfettamente dimensionata porta ad un *EER* del 45% (appena poco meglio di quello che si otterrebbe gettando in aria una moneta).

Utilizzando un indice di qualità, in particolare l'indice basato sulla *skin map* a blocchi (definito nel paragrafo 5.3.2), l'errore *EER* già diminuisce e diventa del 38%.

Lanciando il programma di test sulle sole immagini marcate manualmente come a fuoco l'errore si riduce notevolmente, come riportato in tabella 6.3 .

Sessione	EER	FMR100
1	14,31%	23,69%

Tabella 6.3 Risultati ottenuti in fase di test

In seguito al miglioramento del calcolo della *bounding box* (illustrato in sezione 4.3) si ottengono i risultati riportati in tabella 6.4 .

Sessione	EER	FMR100
1	14,60%	22,04%

Tabella 6.4 Risultati ottenuti in fase di test

Portando le immagini a 500 dpi (come illustrato in sezione 5.4), non considerando le minuzie presenti ai bordi del dito (sezione 5.5), eseguendo il test so-

lamente sulle immagini selezionate dall'algoritmo di selezione degli *"n"* frame di miglior qualità (sezione 5.3) si ottengono i risultati in tabella 6.5 al variare del parametro *nScore*.

Sessione	*n*	*nScore*	*Minutiae Minimum Border Distance*	EER	FMR100
1	10	10	25	20,00%	41,47%
1	3	1	25	13,46%	30,77%

Tabella 6.5 Risultati ottenuti in fase di test

Il ridimensionamento iterativo delle immagini in base alla frequenza media (sezione 5.4) porta notevoli miglioramenti al sistema; si ottengono infatti i risultati riportati in tabella 6.6 .

Sessione	*n*	*nScore*	*Minutiae Minimum Border Distance*	EER	FMR100
1	3	1	25	7,69%	9,62%

Tabella 6.6 Risultati ottenuti in fase di test

Andando a migliorare il calcolo della *bounding box*, si ottengono i risultati riportati in tabella 6.7 .

Sessione	*n*	*nScore*	*Minutiae Minimum Border Distance*	EER	FMR100
1	3	1	25	6,62%	7,69%

Tabella 6.7 Risultati ottenuti in fase di test

Utilizzando l'insieme degli algoritmi sviluppati, si ottengono i risultati finali riportati in tabella 6.8 .

Sessione	*EER*	*FMR100*
1	1,92%	3,85%
2	1,92%	5,77%

Tabella 6.8 Risultati finali ottenuti

Nelle figure 6.1 e 6.2 vengono riportati i grafici DET (*Detection Error TradeOff)* della sessione 1 e della sessione 2. Tali grafici mostrano come variano i due errori *FMR* e *FNMR* a seconda delle possibili soglie di sicurezza del sistema: si noti che le scale dei due assi sono logaritmiche.

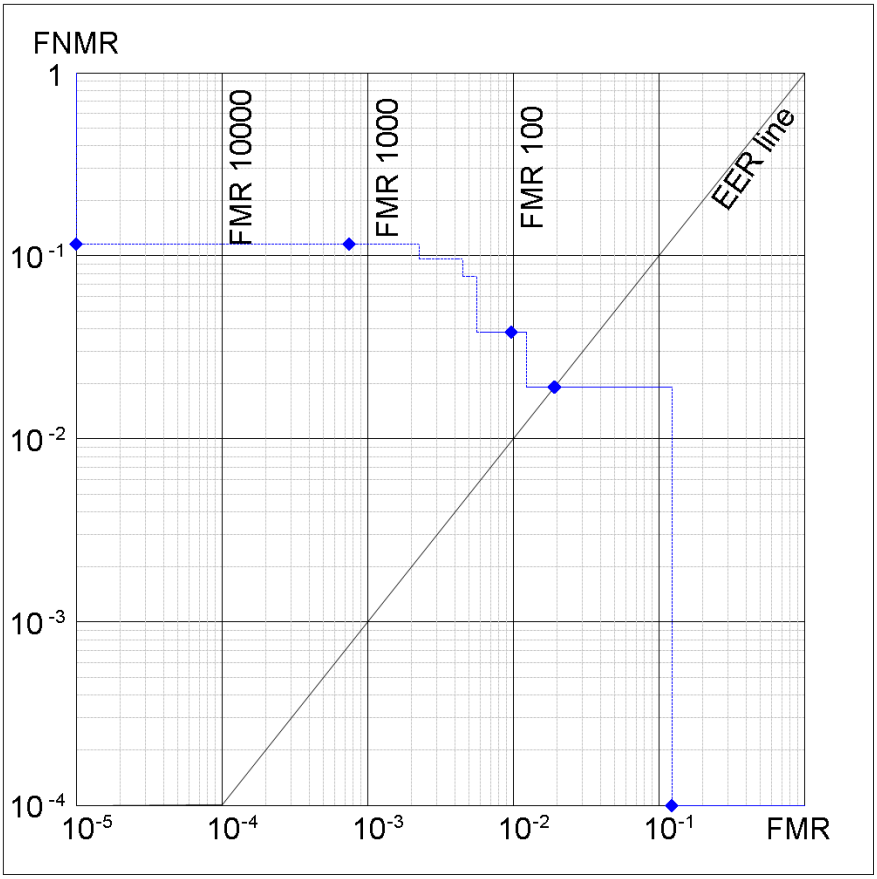

Figura 6.1 Grafico DET relativo alla sessione 1

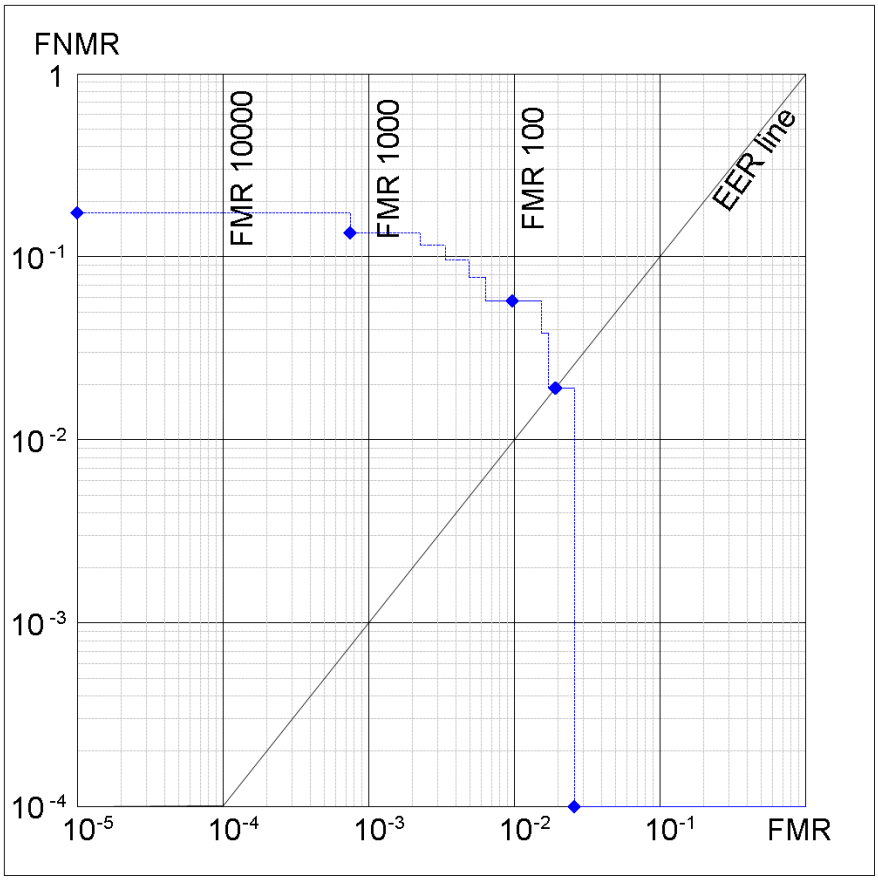

Figura 6.2 Grafico DET relativo alla sessione 2

Nelle figure 6.3 e 6.4 vengono riportati i grafici "*Match Error Rate*" che mostrano esplicitamente l'andamento di *FMR* e *FNMR* in funzione della soglia di sicurezza.

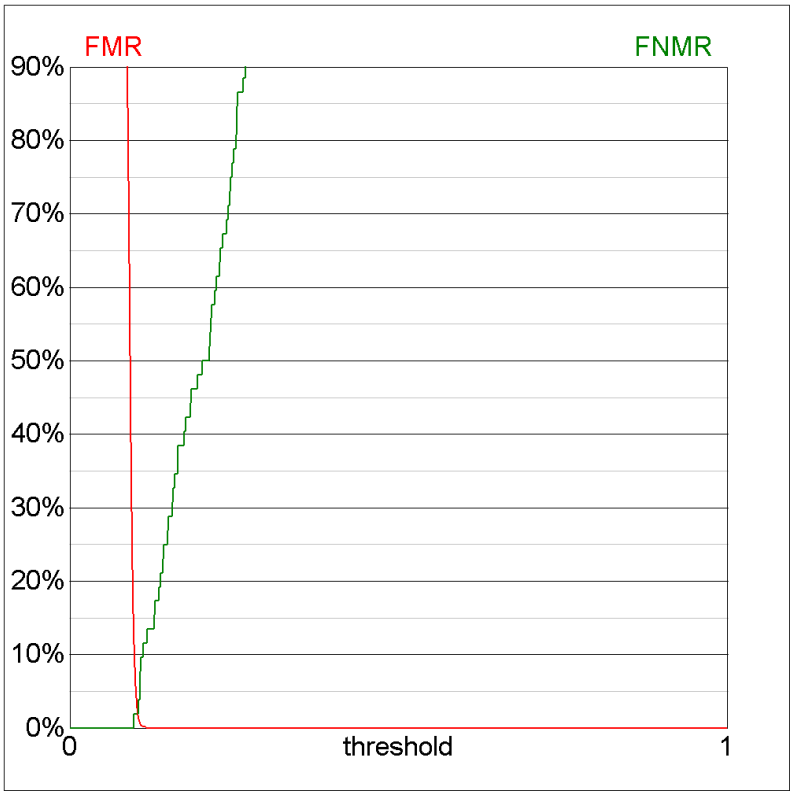

Figura 6.3 Grafico "Match Error Rate" relativo alla sessione 1

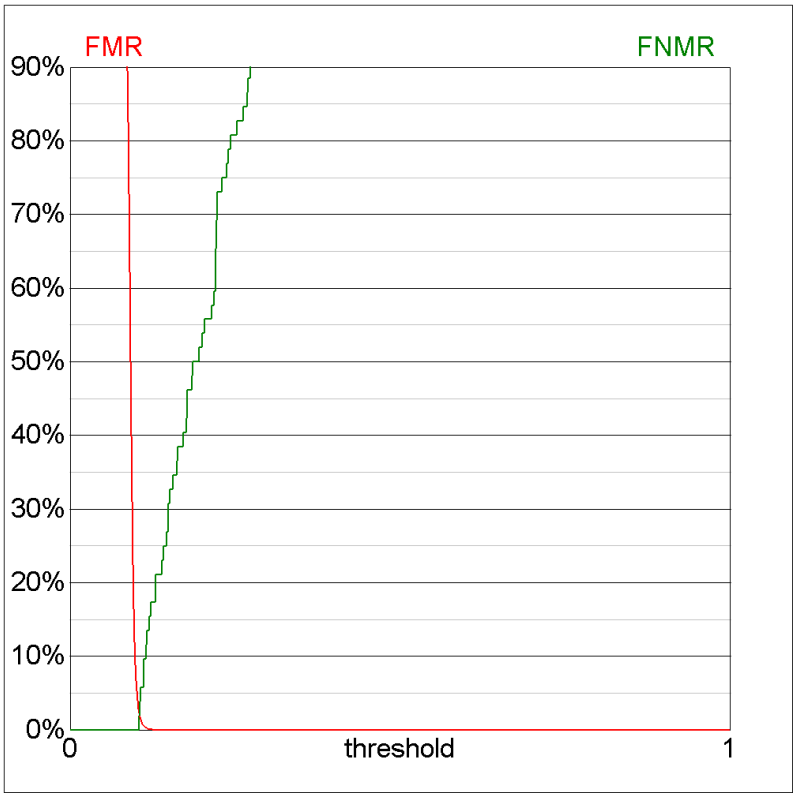

Figura 6.4 Grafico "Match Error Rate" relativo alla sessione 2

Capitolo 7.
Conclusioni

7.1 Risultati ottenuti

L'obiettivo di questo lavoro di tesi era quello di valutare la fattibilità di un sistema di acquisizione di impronte digitali mediante la fotocamera di uno Smartphone; si considerava un obiettivo ambizioso il raggiungimento di un errore di riconoscimento inferiore al 5%.

Sulla base dei risultati riportati nel capitolo 6, gli obiettivi sono stati raggiunti e superati, evidenziando non solo che un sistema che acquisisce le impronte digitali tramite la fotocamera di uno Smartphone è fattibile, ma che può anche avere buone prestazioni di riconoscimento biometrico.

I vari algoritmi sviluppati hanno consentito di portare l'accuratezza del sistema da un *EER* del 45% fino ad un *EER* di 1,92%.

Il sistema, tuttavia, è fortemente influenzato dalla qualità delle immagini: esse hanno un ruolo fondamentale sui risultati ottenuti. In particolare, con il metodo di acquisizione utilizzato, è molto facile ottenere immagini di medio/bassa qualità, a differenza di acquisizioni eseguite tramite scanner per impronte di alta qualità.

Infine si è notato che il sistema risulta avere una minore accuratezza anche su impronte digitali di bassa qualità, non per il tipo di acquisizione ma per le caratteristiche intrinseche del dito, ad esempio impronte nelle quali le *ridge line* non sono molto evidenti.

7.2 Sviluppi ed ampliamenti futuri

Si ritiene che il sistema sia promettente e che in futuro potrebbe essere ulteriormente sviluppato e ampliato nei seguenti modi:

- Eseguendo il porting dell'intero sistema su piattaforma mobile: questo non risulterebbe essere un compito oneroso in quanto Windows Phone utilizza il linguaggio C# (con il quale è stato sviluppato il sistema su PC). La difficoltà principale risiede nell'efficienza: gli Smartphone, infatti, sono dotati di processori meno potenti di un PC, e questo causerebbe problemi di latenza durante l'esecuzione degli algoritmi del sistema.
- Eseguendo il porting del sistema su altri sistemi operativi mobile, quali Android e iOS. Questo risulta essere un compito decisamente più arduo, poiché sarebbe necessario riscrivere nei diversi linguaggi tutti gli algoritmi implementati e le librerie utilizzate.
- Acquisire un nuovo database in ambiente outdoor, quindi con illuminazione e sfondo variabili. Questo potrebbe comportare maggiori problemi all'algoritmo presente sullo Smartphone, in particolare nella fase di messa a fuoco dell'immagine, che al momento è effettuata tramite un analisi del colore.

Bibliografia

Application Store.
http://en.wikipedia.org/wiki/Application_store

Brunetti, Roberto.
"Introduzione a Windows Phone 7.5"
http://www.html.it/pag/19131/introduzione-a-windows-phone-75/

Cappelli, R; Ferrara, M; Maio, D.
"A Fast and Accurate Palmprint Recognition System Based on Minutiae", Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on , vol.42, no.3, pp.956,962, June 2012

Cappelli, R; Ferrara, M; Maltoni, D.
"MCC: a baseline algorithm for fingerprint verification in FVC-onGoing", in proceedings 11th International Conference on Control, Automation, Robotics and Vision (ICARCV),
Singapore, December 2010

Cappelli, R; Ferrara, M; Maltoni, D.
"Minutia Cylinder-Code: a new representation and matching technique for fingerprint recognition",
IEEE Transactions on Pattern Analysis Machine Intelligence,
vol.32, no.12, pp.2128-2141, December 2010

Ferrara, M; Franco, A; Maltoni, D.
"Fingerprint scanner focusing estimation by Top Sharpening Index",
in proceedings 14th International Conference on Image Analysis and Processing (ICIAP07), Modena, Italy,
pp.223-228, September 2007

History of the Smartphone.
http://qrcodescanning.com/smartphonehist.html

Iacubino, Angelo.
"Creare applicazioni di successo per iPhone e iPad"
Edizioni Hoepli, 2010, pp. 312

Maltoni, D; Maio, D; Jain, A. K. e Prabhakar, S.
"Handbook of Fingerprint Recognition".
Springer: New York, NY, 2003.

Reed, B.
"A Brief History of Smartphones", 2010.
http://www.techhive.com/article/199243/a_brief_history_of_smartphones.html

Salvioli, Luca.
"Microsoft acquista i cellulari di Nokia per 7,17 miliardi di dollari",
Il Sole 24 Ore, 3 settembre 2013

Stark, Jonathan.
"Sviluppare applicazioni per iPhone",
Ed. Tecniche Nuove, 2010, pp. 161

Wu, Jin Chu.
"Statistical analysis of widths and heights of fingerprint images in terms of ages from segmentation data",
15 ottobre 2008