ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA
SCUOLA DI SCIENZE

Corso di Laurea in Ingegneria e Scienze Informatiche

ANALISI DEGLI ERRORI DI ESTRAZIONE DI MINUZIE IN IMPRONTE DI BASSA QUALITÀ

Relazione finale in: Fondamenti di Elaborazione di Immagini

Relatore:
Chiar.mo Prof.
RAFFAELE CAPPELLI
Co-relatore:
Dott. MATTEO FERRARA

Presentata da:
ALESSANDRO BAGNOLI

II SESSIONE
ANNO ACCADEMICO 2014–2015

PAROLE CHIAVE

Biometria

Impronte digitali

Estrazione minuzie

Miglioramento prestazioni

Ground truth

Indice

Introduzione

In un' epoca in cui la società è costantemente connessa alla rete come quella in cui stiamo vivendo, è sempre più rischioso affidarsi a mezzi di riconoscimento ormai anacronistici come password o tessere magnetiche. Diviene quindi indispensabile trovare metodi alternativi per verificare l'identità della persona che cerca di accedere a sistemi elettronici personali (come ad esempio smartphone o computer), ad account su internet o a luoghi fisici in cui l'accesso è riservato solo ad una stretta cerchia di individui autorizzati.

Una valida soluzione è stata trovata nell'utilizzo della biometria. Essa è la disciplina che studia le caratteristiche fisiologiche e comportamentali dell'uomo allo scopo di identificarne i più adatti parametri e metriche di confronto per poter riconoscere in maniera sicura l'identità delle persone. Alcune delle caratteristiche più promettenti risiedono nel DNA, nell'orecchio, nel volto, nella mano o in parti dell'occhio, tuttavia le caratteristiche più affidabili nella maggior parte delle applicazioni sono le impronte digitali, tuttora le più utilizzate nei sistemi automatici di riconoscimento di identità.

Una generica impronta digitale può essere acquisita tramite scanner o inchiostro e il suo riconoscimento avviene mediante caratteristiche intrinseche dell'impronta stessa, come sarà approfondito nel primo capitolo di questo documento di tesi.

La corretta individuazione di questa caratteristiche è una parte cruciale per il riconoscimento di un'impronta e per risparmiare tempo e forza lavoro sono stati sviluppati algoritmi in grado di automatizzare questo processo. Le prestazioni di questi algoritmi sono generalmente soddisfacenti, ma in alcuni casi particolari possono commettere sostanziali errori.

Lo scopo di questa tesi è quello di analizzare i casi in cui un particolare algoritmo di estrazione compie notevoli errori e di migliorarne le prestazioni. Per giungere a questo obiettivo si è seguito un metodo di lavoro ben definito basato su una serie di fasi.

Prima di descriverle, nel Capitolo 1 vengono fornite le definizioni e le nozioni base che ruotano attorno al mondo del riconoscimento delle impronte. Successivamente, nel Capitolo 2 viene descritta la creazione di un database di impronte su cui effettuare le analisi e vengono confrontate le prestazioni di due diversi algoritmi di estrazione. Il Capitolo 3 e il Capitolo 4 si occupano nello specifico degli approcci utilizzati e dei metodi sviluppati per migliorare i risultati di uno dei due algoritmi. Infine nel Capitolo 5 sono tratte alcune conclusioni sul lavoro svolto.

Capitolo 1

Il riconoscimento delle impronte

1.1 Cenni storici

Rappresentazioni di impronte digitali sono state trovate in un grande numero di reperti archeologici e artefatti [5]. Nonostante questi ritrovamenti dimostrino come già nell'antichità l'uomo fosse a conoscenza dell'unicità delle impronte digitali, questa consapevolezza non aveva alcuna base scientifica.

I primi studi scientifici approfonditi riguardo alle impronte si devono al botanico inglese Nehemian Grew che pubblicò nel 1864 il primo documento scientifico sulla struttura delle creste, delle valli e dei pori. Nel 1823 John Evangelist Purkinje presentò il primo schema di classificazione delle impronte, classificandole in nove diverse categorie basandosi sulla struttura generale delle creste. Nel 1880 Henry Fauld propose scientificamente per la prima volta l'unicità delle impronte basandosi su osservazioni empiriche, mentre nello stesso periodo William Herschel affermò di aver sperimentato nel campo del riconoscimento di impronte per circa 20 anni. Questi avvenimenti sancirono la nascita della moderna *fingerprint recognition*.

Una svolta in questo ambito si ebbe nel 1888, quando Sir Francis Galton introdusse il concetto di minuzie come base per la verifica dell'uguaglianza di due impronte. Qualche anno dopo, più precisamente nel 1899, Edward Henry definì un sistema di classificazione delle impronte in grado di consentire l'archiviazione e la comparazione.

Nei primi anni del ventesimo secolo l'impronta digitale viene formalmente riconosciuta come un valido metodo di identificazione e il suo riconoscimento

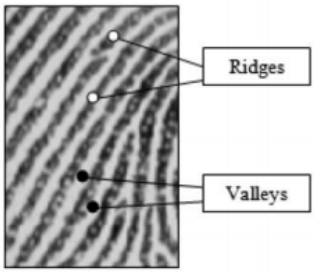
diventa un'operazione standard in campo forense. A partire dal 1960 l'FBI sviluppò il primo sistema automatico di identificazione delle impronte.

Le tecnologie di riconoscimento di impronte oggigiorno trovano utilizzo anche in campi diversi da quello forense, come in applicazioni commerciali e indirizzate a un vasto numero di utenti. L'aumentare della loro popolarità e diffusione ha permesso ai sistemi biometrici di riconoscimento di impronte di diventare quasi il sinonimo di sistema biometrico stesso.

1.2 Caratteristiche

La caratteristica strutturale più evidente di un'impronta è una trama di creste ("*ridge lines*") e valli ("*valleys*") che formano un disegno complesso chiamato *ridge pattern*, come si può vedere in Figura 1.1a. In un'immagine di una generica impronta digitale, le creste si possono riconoscere facilmente poichè più scure delle valli (Figura 1.1b). Lo spessore delle creste varia solitamente da un minimo di circa 100μm per quelle più sottili ad un massimo di 300μm per quelle più spesse. La maggior parte delle ferite al dito come bruciature, abrasioni o tagli non influenzano la struttura sottostante delle creste, e il *ridge pattern* originale rimane invariato anche quando le ferite si risanano [5].

(a) Il *ridge pattern*



(b) Creste e valli nel dettaglio

Figura 1.1: Struttura di un'impronta.

I dettagli delle creste sono organizzati con un ordine gerarchico a tre livelli di seguito descritti:

- **Livello 1** - Chiamato anche livello globale. Le creste avanzano perlopiù l'una parallela alle altre, ma in una o più regioni acquisiscono forme particolari, che possono essere caratterizzate da un alto angolo di curvatura, una maggiore densità di terminazioni di creste, etc. Queste regioni, chiamate singolarità o punti singolari, possono essere classificate in tre tipi: cicli ("*loop*"), delta e spirali ("*whorl*"), visibili in Figura 1.2. Queste tre tipologie di singolorità possono essere riconosciute dalla loro forma, rispettivamente simili a $\cap$, Δ e O. Talvolta le spirali non sono classificate come tipo di singolarità a se stante poichè possono essere descritte come due cicli uno in fronte all'altro. Si definisce inoltre il punto di *core* come il punto più a nord della *ridge line* più interna. Il punto di *core* corrisponde quindi al centro del ciclo più a nord. Per impronte che non contengono cicli o spirali diventa difficile individuare il *core*, in questi casi viene definito come il punto di massima curvatura delle creste.

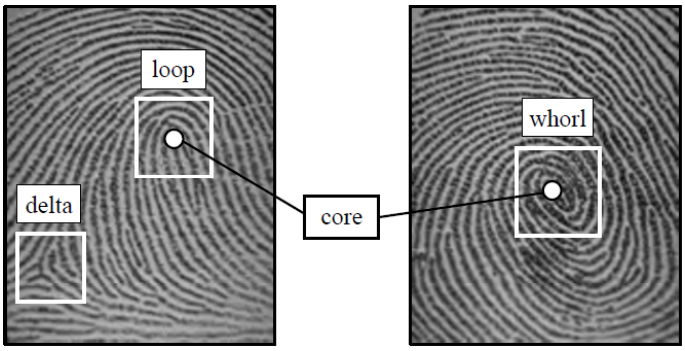
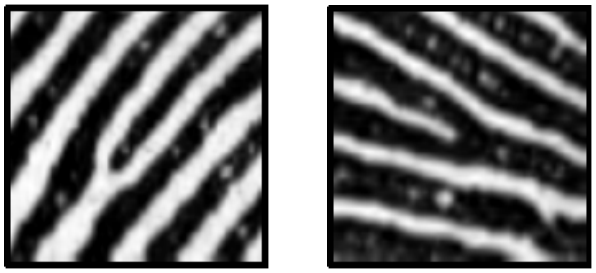


Figura 1.2: Singolarità: cicli, delta e spirali.

- **Livello 2** - Chiamato anche livello locale. In questo livello possono essere trovate altre importanti caratteristiche, chiamate minuzie ("*minutiae*"). La parola minuzia letteralmente significa piccolo particolare, ma nel contesto delle impronte si riferisce a discontinuità nella struttura creste/valli. Sir Francis Galton fu il primo a classificare i vari tipi di minuzia e verificò che esse non cambiano mai per tutta la vita di un individuo. Sono stati individuate sette principali categorie di minuzie ma per motivi di semplicità si utilizza il modello proposto dall'FBI, che attualmente considera

solo due categorie principali di minuzie: le terminazioni ("*endings*") e le biforcazioni ("*bifurcations*") visibili in Figura 1.3. Si parla di terminazione della cresta quando questa si interrompe bruscamente, mentre si ha una biforcazione quando essa diverge in due rami figli.



(a) Terminazione (b) Biforcazione

Figura 1.3: Le due principali categorie di minuzie.

- **Livello 3** - A questo livello possono essere estratte ulteriori caratteristiche che includono tutti i dettagli dimensionali delle creste come la larghezza e la forma, ma le *feature* più importanti sono soprattutto i pori (Figura 1.4). Ogni cresta è infatti caratterizzata da pori che si estendono per tutta la lunghezza della cresta. Sono presenti dai 9 ai 18 pori per ogni centimetro di cresta e si sostiene che un numero di pori compreso tra 20 e 40 sia sufficiente per determinare l'identità di un individuo. Nonostante le caratteristiche presenti a questo livello siano altamente peculiari e importanti, al momento non sono largamente utilizzate nella *fingerprint recognition* poichè per essere individuate richiedono scanner di impronte ad alta risoluzione (1,000 dpi).

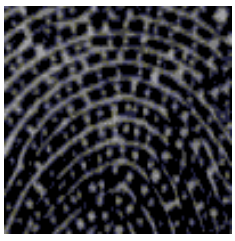


Figura 1.4: Pori. Individuabili dai cerchietti bianchi all'interno delle creste.

1.3 Estrazione delle minuzie

Come è stato detto precedentemente le caratteristiche che possono essere trovate a livello locale, chiamate minuzie, non cambiano mai durante la vita di una persona, e sono quindi le migliori candidate ad essere usate come tratti altamente distintivi che permettono di distinguere un'impronta da un'altra. Il processo di estrazione delle minuzie prevede precisi passi.

1.3.1 Orientazione delle creste

Il primo passo include l'individuazione e l'estrazione delle giuste orientazioni locali delle *ridge line*. L'orientazione locale delle *ridge line* in posizione [*x*,*y*] è definita come l'angolo $\theta_{x,y}$ che le creste, le quali attraversano un intorno piccolo a piacere del punto [*x*,*y*], formano con l'asse orizzontale. La maggior parte dei metodi di estrazione tuttavia effettua una stima in posizioni discrete riducendo i costi computazionali e permettendo di ottenere valutazioni sugli altri pixel tramite interpolazione.

L'immagine delle orientazioni di un'impronta, chiamata anche immagine direzionale, è una matrice D in cui ogni elemento $\theta_{i,j}$ corrispondente al nodo [*i*,*j*] di una griglia quadrata posizionata sul pixel $[x_i, y_j]$, denota l'orientazione media delle creste in un intorno di $[x_i, y_j]$. L'approccio più semplice e naturale per l'estrazione delle orientazioni locali è basato sul calcolo del gradiente dell'immagine. Il gradiente $\nabla(x, y)$ nel punto [*x*,*y*] dell'immagine I è un vettore bidimensionale $[\nabla_x(x, y), \nabla_y(x, y)]$ dove ∇_x e ∇_y sono rispettivamente le derivate di I in [*x*,*y*] rispetto alle direzioni x e y. L'angolo di fase del gradiente indica la direzione del massimo cambiamente d'intensità dei pixel e la direzione θ di un ipotetico edge che attraversa la regione centrata in [*x*,*y*] è ortogonale all'angolo di fase del gradiente in [*x*,*y*]. Questo metodo, nonostante la semplicità e l'efficienza, ha qualche lato negativo:

- Non linearità e discontinuità attorno a 90°.
- La stima di una singola orientazione rappresenta un'analisi di basso livello eccessivamente sensibile al rumore. D'altra parte non è possibile fare una semplice media di più gradienti a causa della circolarità degli angoli.
- Il concetto di orientazione media non è sempre ben definito; ad esempio qual'è la media di due orientazioni ortogonali 0° e 90°? 45° o 135°?

Una soluzione semplice ed elegante per eliminare il problema della circolarità consiste nel raddoppiare gli angoli; in questo modo ogni elemento dell'immagine direzionale D è codificato dal vettore $d = [r \cdot cos(2\theta), r \cdot sin(2\theta)]$ dove r denota la coerenza dell'orientazione θ. La media degli angoli in una finestra locale n x n può essere calcolata effettuando separatamente la media per le due componenti x e y.

1.3.2 Enhancement

Le prestazioni degli algoritmi di estrazione delle caratteristiche e riconoscimento di impronte dipendono molto dalla qualità dell'immagine. Una parte non trascurabile delle immagini acquisite (circa il 10% [5]) è di scarsa qualità per varie cause che saranno osservate nel dettaglio nel Capitolo 1.5. Esempi di impronte di diverse qualità possono essere osservati in Figura 1.5. Esistono diversi tipi di degradazioni associate a immagini di impronte:

- Le creste non sono continue.
- Le creste parallele non sono ben separate a causa della presenza di rumore presente nell'immagine.
- Tagli, ferite e abrasioni.

(a) Buona (b) Media (c) Cattiva

Figura 1.5: Impronte di diverse qualità.

Questi problemi rendono l'individuazione delle creste, e quindi delle *feature*, ardua. Il miglioramento della qualità delle immagini di impronte è l'obiettivo principale delle tecniche di enhancement.

L'input per un algoritmo di enhancement è solitamente un'immagine a livelli di grigio mentre l'output può essere un'immagine a livelli di grigio o un'immagine binaria. Le tecniche di miglioramento della qualità delle immagini generiche non producono risultati molto soddisfacenti con immagini di impronte; tuttavia alcune operazioni come l'aumento del contrasto, la modifica dell'istogramma o la normalizzazione possono risultare utili per una prima fase di pre-processing. La tecnica più diffusa per l'enhancement di immagini di impronte è basata sul filtraggio contestuale, dove le caratteristiche del filtro utilizzato cambiano a seconda del contesto locale: viene precalcolato un insieme di filtri e in ogni regione dell'immagine si seleziona quello più idoneo. Il filtro più utilizzato per queste operazioni è quello di Gabor, i cui parametri e caratteristiche saranno analizzati dettagliatamente nel Capitolo 3.

1.3.3 Binarizzazione, thinning e localizzazione delle minuzie

Dopo aver applicato uno degli algoritmi di enhancement, si passa alla fase finale che prevede una binarizzazione dell'immagine ottenuta come output dell'enhancement, un thinning dell'immagine binaria, e infine si passa alla localizzazione vera e propria delle minuzie.

La binarizzazione può essere ottenuta localizzando i picchi nei profili dei livelli di grigio in una sezione ortogonale all'orientazione delle creste. Attorno a ogni pixel $[x,y]$ si posiziona una finestra orientata di dimensione 16 x 16 e il profilo dei livelli di grigio si ottiene proiettando l'intensità dei pixel nella sezione centrale. Il profilo viene poi smussato tramite un'operazione di media locale; i picchi e i due pixel a essi adiacenti da entrambi i lati sono associati al foreground nell'immagine binarizzata.

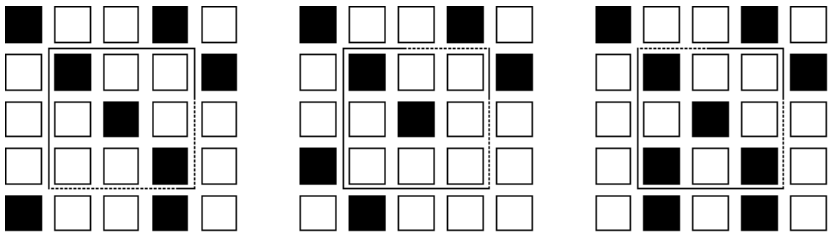
Nel processo di thinning l'immagine binaria è sottoposta a un passo di assottigliamento che riduce lo spessore delle creste a 1 pixel. L'algoritmo di thinning più utilizzato è quello di Hilditch.

Per la localizzazione delle minuzie sullo scheletro ottenuto dallo thinning si fa ricorso a una scansione dell'immagine per localizzare i pixel corrispondenti alle minuzie e si basa sull'analisi del *crossing number*. Il crossing number $cn(\mathbf{p})$ di un pixel $\mathbf{p}$ in un'immagine binaria è definito come la sommatoria delle differenze tra coppie di pixel adiacenti nell'8-intorno di $\mathbf{p}$, diviso 2:

$$cn(\mathbf{p}) = \frac{1}{2} \sum_{i=1...8} |val(\mathbf{p}_{i \bmod 8}) - val(\mathbf{p}_{i-1})|$$

dove $\mathbf{p}_0$, $\mathbf{p}_1$, ..., $\mathbf{p}_7$ sono i pixel appartenenti a una sequenza ordinata di pixel che costituiscono l'intorno del pixel $\mathbf{p}$ e $val(\mathbf{p}) \in \{0,1\}$ è il valore del pixel $\mathbf{p}$. Come si evince dalla Figura 1.6, un pixel $\mathbf{p}$ con $val(\mathbf{p}) = 1$:

- È un punto interno a una cresta se $cn(\mathbf{p}) = 2$.
- Corrisponde a una terminazione se $cn(\mathbf{p}) = 1$.
- Corrisponde a una biforcazione se $cn(\mathbf{p}) = 3$.
- Appartiene a una minuzia più complessa se $cn(\mathbf{p}) > 3$.



(a) $cn(\mathbf{p}) = 2$ (b) $cn(\mathbf{p}) = 1$ (c) $cn(\mathbf{p}) = 3$

Figura 1.6: Individuazione di minuzie grazie al crossing number

1.4 Il confronto

Un algoritmo di *matching* confronta due impronte e restituisce un numero che rappresenta il grado di similarità, compreso tra 0 e 1. Confrontare immagini di impronte non è un compito banale, poichè uno stesso dito può essere impresso sul sensore in vari modi. I possibili fattori responsabili di questa variabilità sono molteplici: dito posizionato o ruotato in maniera diversa, pressione del dito sul sensore e condizione della pelle, rumore causato da residui di sporcizia sul sensore, etc.

I metodi per il confronto possono essere classificati in tre diverse famiglie.

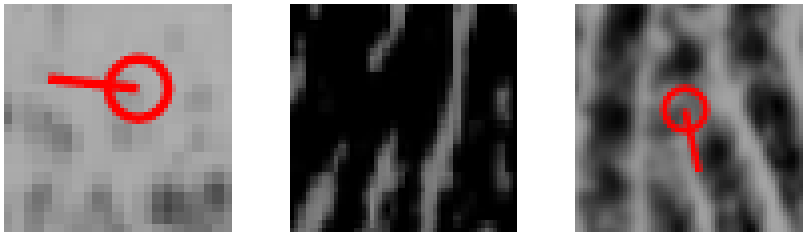
- Confronto basato su correlazione: le due immagini vengono sovrapposte e viene calcolata la correlazione tra i pixel corrispondenti. Poichè il posizionamento e la rotazione del dito sono sconosciuti, è necessario applicare la correlazione per tutte le possibili sovrapposizioni. Il difetto principale di questo metodo è il suo costo computazionale oltre a una bassa tolleranza alla distorsione lineare della pelle.

- Confronto basato su minuzie: la tecnica più diffusa e famosa, ormai diventata standard per il confronto di impronte. Le minuzie vengono estratte dalle due impronte e salvate come set di punti sul piano bidimensionale. Questo metodo prova ad allineare le minuzie dell'immagine fornita in input e dell'immagine di riferimento, e a trovare il numero di minuzie correttamente abbinate. Dopo l'allineamento, due minuzie sono considerate corrispondenti se la distanza spaziale e la direzione differiscono tra di loro in quantità inferiore di una tolleranza definita in precedenza. [3]

- Confronto basato su analisi del ridge pattern: l'estrazione di minuzie in impronte di bassa qualità è un compito problematico. Le altre *feature* del *ridge pattern* come le orientazioni locali o la forma delle creste, possono essere estratte in maniera più affidabile, ma la loro persistenza e la loro unicità non sono ai livelli delle minuzie. In teoria, il confronto basato su correlazione puà essere pensato come una sottocategoria del confronto basato su feature diverse da minuzie, in quanto le intensità dei pixel sono esse stesse feature del pattern del dito.

1.5 Problemi nell'estrazione di minuzie

Come si è detto nel corso del Capitolo 1.3, è più che plausibile che un sensore acquisisca immagini di impronte di bassa qualità. In tali impronte, la localizzazione ed estrazione delle minuzie risulta difficoltosa poichè una cattiva qualità può causare errori in una o più fasi del processo di estrazione, portando a risultati finali non coerenti con la realtà. Gli errori che possono essere generati in impronte di bassa qualità sono generalmente minuzie false, ovvero minuzie segnalate in posizioni in cui realmente non ce ne sono, minuzie non trovate (*missed*), oppure minuzie segnalate del tipo sbagliato (biforcazioni al posto di terminazioni o viceversa). Esempi di questi errori possono essere osservati in Figura 1.7. Gli errori più gravi che incidono sulle prestazioni di un algoritmo di estrazione e riconoscimento sono i primi due descritti.



(a) Minuzia falsa

(b) Minuzia non trovata

(c) Biforcazione segnalata al posto di una terminazione

Figura 1.7: Errori di estrazione di minuzie.

La classificazione della qualità dell'immagine di un'impronta viene effettuata tramite un indice definito dal NIST (*National Institute of Standards and Technlogy*) chiamato NFIQ. Questo indice divide le impronte in cinque categorie di qualità.

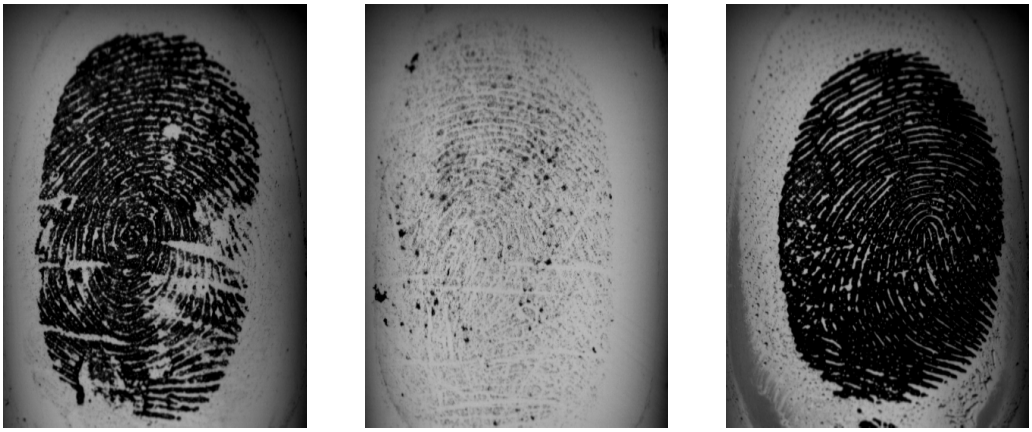
- NFIQ 1 - Qualità eccellente.
- NFIQ 2 - Qualità molto buona.
- NFIQ 3 - Qualità buona.
- NFIQ 4 - Qualità appena sufficiente.
- NFIQ 5 - Qualità scadente.

Le impronte più problematiche appartengono alle ultime due categorie: NFIQ 4 e NFIQ 5.

I fattori che determinano l'acquisizione di un'immagine di bassa qualità sono molteplici:

- Condizioni della pelle (umida, secca, o con tagli).
- Pressione non ideale del dito sul sensore.
- Bassa qualità intrinseca delle impronte di particolari categorie di persone (anziani, lavoratori manuali).

In Figura 1.8 vengono mostrati alcuni esempi di impronte di bassa qualità.



(a) Impronta con abrasioni

(b) Impronta secca

(c) Impronta umida o troppa pressione esercitata sul sensore

Figura 1.8: Impronte di bassa qualità.

Capitolo 2

Database con Ground Truth delle minuzie

In questo capitolo verranno illustrate tutte le fasi che hanno portato alla creazione di un database di immagini di impronte di bassa qualità e alla sua successiva analisi. Con l'espressione Ground Truth si vuole indicare la verità assoluta di qualcosa. Un database contenente il Ground Truth delle minuzie equivarrà quindi ad un archivio di immagini di impronte contenenti le minuzie estratte in maniera ideale, senza errori. Tale database è ottenibile soltanto etichettando manualmente le minuzie impronta per impronta.

2.1 Selezione delle impronte

Prima di partire con l'etichettatura manuale delle minuzie e quindi con la creazione del Ground Truth, si è reso necessario svolgere alcune operazioni preliminari. I database di partenza infatti includevano immagini di impronte di tutte le qualità, da NFIQ 1 a NFIQ 5, mentre quelle interessanti da prendere in analisi sono solo quelle con NFIQ 4 e NFIQ 5. Ogni immagine ha associato un file in formato standard, ISO minutiae template, con estensione .ist che contiene in forma binaria i dati sulle minuzie estratte dall'immagine a cui fanno riferimento.

Tramite l'utilizzo di una console application che automatizzasse il processo, si sono divise le immagini secondo il loro indice NFIQ. La parte centrale di questa applicazione viene illustrata nel Listato 2.1. Dato il percorso della cartella contenente le immagini, e data la loro estensione, l'applicativo esa-

mina ciascuna immagine e, tramite il metodo statico `Calculate` della classe `NistFingerprintImageQuality`, viene stabilito per ogni immagine il relativo indice di qualità. Infine avviene la separazione vera e propria delle immagini, creando una cartella per ciascun indice NFIQ. Ogni cartella conterrà i file immagine e i relativi file .ist.

```
...
var imageFileList = Directory.GetFiles(dbFolder, searchPattern);
var nFiqs = new List<string>[numNifqClasses];
for (int i = 1; i < nFiqs.Length; i++)
{
   nFiqs[i] = new List<string>(imageFileList.Length);
   var nfiqFolderPath=Path.Combine(outputFolder, "Nfiq_" + i.ToString());
   if (!Directory.Exists(nfiqFolderPath))
   {
      Directory.CreateDirectory(nfiqFolderPath);
   }
}
foreach (var file in imageFileList)
{
   var img = ImageBase.LoadFromFile(file).ToByteImage();
   int nfiq;
   float confidence;
   if (NistFingerprintImageQuality.Calculate(img, 500, out nfiq, out
       confidence))
   {
      var imageName = useFullPath ? file :
          Path.GetFileNameWithoutExtension(file);
      nFiqs[nfiq].Add(imageName);
      if (Directory.Exists(ISOTemplateFolder))
      {
         string fileISO = Path.ChangeExtension(Path.GetFileName(file),
             "ist");
         File.Copy(Path.Combine(ISOTemplateFolder, fileISO) ,
             Path.Combine(outputFolder, "Nfiq_" + nfiq.ToString(),
             fileISO), true);
      }
      File.Copy(file, Path.Combine(outputFolder, "Nfiq_" + nfiq.ToString(),
          Path.GetFileName(file)), true);
   }
}
...
```

Listato 2.1: Divisione delle immagini in base al loro indice NFIQ.

2.2 Etichettatura manuale delle minuzie

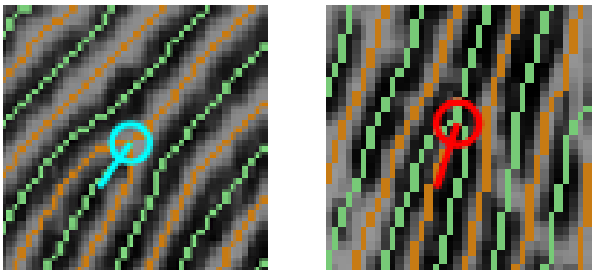
Una volta ottenute le immagini delle impronte e i relativi ISO template divisi in base al loro indice di qualità è iniziata la fase di etichettatura manuale. L'etichettatura delle minuzie prevede precise regole che vanno seguite per avere un risultato finale soddisfacente.

2.2.1 Regole generali per una corretta etichettatura

Il sistema di coordinate utilizzato per segnalare le minuzie di un'impronta è quello Cartesiano. I punti corrispondenti alle minuzie sono rappresentati dalle loro coordinate x e y. L'origine del sistema di riferimento è localizzato nell'angolo in alto a sinistra dell'immagine originale con la coordinata x che cresce da sinistra verso destra e la coordinata y che cresce dall'alto verso il basso. Le coordinate delle minuzie vengono espresse in pixel.

La minuzia corrispondente a una terminazione è definita come il punto di biforcazione della valle posta di fronte alla terminazione di una cresta. Se l'area di una valle è stata assottigliata ad uno scheletro di un solo pixel di larghezza, il punto in cui i tre rami si intersecano è la coordinata della minuzia.

Viceversa, la minuzia corrispondente a una biforcazione è definita come il punto di biforcazione dello scheletro di una cresta. Se la cresta è stata assottigliata ad uno scheletro di un solo pixel di larghezza, il punto in cui i tre rami si intersecano è la coordinata della minuzia. La precisa localizzazione delle minuzie che soddisfa questi requisiti può essere osservata in Figura 2.1.



(a) Terminazione (b) Biforcazione

Figura 2.1: La corretta localizzazione delle minuzie. Le linee verdi rappresentano lo scheletro delle creste, mentre le linee arancioni quello delle valli.

Ogni minuzia ha inoltre una direzione, individuabile dall'angolo che essa forma con l'asse cartesiano orizzontale. Gli angoli vengono espressi in forma standard, con il valore zero a destra e crescente in senso antiorario.

Nel caso di una terminazione, l'angolo da prendere in considerazione è quello formato dall'asse orizzontale e dal segmento che ha origine nella coordinata della terminazione e che si estende passando per lo scheletro della cresta.

Per quel che riguarda le biforcazioni invece, l'angolo da considerare è quello compreso tra l'asse orizzontale e il segmento che ha origine nella coordinata della biforcazione e che si estende passando per l'area compresa tra le due creste figlie [4]. Una rappresentazione grafica degli angoli e delle direzioni delle minuzie può essere osservata in Figura 2.2.

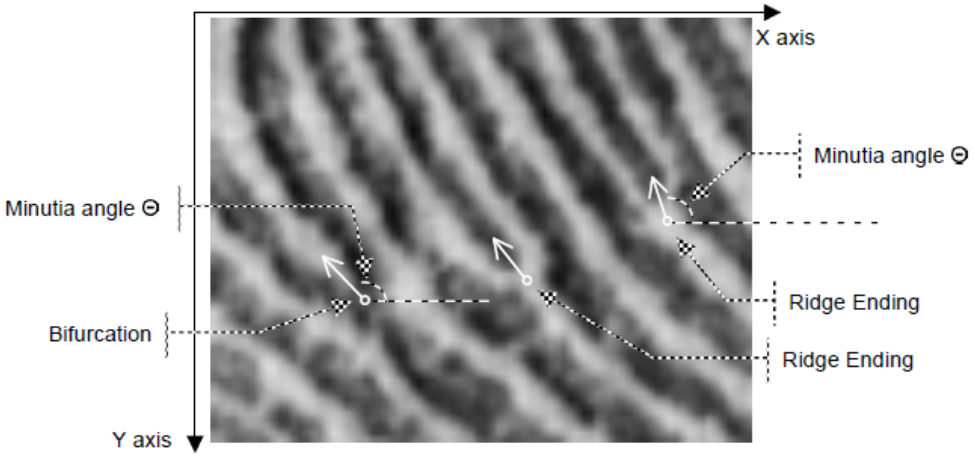


Figura 2.2: Angoli e direzioni per i due tipi di minuzia.

2.2.2 Uno strumento per l'etichettatura manuale delle minuzie: *Minutiae Template Modifier*

I file .ist associati alle immagini prese in analisi contengono i dati sulle minuzie estratte attraverso un algoritmo (che da ora in poi sarà chiamato Algoritmo B) partecipante ad un'edizione del *Fingerprint Verification Competition* (FVC), una delle competizioni di riconoscimento di impronte organizzate dal *Biometric System Laboratory* di Cesena (BioLab) alcuni anni fa. Esso è uno dei migliori algoritmi di estrazione minuzie allo stato dell'arte e spicca per prestazioni

migliori rispetto a quello sviluppato internamente al BioLab, che sarà chiamato Algoritmo A. Il processo di etichettatura non parte quindi da zero, ma si basa sulla correzione manuale delle minuzie estratte da Algoritmo B.

Lo strumento con cui è stato possibile portare a termine il lavoro di etichettatura prende il nome di *Minutiae Template Modifier*, un applicativo sviluppato internamente al BioLab. L'applicazione permette di modificare gli ISO template con l'aggiunta, la rimozione o la modifica delle posizioni delle minuzie. Come si può osservare dalla Figura 2.3, che mostra la schermata principale dell'applicativo, i colori utilizzati per etichettare le minuzie sono tre:

- Azzurro per le terminazioni.
- Rosso per le biforcazioni.
- Verde per minuzie non distinguibili chiaramente tra terminazione o biforcazione.

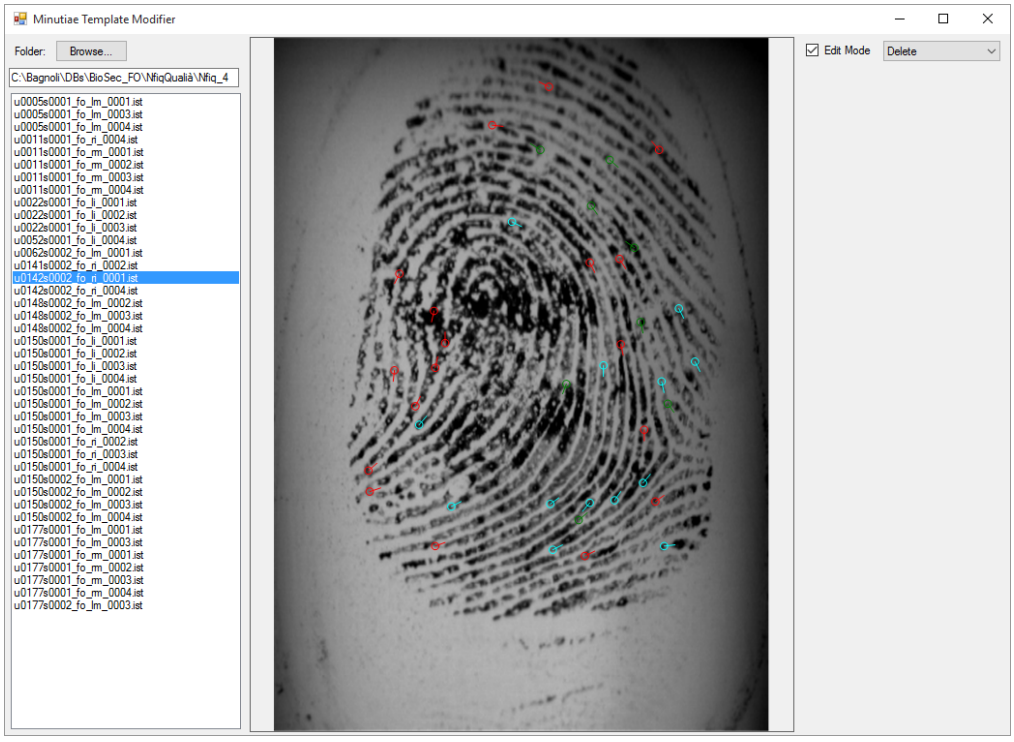
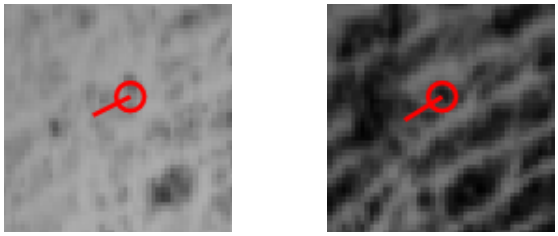


Figura 2.3: *Minutiae Template Modifier.*

Alcune delle minuzie etichettate si sono rivelate essere particolarmente complesse da individuare, e spesso sono risultati fondamentali confronti diretti con immagini della stessa impronta per localizzarle correttamente. Un esempio si trova in Figura 2.4: la minuzia non trovata etichettata in 2.4a è stata individuata solo dopo aver analizzato un'altra immagine della stessa impronta in cui le creste erano meglio definite (2.4b).



(a) (b)

Figura 2.4: Minuzia non trovata difficile da etichettare.

2.3 Il database realizzato

L'etichettatura manuale è stata realizzata su cento immagini di impronte di un database biometrico. I file immagine e ISO template erano salvati secondo il seguente pattern: utente_sessione_sensore_dito_acquisizione. Un esempio può essere u0001_s0002_fo_li_0003, dove u0001 rappresenta l'utente 1, s0002 la sessione 2, fo è il nome abbreviato del sensore utilizzato, li è l'acronimo per il dito scannerizzato in questo caso left index, e 0003 è il numero di acquisizione per quel dito, in quella sessione, di quello specifico utente.

L'ultimo step prima della fase di analisi vera e propria ha riguardato l'estrazione delle minuzie dalle stesse immagini etichettate manualmente con l'Algoritmo A. Nel Listato 2.2 è riportato il codice C# responsabile dell'estrazione tramite questo algoritmo.

```
1 class Extractor
2 {
3    private const string inputDBFolder = @"C:\Bagnoli\DBs\NFIQ_4_5\Images";
4    private const string outputFolder =
         @"C:\Bagnoli\DBs\NFIQ_4_5\ISOTemplates\BioLab";
5    private const string inputSearchPattern = "*.bmp";
6    private const bool parallelExecution = true;
```

```
7     private const bool skipExistingTemplates = false;
8
9     static void Main(string[] args)
10    {
11       var files = Directory.GetFiles(inputDBFolder, inputSearchPattern);
12       using (var progress = new SmartConsoleProgress(files.Length, 5000))
13       {
14          var partition = Partitioner.Create(0, files.Length);
15          var parallelOptions = new ParallelOptions()
16          {
17             MaxDegreeOfParallelism = (parallelExecution) ? -1 : 1,
18          };
19          Parallel.ForEach(partition, parallelOptions, p =>
20          {
21             var defaultExtractor = new MinutiaeExtraction();
22
23             for (int i = p.Item1; i < p.Item2; i++)
24             {
25                try
26                {
27                   var templateFilePath = Path.Combine(outputFolder,
                          Path.ChangeExtension(Path.GetFileName(files[i]),
                          ".ist"));
28
29                   if (!skipExistingTemplates ||
                          !File.Exists(templateFilePath))
30                   {
31                      var image =
                             ImageBase.LoadFromFile(files[i]).ToByteImage();
32
33                      var fingerprintTemplate =
                             defaultExtractor.Enroll(image);
34                      var minutiae = fingerprintTemplate.Minutiae;
35
36                      minutiae.SaveToIsoFile(templateFilePath);
37                   }
38                }
39                catch (Exception ex)
40                {
41                   Console.WriteLine(Path.GetFileName(files[i]));
42                   Console.WriteLine(ex.Message);
43                }
44                progress.StepIt();
45             }
46          }
47          );
48       }
49    }
50 }
```

Listato 2.2: Estrazione delle minuzie tramite Algoritmo A.

Compiuto questo ultimo passaggio, il risultato finale sono tre cartelle, ognuna contenente rispettivamente gli ISO Template per Ground Truth, Algoritmo A, e Algoritmo B.

2.4 Analisi di algoritmi di estrazione minuzie

La fase di analisi è stata facilitata dall'utilizzo di un altro applicativo, *ISO Minutiae Template Quality Analyzer*, che confronta i template del Ground Truth con quelli generati da Algoritmo A e Algoritmo B. I dati mostrati dal tool comprendono il numero di minuzie false e quello delle missed generato da uno dei due algoritmi confrontandolo direttamente col Ground Truth, insieme ad altre statistiche. La schermata principale del programma può essere osservata in Figura 2.5.

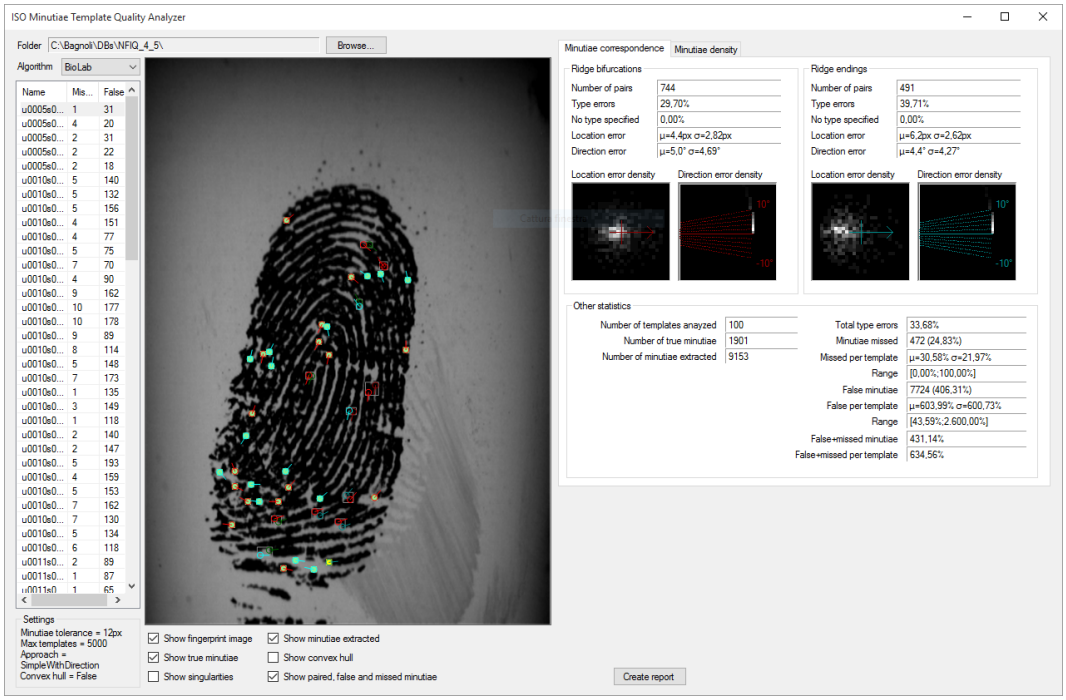


Figura 2.5: *ISO Minutiae Template Quality Analyzer.*

Nel pannello di sinistra si può osservare per ciascun template il numero di minuzie missed e false, che vengono evidenziate anche graficamente sull'im-

magine dell'impronta, insieme a quelle minuzie considerate accoppiate, ovvero presenti sia nel template del Ground Truth che in quello dell'estrattore preso in esame. Nel pannello inferiore a destra sono riportati i dati più importanti sugli errori di estrazione: il numero totale di minuzie false e missed.

Il programma ha confermato le prestazioni migliori di Algoritmo B con un numero di errori totali assai inferiore rispetto all'estrattore Algoritmo A, come si può vedere dalla Tabella 2.1.

	Algoritmo B	Algoritmo A
Missed	346	451
False	3042	9125

Tabella 2.1: Errori di estrazione dei due algoritmi a confronto.

2.4.1 Prestazioni sul confronto di impronte dei due algoritmi

Un algoritmo di confronto è definito come un algoritmo in grado di prendere una decisione sulla natura di un confronto tra due impronte, che può risultare autentico (*genuine*) o falso (*impostor*). Esso è caratterizzato da due fasi. Nella prima fase viene assegnato un punteggio di similarità. Questo punteggio assume un valore compreso tra 0 e 1, più è alto, più sono simili le due impronte. Nella seconda fase viene stabilito se il confronto è genuine o impostor usando una soglia. Se il punteggio ottenuto nella prima fase è più alto della soglia, il confronto è classificato come genuine, altrimenti come impostor.
Il confronto può compiere due tipi di errori:

- False Match - Quando l'algoritmo classifica come genuine un confronto di tipo impostor.
- False Non Match - Quando l'algoritmo classifica come impostor un confronto di tipo genuine.

Sulla base di questi errori, si possono calcolare dati utili per valutare le prestazioni del confronto.

- False Match Rate (FMR) - La probabilità che un algoritmo di riconoscimento classifichi come genuine un confronto che in realtà è di tipo

impostor. Viene calcolato come quoziente tra il numero di confronti impostor con punteggio di similarità più alto della soglia stabilita e il numero totale di confronti impostor.

- False Non Match Rate (FNMR) - La probabilità che un algoritmo di riconoscimento classifichi come impostor un confronto che in realtà è di tipo genuine. Viene calcolato come quoziente tra il numero di confronti genuine con punteggio di similarità più basso della soglia stabilita e il numero totale di confronti genuine.

- Equal Error Rate (EER) - Il punto in cui FMR e FNMR assumono lo stesso valore.

Avere un discreto numero di impronte delle stesse dita nel database creato ha reso sensata la creazione di una console application in grado di calcolare l'EER e l'FMR100 (rappresenta qual è l'FNMR ottenuto quando l'FMR è fissato all'1%) partendo da due file di testo, il primo contenente tutti i possibili confronti di tipo genuine, il secondo tutti quelli di tipo impostor. I risultati vengono riportati nella Tabella 2.2. Come si può osservare, la migliore estrazione delle minuzie da parte di Algoritmo B si ripercuote su dei valori minori di EER e FMR100 rispetto a Algoritmo A. Come sarà trattato nel Capitolo 2.5, alcuni casi limite in cui le impronte si presentavano con una qualità pessima non hanno permesso di ottenere un EER e un FMR100 pari a 0 nemmeno per il Ground Truth.

	Algoritmo B	Algoritmo A	GroundTruth
EER	10.58%	10.94%	0.94%
FMR100	20.28%	27.36%	0.94%

Tabella 2.2: Prestazioni sul confronto utilizzando le minuzie estratte coi due algoritmi e del Ground Truth.

2.5 Analisi delle impronte più problematiche

Utilizzando la stessa console application in grado di calcolare l'FMR100 e l'EER citata nel Capitolo 2.4, è stato possibile ottenere anche le coppie di impronte che contribuiscono agli errori a FMR100. Definiremo le impronte presenti in questo insieme come quelle più problematiche. Sul totale delle 100

pronte prese in analisi, le più problematiche per Algoritmo A sono riportate in Tabella 2.3, mentre quelle più problematiche per Algoritmo B sono riportate in Tabella 2.4.

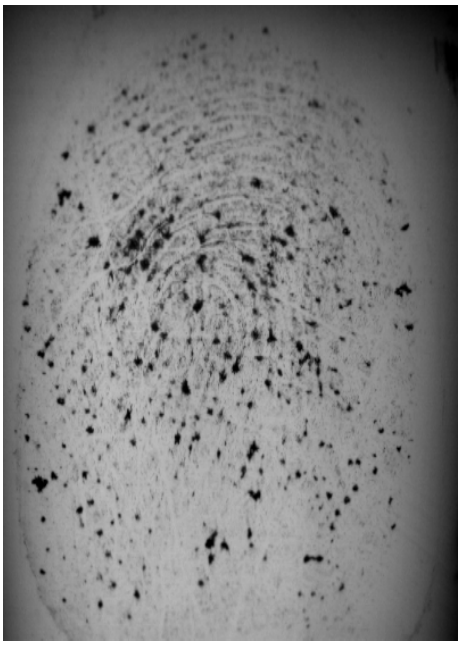
u0010s0001_fo_lm_0001	u0010s0001_fo_lm_0002	u0010s0001_fo_lm_0003	u0010s0001_fo_lm_0004
u0010s0002_fo_lm_0001	u0010s0002_fo_lm_0002	u0010s0002_fo_lm_0003	u0010s0002_fo_lm_0004
u0010s0001_fo_li_0001	u0010s0001_fo_li_0002	u0010s0001_fo_li_0003	u0010s0001_fo_li_0004
u0010s0002_fo_li_0001	u0010s0002_fo_li_0002	u0010s0002_fo_li_0003	u0010s0002_fo_li_0004
u0010s0001_fo_rm_0001	u0010s0001_fo_rm_0002	u0010s0001_fo_rm_0003	u0010s0001_fo_rm_0004
u0010s0001_fo_ri_0001	u0010s0001_fo_ri_0002	u0010s0001_fo_ri_0003	u0010s0002_fo_ri_0001
u0010s0002_fo_ri_0002	u0010s0002_fo_ri_0003	u0010s0002_fo_ri_0004	u0022s0001_fo_lm_0003
u0022s0002_fo_lm_0001	u0023s0001_fo_lm_0001	u0023s0001_fo_lm_0002	u0023s0001_fo_lm_0003
u0023s0001_fo_lm_0004	u0042s0001_fo_lm_0001	u0042s0001_fo_lm_0002	u0150s0001_fo_li_0001
u0150s0001_fo_li_0002	u0150s0001_fo_li_0003	u0150s0002_fo_li_0001	u0150s0002_fo_li_0002
u0150s0002_fo_li_0003	u0150s0002_fo_li_0004		

Tabella 2.3: Impronte più problematiche per Algoritmo A.

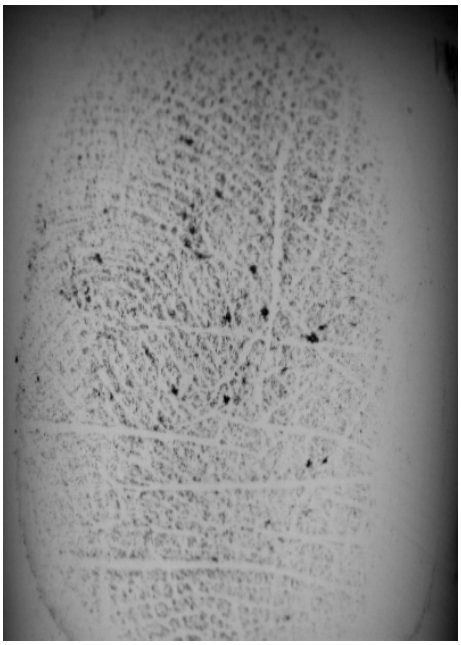
u0010s0001_fo_lm_0001	u0010s0001_fo_lm_0002	u0010s0001_fo_lm_0003	u0010s0001_fo_lm_0004
u0010s0002_fo_lm_0001	u0010s0002_fo_lm_0002	u0010s0002_fo_lm_0003	u0010s0002_fo_lm_0004
u0010s0001_fo_li_0001	u0010s0001_fo_li_0002	u0010s0001_fo_li_0003	u0010s0001_fo_li_0004
u0010s0002_fo_li_0001	u0010s0002_fo_li_0002	u0010s0002_fo_li_0003	u0010s0002_fo_li_0004
u0010s0001_fo_rm_0001	u0010s0001_fo_rm_0002	u0010s0001_fo_rm_0003	u0010s0001_fo_rm_0004
u0010s0001_fo_ri_0001	u0010s0001_fo_ri_0002	u0010s0001_fo_ri_0003	u0010s0002_fo_ri_0001
u0010s0002_fo_ri_0002	u0010s0002_fo_ri_0003	u0010s0002_fo_ri_0004	u0022s0001_fo_lm_0001
u0022s0002_fo_lm_0001	u0150s0001_fo_lm_0003	u0150s0002_fo_lm_0003	

Tabella 2.4: Impronte più problematiche per Algoritmo B.

Come è possibile notare, molte delle impronte risultano complesse sia per Algoritmo A che per Algoritmo B. In particolare quelle dell'utente 10, tutte caratterizzate in molti casi da un'eccessiva secchezza, combinata ad abrasioni e ad una cattiva qualità intrinseca dell'impronta. In Figura 2.6 vengono mostrati gli esempi più significativi.



(a) u0010s0001_fo_lm_0001



(b) u0010s0001_fo_ri_0003

(c) u0010s0001_fo_rm_0002

Figura 2.6: Alcune delle impronte più problematiche.

2.5.1 Gli errori compiuti da Algoritmo A nel processo di estrazione delle minuzie

Utilizzando l'applicazione *Minutiae Extraction Lab* (Figura 2.7), che consente di osservare ogni singola fase dell'estrazione delle minuzie con Algoritmo A, è stato possibile osservare in quali step l'estrattore compie la maggior parte degli errori, causando un'estrazione di minuzie ben lontana da quella ideale. Le fasi più dense di incorrettezze si sono rivelate essere l'estrazione delle orientazioni in punti di scarsa qualità dell'immagine e infine l'enhancement, che non sempre si comporta in maniera precisa in prossimità di tagli e abrasioni presenti sull'impronta. Esempi di questi errori sono visibili in Figura 2.8.

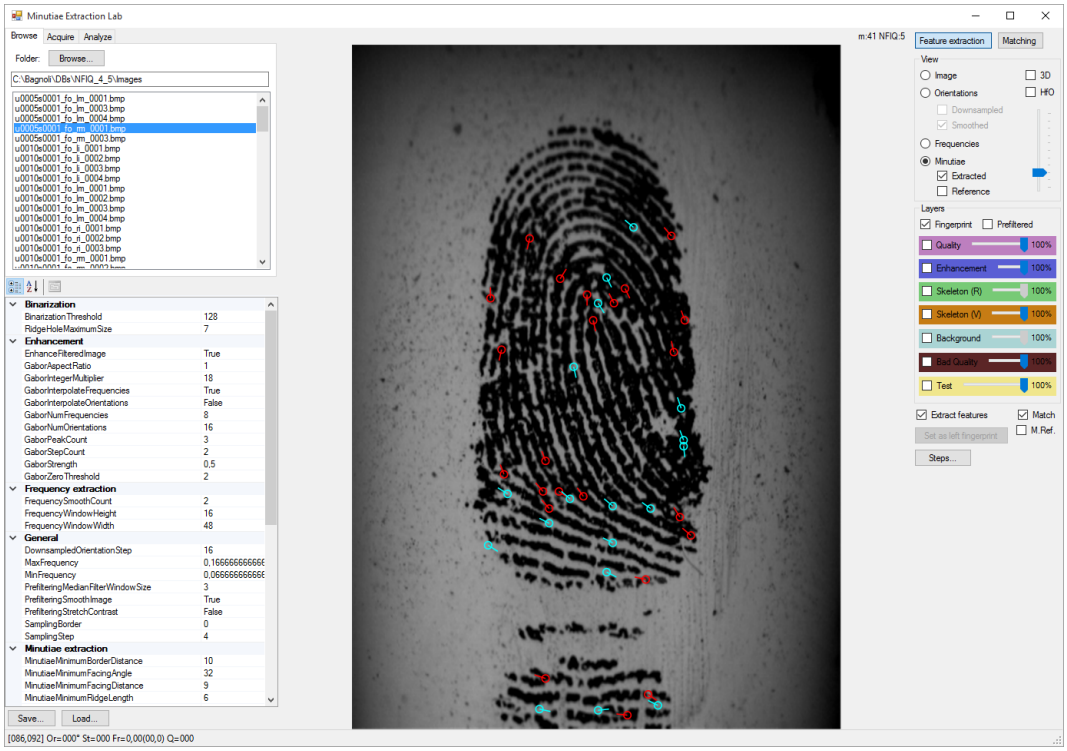
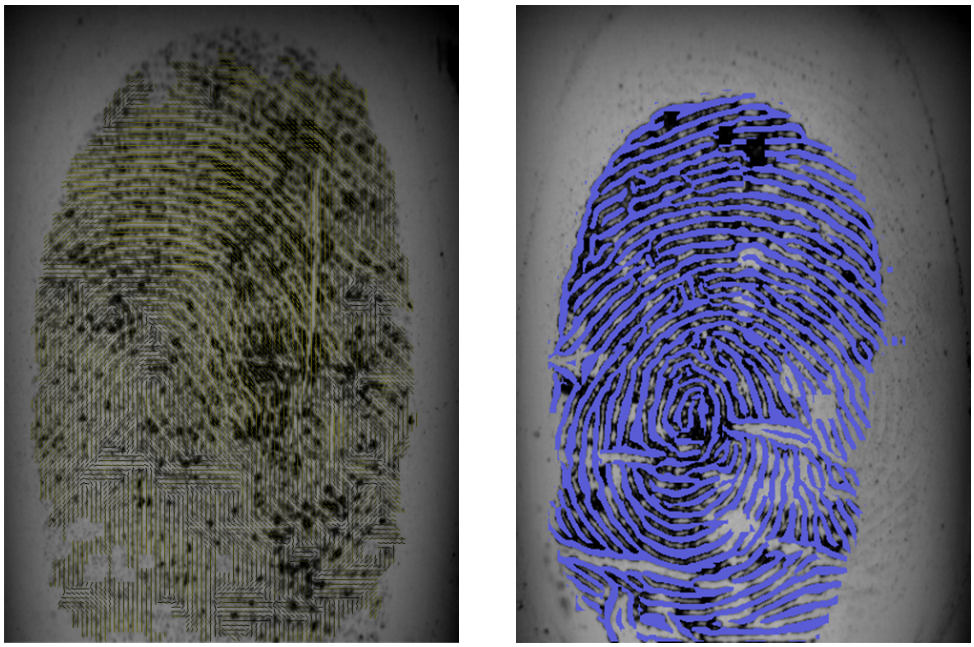


Figura 2.7: *Minutiae Extraction Lab.*



(a) Orientazioni sbagliate (b) Enhancement non ottimizzato

Figura 2.8: Errori nelle fasi precedenti all'estrazione delle minuzie.

2.5.2 Analisi sul numero di minuzie estratte da Algoritmo A

Segue ora una rapida analisi sul numero di minuzie estratte da Algoritmo A sulle 100 impronte prese in considerazione. Nello specifico, sono stati calcolati il numero medio di minuzie estratte dalle impronte che contribuiscono agli errori a FMR100 (riportate nella Tabella 2.3) e le restanti. Il primo è risultato essere 131 mentre il secondo 87, con una differenza piuttosto ampia. Questo divario ha messo in luce come nelle impronte più problematiche vengano tipicamente estratte un numero maggiore di minuzie, il più delle quali presumibilmente false. Per andare più a fondo alla questione puramente numerica delle minuzie, è stato realizzato un istogramma che mostra la distribuzione delle impronte per numero di minuzie (Figura 2.9).

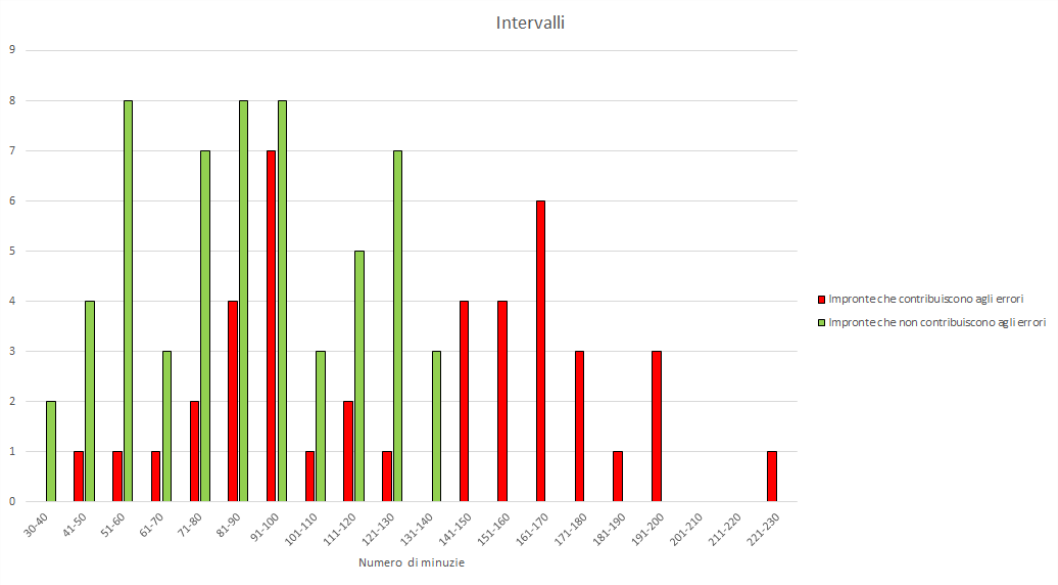


Figura 2.9: Distribuzione delle impronte per numero di minuzie. Viene riportato il numero di impronte che ha numero di minuzie compreso nel relativo intervallo.

Come si può notare, le uniche impronte con un numero di minuzie maggiore di 140 sono quelle responsabili degli errori a FMR100 e viene nuovamente confermata l'osservazione compiuta a seguito del calcolo del numero medio di minuzie: nelle impronte più difficoltose, Algoritmo A tende a estrarre un numero maggiore di minuzie.

Capitolo 3

Ottimizzazione di un algoritmo di estrazione delle minuzie

Come affermato nel corso del Capitolo 2 Algoritmo A, in impronte di bassa qualità, tende a compiere errori in due fasi cruciali del processo di estrazione: la determinazione delle orientazioni e l'enhancement. In questo capitolo verranno illustrati i passi seguiti per ottimizzare i risultati di Algoritmo A attraverso la modifica di alcuni parametri che controllano queste due fasi.

3.1 I parametri considerati

La fase di enhancement di un'immagine è progettata per ridurre il rumore e per migliorare la definizione delle creste dell'impronta. Ciò è reso possibile grazie ad una operazione di convoluzione utilizzando i filtri di Gabor. Il filtro di Gabor è un filtro lineare la cui risposta all'impulso è definita da una funzione armonica moltiplicata per una funzione Gaussiana. Per il teorema di convoluzione la trasformata di Fourier della risposta all'impulso di un filtro di Gabor risulta essere la convoluzione fra la trasformata di Fourier della funzione armonica e la trasformata di Fourier della funzione Gaussiana [2]. Esso viene considerato come uno degli strumenti più utili per la computer vision e l'elaborazione di immagini grazie alle sue proprietà ottimali per l'analisi spaziale e del dominio delle frequenze.

Un filtro di Gabor è in grado di ridurre il rumore presente in una regione dell'impronta quando la sua orientazione e la sua frequenza corrispondono a quelle delle creste in tale regione. Risulta utile quindi realizzare un banco di

filtri di Gabor e in base al numero di orientazioni e il numero di frequenze si realizzano n filtri differenti. Ad esempio, in Figura 3.1 si può notare un banco di 40 filtri, caratterizzato da 8 diverse orientazioni e 5 diverse frequenze.

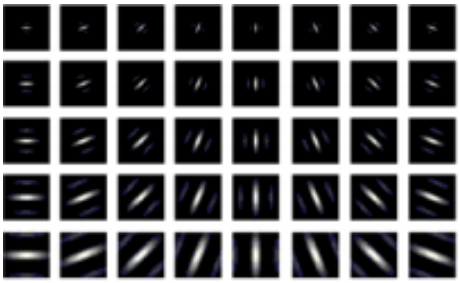


Figura 3.1: Banco di filtri di Gabor.

Una volta creato il banco di filtri, si effettua una convoluzione contestuale su tutta l'immagine. Il filtro da usare in ciascun pixel è quello che ha orientazione e frequenza più simili a quelle del blocco dell'immagine delle orientazioni e delle frequenze in cui ricade il pixel. I parametri dell'enhancement di Algoritmo A considerati e modificati vengono ora elencati e illustrati.

- **GaborPeakCount** - Il numero di picchi effettivi considerati nell'inviluppo Gaussiano durante la creazione del singolo filtro. Valore di default: 3.

- **GaborStepCount** - Il numero di volte che viene applicato il filtro di Gabor su ciascun pixel dell'immagine. Incrementare di una unità questo parametro equivale a fare un ulteriore ciclo di enhancement. Valore di default: 2.

- **GaborNumFrequencies** - Il numero di frequenze da usare per i filtri di Gabor, uniformemente distribuite nell'intervallo [MinimumFrequency, MaximumFrequency]. Valore di default: 8.

- **GaborNumOrientations** - Il numero di orientazioni da usare per i filtri di Gabor, espresse in radianti ed uniformemente distribuite nell'intervallo $[0, \pi]$. Valore di default: 16.

- **GaborInterpolateOrientations** - Flag booleano che indica se ottenere le orientazioni tramite interpolazione oppure direttamente dall'immagine delle orientazioni. Valore di default: false.

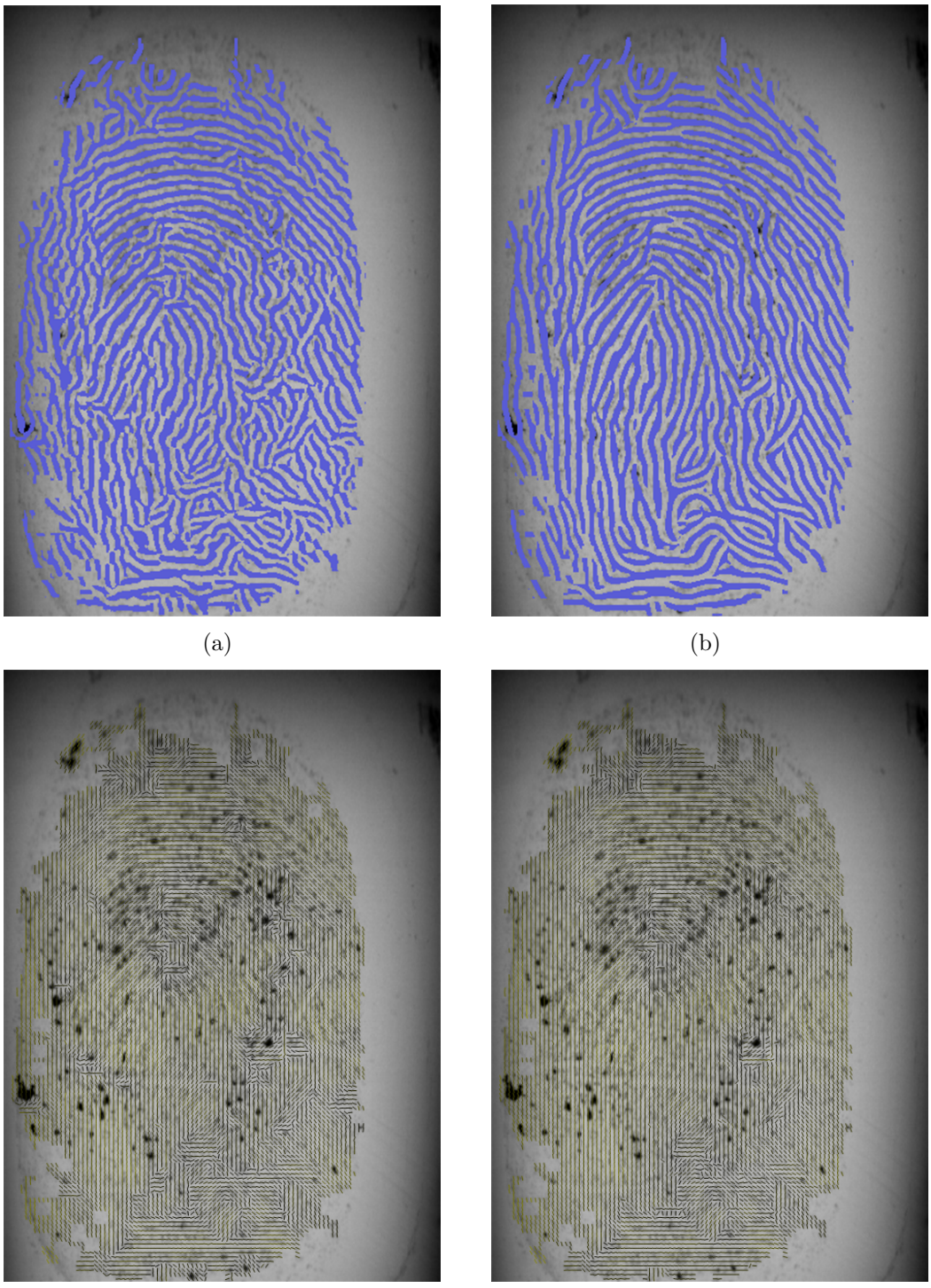
Per quello che riguarda le orientazioni, i parametri considerati sono invece i seguenti:

- **OrientationWindowSize** - La grandezza in pixel della finestra (di dimensioni n x n) su cui calcolare la media locale delle stime del gradiente, grazie al quale è possibile estrarre le orientazioni delle creste. Valore di default: 23.
- **OrientationMaximumSmoothCount** - Il numero massimo di smooth. Incrementare di una unità questo parametro equivale a calcolare un'ulteriore volta ciascuna orientazione come media di quelle vicine.
- **OrientationSmoothWindowSize** - La grandezza in pixel della finestra (di dimensioni m x m) su cui effettuare lo smoothing delle orientazioni. Valore di default: 5.

I parametri di enhancement e orientazioni sono stati modificati nella seguente combinazione:

- **GaborPeakCount** aumentato a 4.
- **GaborStepCount** aumentato a 5.
- **GaborNumFrequencies** aumentato a 12.
- **GaborNumOrientations** aumentato a 18.
- **GaborInterpolateOrientations** impostato a true.
- **OrientationWindowSize** aumentato a 25.
- **OrientationMaximumSmoothCount** aumentato a 9.
- **OrientationSmoothWindowSize** aumentato a 11.

Un confronto tra i risultati ottenuti su enhancement e orientazioni con questo set di parametri e con quelli originali è visibile in Figura 3.2. Come si può notare, le orientazioni con i nuovi parametri appaiono più coerenti con la realtà, e l'enhancement mostra segni di miglioramento evidenziando una maggiore uniformità.



(a) (b)

(c) (d)

Figura 3.2: Confronto tra risultati ottenuti con parametri di default (a sinistra) e modificati (a destra).

3.2 Le prove effettuate

Vengono ora riportati dettagliatamente tutti i test e le prove effettuate che hanno portato a considerare la combinazione precedentemente descritta come una candidata a rimpiazzare quella di default. Tutti i parametri sono stati modificati direttamente da *Minutiae Extraction Lab* per vedere in tempo reale come cambiavano i risultati su enhancement, orientazioni e minuzie estratte. È stato creato inoltre un oracolo[1] che, datogli in input un file di testo contenente i confronti di tipo genuine responsabili degli errori e la soglia necessaria a stabilire se il confronto è di tipo genuine o impostor, restituisce in output la percentuale di confronti genuine che continuano ad avere punteggio di similarità minore della soglia. Il codice dell'oracolo può essere osservato nel Listato 3.1.

```
1  class Oracle
2      {
3          private const string inputDBFolder =
               @"C:\Bagnoli\DBs\NFIQ_4_5\Images";
4          private const string imageExt = ".bmp";
5          private const string attemptFilePath =
               @"C:\Bagnoli\temp\attempts.txt";
6          private const string resultFolder = @"C:\Bagnoli\temp\Res";
7
8          private const bool parallelExecution = true;
9          private const double fmr100Thr = 0.113; //0.11235471617631362;
10
11         static void Main(string[] args)
12         {
13             var lines = File.ReadAllLines(attemptFilePath);
14
15             var scores = new double[lines.Length];
16             using (var progress = new SmartConsoleProgress(lines.Length,
                   2000))
17             {
18                 var partition = Partitioner.Create(0, lines.Length);
19                 var parallelOptions = new ParallelOptions()
20                 {
21                     MaxDegreeOfParallelism = (parallelExecution) ? -1 : 1,
22                 };
23                 Parallel.ForEach(partition, parallelOptions, p =>
24                 {
25                     var minutiaeExtractor = new MinutiaeExtraction();
26                     minutiaeExtractor.GaborNumFrequencies = 12;
27                     minutiaeExtractor.GaborPeakCount = 4;
28                     minutiaeExtractor.GaborStepCount = 5;
```

[1] Un meccanismo usato nel collaudo del software per determinare se un test ha avuto successo o è fallito [1].

```
                minutiaeExtractor.OrientationMaximumSmoothCount = 9;
                minutiaeExtractor.OrientationWindowSize = 25;
                minutiaeExtractor.OrientationSmoothWindowSize = 11;
                minutiaeExtractor.GaborNumOrientations = 18;
                minutiaeExtractor.GaborInterpolateOrientations = true;

            /*
            Estrazione delle minuzie coi nuovi parametri
            e matching delle impronte presenti nel file attemptFilePath.
            Riempimento del vettore scores.
            */
            ...

             }
             );
        }

        var errorCount = 0;
        var strBuilder = new StringBuilder();
        for (int i = 0; i < scores.Length; i++)
        {
            if (scores[i] < fmr100Thr)
            {
                errorCount++;
                strBuilder.AppendLine(string.Format("{0}\t{1:R}",
                    lines[i], scores[i]));
            }
            else
            {
                Console.WriteLine(string.Format("{0}\t{1:R}", lines[i],
                    scores[i]));
            }
        }

        File.WriteAllText(Path.Combine(resultFolder,"res.txt"),
                    strBuilder.ToString());
        Console.WriteLine(string.Format(CultureInfo.InvariantCulture,
                    "Error: {0:p2}",
                    (double) errorCount/scores.Length));
    }

}
```

Listato 3.1: L'oracolo sviluppato per verificare i risultati dei nuovi parametri sulle coppie di impronte più problematiche.

Come si può notare, i parametri sono stati modificati grazie alle proprietà dell'oggetto `minutiaeExtractor`. Dopo aver estratto le minuzie con la nuova configurazione e aver eseguito il confronto tra le coppie di impronte elencate in `attemptFilePath` viene riempito il vettore `scores` contenente i punteggi

di similarità di ciascun confronto. Successivamente vengono stampati a video i confronti con punteggio superiore alla soglia stabilita, insieme alla percentuale di confronti con punteggio inferiore. Come soglia, viene utilizzato il punteggio di similarità dell'ultimo confronto impostor a non dare errore con FMR fissato all'1%, chiamato FMR100 Score. Questo punteggio è preventivamente calcolato tramite lo stesso applicativo che calcola l'EER e l'FMR100, già utilizzato nella fase di analisi del Capitolo 2. L'FMR100 Score utilizzato come soglia è stato volutamente arrotondato per eccesso, come si può osservare nella riga 9 del listato, per rendere il test più severo.

I primi parametri considerati sono stati GaborPeakCount e GaborStepCount. Impostandoli rispettivamente a 6 e 10 sono venuti alla luce miglioramenti su alcune delle impronte problematiche ma un peggioramento generale di EER e FMR100. Successivamente sono stati abbassati a 4 e 5 portando i primi risultati soddisfacenti su EER e FMR100. Lavorando poi sui parametri relativi all'orientazione OrientationWindowSize, OrientationMaximumSmoothCount e OrientationSmoothWindowSize si sono potuti apprezzare altri piccoli miglioramenti anche per false e missed minuzie. Le ultime due modifiche, relative ai parametri GaborNumOrientations e GaborInterpolateOrientations, hanno evidenziato un ulteriore miglioramento generale di EER e FMR100. Sono state tentate infine alcune modifiche su parametri diversi da quelli di enhancement e orientazioni, ma senza risultati degni di nota.

3.3 Risultati ottenuti

Tutte le configurazioni provate:

1. GaborPeakCount = 6.
2. GaborPeakCount = 6; GaborStepCount = 10.
3. GaborPeakCount = 4; GaborStepCount = 5.
4. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25.
5. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25; OrientationMaximumSmoothCount = 9.

6. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25; OrientationMaximumSmoothCount = 9; GaborNumFrequencies = 12.

7. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25; OrientationMaximumSmoothCount = 9; GaborNumFrequencies = 12; OrientationSmoothWindowSize = 7.

8. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25; OrientationMaximumSmoothCount = 9; GaborNumFrequencies = 12; OrientationSmoothWindowSize = 11.

9. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25; OrientationMaximumSmoothCount = 9; GaborNumFrequencies = 12; OrientationSmoothWindowSize = 11; GaborNumOrientations = 18; GaborInterpolateOrientations = true.

10. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25; OrientationMaximumSmoothCount = 9; GaborNumFrequencies = 12; OrientationSmoothWindowSize = 11; GaborNumOrientations = 18; GaborInterpolateOrientations = true; FrequencyWindowHeight = 14.

11. GaborPeakCount = 4; GaborStepCount = 5; OrientationWindowSize = 25; OrientationMaximumSmoothCount = 9; GaborNumFrequencies = 12; OrientationSmoothWindowSize = 11; GaborNumOrientations = 18; GaborInterpolateOrientations = true; FrequencyWindowHeight = 14; FrequencyWindowWidth = 50.

I risultati di tutti i test effettuati sono visibili nella Tabella 3.1.

Configurazione	EER	FMR100	Missed	False	TestOracolo
0 (default)	10.94%	27.36%	451	9125	100.00%
1	12.26%	27.83%	557	6812	91.38%
2	13.20%	30.66%	801	6583	93.10%
3	8.96%	23.58%	501	8298	79.31%
4	7.56%	24.53%	498	8209	77.59%
5	8.01%	21.70%	498	8142	75.86%
6	7.31%	20.28%	499	8166	74.14%
7	8.10%	22.17%	490	7801	77.59%
8	9.42%	20.28%	496	7569	62.07%
9	6.32%	18.40%	472	7724	68.97%
10	8.07%	19.81%	474	7721	67.24%
11	8.08%	19.34%	471	7728	65.52%

Tabella 3.1: Risultati dei test effettuati con le varie configurazioni.

Gli esiti ottenuti su EER e FMR100 hanno portato a considerare la configurazione 9 come la migliore finora ottenuta per migliorare le prestazioni di Algoritmo A sul database di 100 impronte. Grazie a questa combinazione è possibile apprezzare infatti una diminuzione di EER del 4.62% e di FMR100 del 8.96%, oltre ad una discreta diminuzione del numero di false minuzie. La colonna TestOracolo mostra la percentuale restituita come output dall'oracolo precedentemente descritto. Il 32.03% dei confronti che prima risultavano come false non match, ora vengono correttamente riconosciuti come genuine.

Capitolo 4

Riduzione del numero di minuzie estratte

L'ottimizzazione eseguita ed illustrata nel Capitolo 3 ha portato un miglioramento sia sulle prestazioni del confronto, che sul numero di false minuzie. Tuttavia, alcune delle impronte più problematiche hanno ancora un alto numero di false minuzie, che continuano ad influenzare negativamente le prestazioni. Nel corso di questo Capitolo saranno analizzati e sperimentati alcuni metodi ideati per scartare alcune delle minuzie secondo determinati criteri.

4.1 I metodi sviluppati

Sono state implementate nove diverse strategie di scarto delle minuzie, di seguito elencate.

1. Scarto senza alcun controllo.
2. Scarto delle minuzie più a sinistra dell'immagine.
3. Scarto delle minuzie più a destra dell'immagine.
4. Scarto delle minuzie più in alto dell'immagine.
5. Scarto delle minuzie più in basso dell'immagine.
6. Scarto delle minuzie più lontane dal baricentro.

7. Scarto delle minuzie più vicine l'una dall'altra.

8. Scarto dei gruppi di minuzie più densi considerato un raggio di n pixel.

9. Scarto delle minuzie presenti nelle zone di scarsa qualità dell'immagine.

Il codice dei metodi che si occupano dello scarto si possono osservare nei Listati 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7.

```
private static MinutiaeTemplate BrutalDiscard(MinutiaeTemplate minutiae,
    int numberToKeep)
{
   var newMinutiae = (MinutiaeTemplate) minutiae.Clone();
   if (newMinutiae.Count > numberToKeep)
   {
      var selectedMinutiae = new Minutia[numberToKeep];
      for (int k = 0; k < selectedMinutiae.Length; k++)
      {
         selectedMinutiae[k] = newMinutiae[k];
      }
      newMinutiae = new MinutiaeTemplate(newMinutiae.ImageWidth,
                           newMinutiae.ImageHeight,
                           newMinutiae.ResolutionDpiX,
                           newMinutiae.ResolutionDpiY,
                           newMinutiae.FingerQuality,
                           selectedMinutiae,
                           null);
   }
   return newMinutiae;
}
```

Listato 4.1: Scarto senza alcun controllo.

```
private static MinutiaeTemplate LeftOrRightDiscard(MinutiaeTemplate
    minutiae, int numberToKeep, string leftOrRight)
{
   var newMinutiae = (MinutiaeTemplate)minutiae.Clone();
   if (newMinutiae.Count > numberToKeep)
   {
      var xCoordinates = new int[newMinutiae.Count];

      //Salvataggio delle coordinate X di tutte le minuzie e ordinamento
          crescente o decrescente
      for (int i = 0; i < newMinutiae.Count; i++)
      {
         xCoordinates[i] = newMinutiae[i].Location.X;
      }
      Array.Sort(xCoordinates);
      if (leftOrRight.Equals("right"))
```

```
15      {
16         Array.Reverse(xCoordinates);
17      }
18
19      //Salvataggio delle coordinate X delle minuzie da eliminare
20      int numberToDiscard = newMinutiae.Count - numberToKeep;
21      var xCoordinatesToDelete = new int[numberToDiscard];
22      for (int i = 0; i < xCoordinatesToDelete.Length; i++)
23      {
24         xCoordinatesToDelete[i] = xCoordinates[i];
25      }
26
27      //Creazione della lista delle minuzie da mantenere
28      var selectedMinutiae = new List<Minutia>();
29      for (int i = 0; i < newMinutiae.Count; i++)
30      {
31         if (!xCoordinatesToDelete.Contains(newMinutiae[i].Location.X))
32         {
33            selectedMinutiae.Add(newMinutiae[i]);
34         }
35      }
36
37      newMinutiae = new MinutiaeTemplate(newMinutiae.ImageWidth,
38                              newMinutiae.ImageHeight,
39                              newMinutiae.ResolutionDpiX,
40                              newMinutiae.ResolutionDpiY,
41                              newMinutiae.FingerQuality,
42                              selectedMinutiae,
43                              null);
44   }
45
46   return newMinutiae;
47}
```

Listato 4.2: Scarto delle minuzie più a sinistra o a destra dell'immagine, a seconda del parametro `leftOrRight`.

```
1  private static MinutiaeTemplate UpOrDownDiscard(MinutiaeTemplate minutiae,
       int numberToKeep, string upOrDown)
2  {
3     var newMinutiae = (MinutiaeTemplate)minutiae.Clone();
4     if (newMinutiae.Count > numberToKeep)
5     {
6        var yCoordinates = new int[newMinutiae.Count];
7
8        //Salvataggio delle coordinate Y di tutte le minuzie e ordinamento
             crescente o decrescente
9        for (int i = 0; i < newMinutiae.Count; i++)
10       {
11          yCoordinates[i] = newMinutiae[i].Location.Y;
12       }
```

```
13      Array.Sort(yCoordinates);
14      if (upOrDown.Equals("down"))
15      {
16         Array.Reverse(yCoordinates);
17      }
18
19      //Salvataggio delle coordinate Y delle minuzie da eliminare
20      int numberToDiscard = newMinutiae.Count - numberToKeep;
21      var yCoordinatesToDelete = new int[numberToDiscard];
22      for (int i = 0; i < yCoordinatesToDelete.Length; i++)
23      {
24         yCoordinatesToDelete[i] = yCoordinates[i];
25      }
26
27      //Creazione della lista delle minuzie da mantenere
28      var selectedMinutiae = new List<Minutia>();
29      for (int i = 0; i < newMinutiae.Count; i++)
30      {
31         if (!yCoordinatesToDelete.Contains(newMinutiae[i].Location.Y))
32         {
33            selectedMinutiae.Add(newMinutiae[i]);
34         }
35      }
36
37      newMinutiae = new MinutiaeTemplate(newMinutiae.ImageWidth,
38                              newMinutiae.ImageHeight,
39                              newMinutiae.ResolutionDpiX,
40                              newMinutiae.ResolutionDpiY,
41                              newMinutiae.FingerQuality,
42                              selectedMinutiae,
43                              null);
44   }
45
46   return newMinutiae;
47}
```

Listato 4.3: Scarto delle minuzie più in alto o in basso dell'immagine, a seconda del parametro `upOrDown`.

```
1  private static MinutiaeTemplate CenterOfGravityDiscard(MinutiaeTemplate
       minutiae, int numberToKeep)
2  {
3     var newMinutiae = (MinutiaeTemplate)minutiae.Clone();
4     if (newMinutiae.Count > numberToKeep)
5     {
6        //Calcolo del baricentro
7        var sumX = 0;
8        var sumY = 0;
9        for (int i = 0; i < newMinutiae.Count; i++)
10       {
11          sumX += newMinutiae[i].Location.X;
```

```
12          sumY += newMinutiae[i].Location.Y;
13       }
14       IntPoint2D centerOfGravity = new IntPoint2D(sumX / newMinutiae.Count,
             sumY / newMinutiae.Count);
15
16       //Salvataggio delle distanze di tutte le minuzie dal baricentro e
             ordinamento crescente
17       var distancesDictionary = new Dictionary<int, double>();
18       for (int i = 0; i < newMinutiae.Count; i++)
19       {
20          distancesDictionary.Add(i,
                IntPoint2D.CalculateEuclideanDistance(newMinutiae[i].Location,
                centerOfGravity));
21       }
22       List<KeyValuePair<int, double>> listKeyValue =
             distancesDictionary.ToList();
23       listKeyValue.Sort((firstPair, nextPair) =>
24          {
25             return firstPair.Value.CompareTo(nextPair.Value);
26          }
27       );
28       //Salvataggio degli indici delle minuzie da mantenere
29       var indexesToKeep = new int[numberToKeep];
30       for (int i = 0; i < numberToKeep; i++)
31       {
32          indexesToKeep[i] = listKeyValue[i].Key;
33       }
34       //Creazione della lista delle minuzie da mantenere
35       var selectedMinutiae = new List<Minutia>();
36       for (int i = 0; i < newMinutiae.Count; i++)
37       {
38          if (indexesToKeep.Contains(i))
39          {
40             selectedMinutiae.Add(newMinutiae[i]);
41          }
42       }
43
44       newMinutiae = new MinutiaeTemplate(newMinutiae.ImageWidth,
45                                newMinutiae.ImageHeight,
46                                newMinutiae.ResolutionDpiX,
47                                newMinutiae.ResolutionDpiY,
48                                newMinutiae.FingerQuality,
49                                selectedMinutiae,
50                                null);
51    }
52
53    return newMinutiae;
54 }
```

Listato 4.4: Scarto delle minuzie più lontane dal baricentro.

```
private static MinutiaeTemplate DistanceDiscard(MinutiaeTemplate minutiae,
    int numberToKeep)
{
   var newMinutiae = (MinutiaeTemplate) minutiae.Clone();
   if (newMinutiae.Count > numberToKeep)
   {
      //Salvataggio della distanza per ogni coppia di minuzia e ordinamento
          crescente
      int numberOfMinutiae = newMinutiae.Count;
      var distancesDictionary = new Dictionary<Pair<int, int>, double>();
          //chiave = coppia di minuzie, valore = distanza tra le due minuzie
      for (int i = 0; i < numberOfMinutiae; i++)
      {
         for (int j = i + 1; j < numberOfMinutiae - 1; j++)
         {
            var couple = new Pair<int, int>(i, j);
            double distance =
                IntPoint2D.CalculateEuclideanDistance(newMinutiae[i].Location,
                newMinutiae[j].Location);
            distancesDictionary.Add(couple, distance);
         }
      }
      List<KeyValuePair<Pair<int, int>, double>> listKeyValue =
          distancesDictionary.ToList();
      listKeyValue.Sort((firstPair, nextPair) =>
         {
            return firstPair.Value.CompareTo(nextPair.Value);
         }
      );

      //Salvataggio degli indici delle minuzie da scartare
      var indexesToDiscard = new HashSet<int>();
      for (int i = 0; i < numberOfMinutiae - numberToKeep; i++)
      {
         indexesToDiscard.Add(listKeyValue[i].Key.getFirst());
         if (indexesToDiscard.Count != numberOfMinutiae - numberToKeep)
         {
            indexesToDiscard.Add(listKeyValue[i].Key.getSecond());
         }
         if (indexesToDiscard.Count == numberOfMinutiae - numberToKeep)
         {
            break;
         }
      }

      //Creazione della lista delle minuzie da mantenere
      var selectedMinutiae = new List<Minutia>();
      for (int i = 0; i < newMinutiae.Count; i++)
      {
         if (!indexesToDiscard.Contains(i))
         {
            selectedMinutiae.Add(newMinutiae[i]);
         }
```

```
48      }
49
50      newMinutiae = new MinutiaeTemplate(newMinutiae.ImageWidth,
51                              newMinutiae.ImageHeight,
52                              newMinutiae.ResolutionDpiX,
53                              newMinutiae.ResolutionDpiY,
54                              newMinutiae.FingerQuality,
55                              selectedMinutiae,
56                              null);
57   }
58
59   return newMinutiae;
60 }
```

Listato 4.5: Scarto delle minuzie più vicine l'una dall'altra.

```
1  private static MinutiaeTemplate Distance2Discard(MinutiaeTemplate
       minutiae, int numberToKeep, int distanceThr)
2  {
3     var newMinutiae = (MinutiaeTemplate) minutiae.Clone();
4     if (newMinutiae.Count > numberToKeep)
5     {
6        //Salvataggio per ogni minuzia degli indici delle minuzie piu vicine
             della soglia
7        int minutiaeNumber = newMinutiae.Count;
8        List<List<int>> list = new List<List<int>>();
9        for (int i = 0; i < minutiaeNumber; i++)
10       {
11          list.Add(new List<int>());
12          for (int j = i + 1; j < minutiaeNumber - 1; j++)
13          {
14             double distance =
                   IntPoint2D.CalculateEuclideanDistance(newMinutiae[i].Location,
                   newMinutiae[j].Location);
15             if (distance < distanceThr)
16             {
17                list[i].Add(j);
18             }
19          }
20       }
21
22       //Conteggio delle minuzie piu vicine della soglia per ogni minuzia e
             ordinamento decrescente
23       var dictionaryNeighbors = new Dictionary<int, int>(); //chiave =
             indice della minuzia considerata, valore = numero delle minuzie
             vicine
24       for (int i = 0; i < list.Count; i++)
25       {
26          dictionaryNeighbors.Add(i, list[i].Count);
27       }
```

```
28      List<KeyValuePair<int, int>> listKeyValue =
            dictionaryNeighbors.ToList();
29      listKeyValue.Sort((firstPair, nextPair) =>
30         {
31            return firstPair.Value.CompareTo(nextPair.Value);
32         }
33      );
34      listKeyValue.Reverse();
35
36      //Salvataggio degli indici delle minuzie da scartare
37      var indexesToDiscard = new HashSet<int>();
38      for (int i = 0; i < listKeyValue.Count; i++)
39      {
40         var minutia = listKeyValue[i].Key;
41         for (int j = 0; j < list[minutia].Count; j++)
42         {
43            if (indexesToDiscard.Count < minutiaeNumber - numberToKeep)
44            {
45               indexesToDiscard.Add(minutia);
46            }
47            if (indexesToDiscard.Count < minutiaeNumber - numberToKeep)
48            {
49               indexesToDiscard.Add(list[minutia][j]);
50            }
51         }
52      }
53
54      //Creazione della lista delle minuzie da mantenere
55      var selectedMinutiae = new List<Minutia>();
56      for (int i = 0; i < newMinutiae.Count; i++)
57      {
58         if (!indexesToDiscard.Contains(i))
59         {
60            selectedMinutiae.Add(newMinutiae[i]);
61         }
62      }
63
64      newMinutiae = new MinutiaeTemplate(newMinutiae.ImageWidth,
65                              newMinutiae.ImageHeight,
66                              newMinutiae.ResolutionDpiX,
67                              newMinutiae.ResolutionDpiY,
68                              newMinutiae.FingerQuality,
69                              selectedMinutiae,
70                              null);
71   }
72
73   return newMinutiae;
74}
```

Listato 4.6: Scarto dei gruppi di minuzie più densi.

```
private static MinutiaeTemplate QualityDiscard(MinutiaeTemplate minutiae,
    int numberToKeep, Image<byte> quality)
{
   var newMinutiae = (MinutiaeTemplate) minutiae.Clone();
   if (newMinutiae.Count > numberToKeep)
   {
      //Salvataggio delle nuove coordinate mappate per ogni minuzia
      int minutiaeNumber = newMinutiae.Count;
      var newLocationsDictionary = new Dictionary<int, IntPoint2D>();
          //chiave = indice minuzia, valore = nuovo pixel
      for (int i = 0; i < minutiaeNumber; i++)
      {
         var newPoint = new
             IntPoint2D(Convert.ToInt32(newMinutiae[i].Location.X / 4),
             Convert.ToInt32(newMinutiae[i].Location.Y / 4));
         newLocationsDictionary.Add(i, newPoint);
      }

      //Salvataggio della qualità per ogni minuzia e ordinamento decrescente
      var qualityDictionary = new Dictionary<int, int>(); //chiave = indice
          minuzia, valore = qualità
      for (int i = 0; i < minutiaeNumber; i++)
      {
         int Y = newLocationsDictionary[i].Y;
         int X = newLocationsDictionary[i].X;
         qualityDictionary.Add(i, quality[Y, X]);
      }
      List<KeyValuePair<int, int>> listKeyValue =
          qualityDictionary.ToList();
      listKeyValue.Sort((firstPair, nextPair) =>
         {
            return firstPair.Value.CompareTo(nextPair.Value);
         }
      );
      listKeyValue.Reverse();

      //Salvataggio degli indici delle minuzie da mantenere
      var indexesToKeep = new int[numberToKeep];
      for (int i = 0; i < numberToKeep; i++)
      {
         indexesToKeep[i] = listKeyValue[i].Key;
      }

      //Creazione della lista delle minuzie da mantenere
      var selectedMinutiae = new List<Minutia>();
      for (int i = 0; i < newMinutiae.Count; i++)
      {
         if (indexesToKeep.Contains(i))
         {
            selectedMinutiae.Add(newMinutiae[i]);
         }
      }

```

```
48      newMinutiae = new MinutiaeTemplate(newMinutiae.ImageWidth,
49                              newMinutiae.ImageHeight,
50                              newMinutiae.ResolutionDpiX,
51                              newMinutiae.ResolutionDpiY,
52                              newMinutiae.FingerQuality,
53                              selectedMinutiae,
54                              null);
55      }
56
57      return newMinutiae;
58  }
```

Listato 4.7: Scarto delle minuzie presenti nelle zone di scarsa qualità dell'immagine.

Tutti i metodi presenti nei Listati prendono come parametri un oggetto di tipo `MinutiaeTemplate` contenente i dati sulle minuzie di un'immagine, e un intero chiamato `numberToKeep` che rappresenta la quantità di minuzie da mantenere. Come valore di ritorno hanno un nuovo oggetto `MinutiaeTemplate` contenente le sole minuzie da considerare. Alcuni dei metodi hanno anche altri parametri, ad esempio il metodo responsabile dello scarto dei gruppi più densi, prende in input anche un intero che descrive il raggio entro cui contare il numero di minuzie. Il metodo che si occupa dello scarto in base alla qualità dell'immagine prende in input una `Image<byte>` contenente i dati sulla qualità; si noti che per la particolare implementazione dell'algoritmo di analisi della qualità, tale immagine è sedici volte più piccola dell'immagine originale, in quanto la qualità è calcolata su blocchi di 4 x 4 pixel. Si è quindi reso necessario un mapping delle minuzie sull'immagine della qualità modificando opportunamente le coordinate X e Y delle minuzie.

4.2 Prove sperimentali

Tutte le prove effettuate coi metodi di scarto sono state eseguite con la soglia `numberToKeep` da 50 a 200, aumentandola ogni volta di 10 unità, mentre il metodo di scarto dei gruppi più densi è stato testato con raggio uguale a 20 e successivamente uguale a 30. I metodi sono stati testati in combinazione con la configurazione di parametri ottimizzata indicata nel Capitolo 3.

Per osservare in tempo reale quali minuzie vengono scartate coi vari metodi e con le varie soglie, è stata sviluppata un'applicazione Windows Form che mostra visivamente i risultati dei metodi di scarto sulle immagini. Come si

può notare dalla Figura 4.1, in cui viene mostrata la schermata dell'applicativo, le minuzie che vengono scartate sono evidenziate in giallo. Lo strumento sviluppato ha permesso inoltre di verificare che tenere in considerazione le prime n minuzie (quindi effettuare uno scarto senza controllo) di un oggetto `MinutiaeTemplate` contenente x minuzie equivale a scartare le $x - n$ minuzie partendo dal basso scartando prima le sole terminazioni e poi, eventualmente, le biforcazioni: questo corrisponde a quanto ci si attendeva in base al funzionamento dell'algoritmo di estrazione minuzie utilizzato che ricerca le minuzie appunto seguendo quest'ordine.

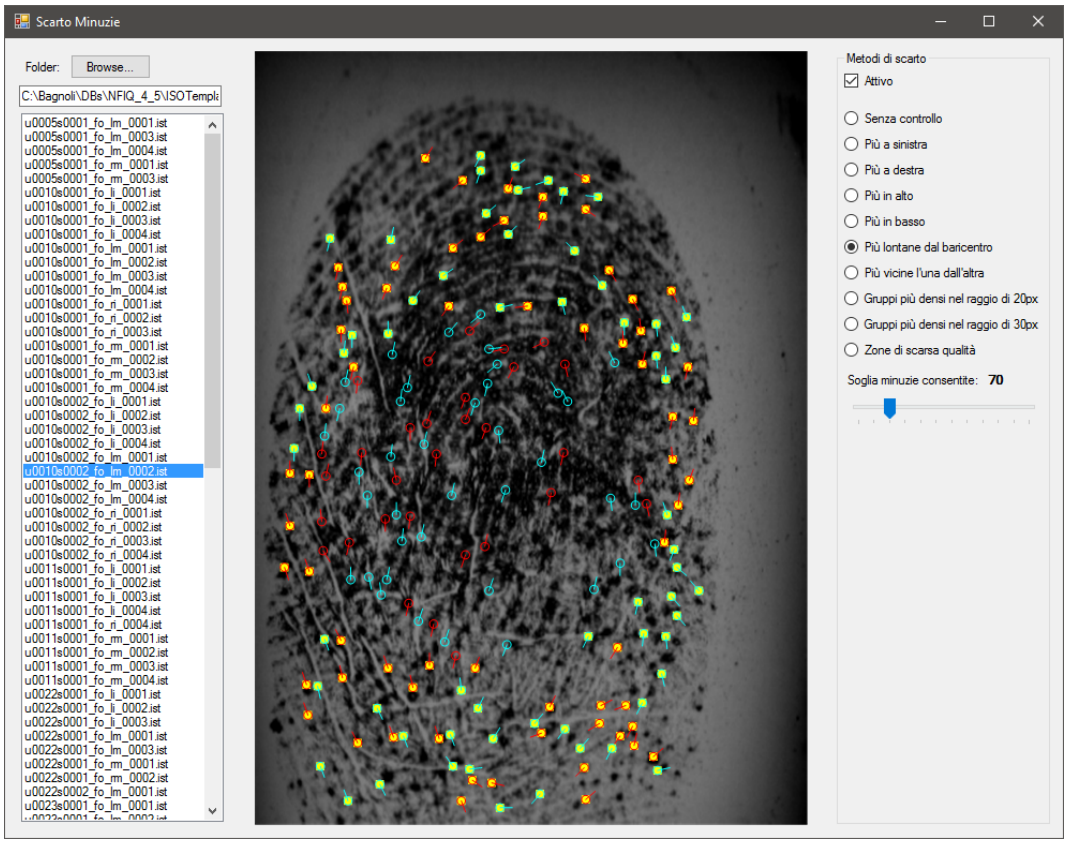


Figura 4.1: Applicazione Windows Form che mostra in tempo reale quali minuzie vengono scartate.

4.3 Analisi dei risultati

I risultati più interessanti ottenuti sul database di 100 impronte iniziale sono stati ottenuti con i seguenti metodi e soglie:

1. Scarto delle minuzie senza controllo con soglia delle minuzie consentite fissata a 150.

2. Scarto delle minuzie più a sinistra con soglia delle minuzie consentite fissata a 150.

3. Scarto delle minuzie più a destra con soglia delle minuzie consentite fissata a 130.

4. Scarto delle minuzie più in alto con soglia delle minuzie consentite fissata a 150.

5. Scarto delle minuzie più in basso con soglia delle minuzie consentite fissata a 140.

6. Scarto delle minuzie più lontane dal baricentro con soglia delle minuzie consentite fissata a 110.

7. Scarto delle minuzie più vicine l'una dall'altra con soglia delle minuzie consentite fissata a 160.

8. Scarto dei gruppi di minuzie più densi con soglia delle minuzie consentite fissata a 170 e raggio di valutazione fissato a 20 pixel.

9. Scarto dei gruppi di minuzie più densi con soglia delle minuzie consentite fissata a 170 e raggio di valutazione fissato a 30 pixel.

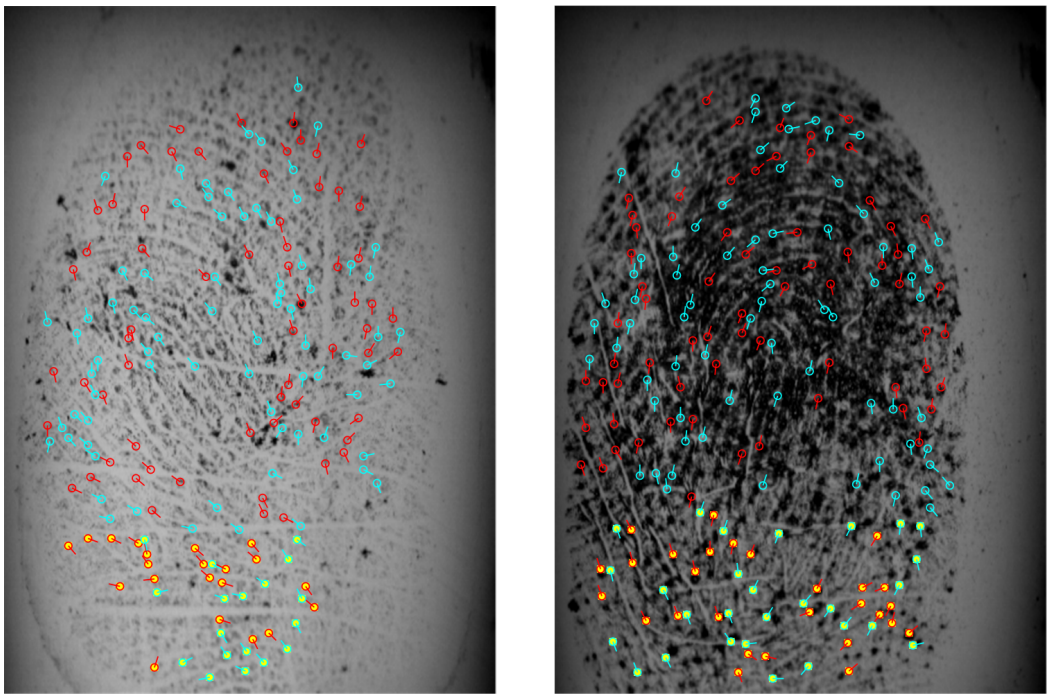
10. Scarto delle minuzie localizzate nelle zone di peggiore qualità dell'immagine con soglia delle minuzie consentite fissata a 160.

I risultati vengono riportati in Tabella 4.1.

Configurazione	EER	FMR100	Missed	False
0 (senza scarto)	6.32%	18.40%	472	7724
1	5.66%	16.51%	472	7514
2	6.31%	17.92%	472	7512
3	6.15%	18.87%	476	7173
4	6.13%	18.40%	483	7524
5	5.77%	18.40%	472	7359
6	6.13%	16.98%	567	6769
7	6.13%	17.45%	473	7614
8	6.24%	19.81%	472	7671
9	6.24%	18.40%	472	7671
10	6.19%	17.45%	472	7613

Tabella 4.1: Risultati dei test effettuati con i vari metodi di scarto.

Come si può osservare, lo scarto ha migliorato ulteriormente i risultati ottenuti con i soli parametri modificati illustrati nel Capitolo 3. Le migliorie più consistenti interessano le configurazioni con scarto delle minuzie senza controllo e delle minuzie situate più in basso nell'immagine. Molte delle impronte più problematiche hanno difatti un numero consistente di false minuzie nella parte inferiore dell'immagine, spesso caratterizzata da elevato rumore. Esempi significativi possono essere osservati in Figura 4.2.



(a) (b)

Figura 4.2: Scarto delle minuzie nella zona inferiore dell'immagine.

4.3.1 I risultati ottenuti su altri database

Gli ultimi test sono stati effettuati su altri tre differenti database biometrici contenenti impronte di tutte le qualità. Le prove su questi database sono state eseguite considerando:

- Parametri di default senza scarto di minuzie su tutte le impronte dei database.
- Parametri modificati senza scarto di minuzie solo sulle impronte di qualità NFIQ 4 e NFIQ 5.
- Parametri modificati con scarto di minuzie solo sulle impronte di qualità NFIQ 4 e NFIQ 5.

- Parametri di default con scarto di minuzie solo sulle impronte di qualità NFIQ 4 NFIQ 5.
- Parametri modificati senza scarto di minuzie su tutte le impronte dei database.
- Parametri modificati con scarto di minuzie su tutte le impronte dei database.
- Parametri di default con scarto di minuzie su tutte le impronte dei database.

Non avendo a disposizione il Ground Truth dei tre database, gli unici indici di prestazione considerati sono quelli relativi all'EER e al FMR. In particolare, oltre al FMR100, sono stati calcolati anche l'FMR1000 e lo ZeroFMR. Il primo rappresenta qual è l'FNMR ottenuto quando l'FMR è fissato allo 0.1%, mentre il secondo rappresenta qual è l'FNMR ottenuto quando l'FMR è fissato allo 0%.

I risultati più interessanti sui tre database sono stati ottenuti con i parametri modificati e scarto delle minuzie più in basso con soglia fissata a 140 solo sulle impronte con indice di qualità NFIQ 4 e NFIQ 5, evidenziando un miglioramento considerevole per quel che riguarda l'FMR100. Nella Tabelle 4.2, 4.3 e 4.4 sono elencati i dati ottenuti sui diversi database con questa configurazione, insieme ai risultati di Algoritmo B e Algoritmo A con parametri di default e senza scarto.

	EER	FMR100	FMR1000	ZeroFMR
Algoritmo B	0.85%	0.82%	1.07%	2.46%
Algoritmo A (default)	0.63%	0.64%	0.79%	1.07%
Algoritmo A (parametri modificati e scarto)	0.63%	0.61%	0.82%	1.07%

Tabella 4.2: Risultati dei test su Database 1.

	EER	FMR100	FMR1000	ZeroFMR
Algoritmo B	3.96%	5.61%	10.93%	13.61%
Algoritmo A (default)	4.00%	6.61%	10.79%	16.07%
Algoritmo A (parametri modificati e scarto)	4.07%	6.54%	11.11%	16.46%

Tabella 4.3: Risultati dei test su Database 2.

	EER	FMR100	FMR1000	ZeroFMR
Algoritmo B	0.07%	0.03%	0.08%	0.12%
Algoritmo A (default)	0.33%	0.25%	0.49%	0.71%
Algoritmo A (parametri modificati e scarto)	0.35%	0.19%	0.50%	0.79%

Tabella 4.4: Risultati dei test su Database 3.

Capitolo 5

Conclusioni

I dati ottenuti dai test dimostrano che migliorare le prestazioni di un algoritmo di estrazione delle minuzie su immagini di bassa qualità è possibile.

I risultati prodotti sul database di analisi iniziale con i parametri modificati e i metodi di scarto implementati sono soddisfacenti e in linea con le aspettative, confermando anche l'asserzione fatta alla fine del Capitolo 2, in cui veniva sottolineato il fatto che le impronte con numero di minuzie superiore a 140 fossero solo quelle responsabili degli errori di riconoscimento. Tuttavia i test effettuati successivamente su database diversi hanno evidenziato che le strategie adottate per il miglioramento delle prestazioni non sempre portano a risultati migliori dello stesso algoritmo con configurazione di default.

I risultati ottenuti lasciano comunque supporre che vi siano altri possibili margini di miglioramento, conseguibili con approcci differenti o più avanzati di quelli finora trattati.

Il lavoro di tesi è stato svolto come pianificato analizzando dapprima i casi più estremi in cui Algoritmo A commetteva errori e confrontandoli con Algoritmo B per poi ideare, sviluppare e testare le soluzioni proposte nei Capitoli precedenti.

Le attività di studio, analisi, implementazione e test delle prove effettuate per la realizzazione di questo lavoro di tesi sono state interessanti e motivanti. Mi hanno infatti permesso di approfondire conoscenze su argomenti precedentemente trattati in modo superficiale, come ad esempio la realizzazione di applicativi utilizzando il framework Microsoft .NET. È stato molto interessante, inoltre, acquisire le prime conoscenze nel campo della biometria, sia a livello teorico ma anche potendo mettere in pratica le nozioni apprese.

Ringraziamenti

Arrivato al termine di questo percorso universitario entusiasmante e ricco di soddisfazioni è giunto il momento di tirare le somme e ringraziare a dovere chi nel corso di questi tre anni mi è stato vicino, soprattutto nei momenti più faticosi e stressanti. Chi mi conosce sa bene che non sono una persona che ama i ringraziamenti plateali, ma in queste circostanze mi è impossibile non spendere qualche buona parola a favore di chi, queste parole, se le merita davvero.

Il primo ringraziamento va al relatore di questa tesi, il Prof. Raffaele Cappelli. Non è una semplice formalità, a lui sono riconoscente per avermi seguito e consigliato durante il periodo di tirocinio e di stesura della tesi, per la grandissima disponibilità offertami durante questi mesi e infine per avermi permesso di svolgere tirocinio e tesi all'interno del *Biometric System Laboratory* di Cesena. Lo stesso ringraziamento va al correlatore della tesi, il Dottor Matteo Ferrara, anche lui presenza fondamentale per lo svolgimento di tirocinio e tesi.

Un grazie a tutti coloro che mi sono stati vicini in università, in particolare i miei compagni di corso Luca e Filippo con i quali ho condiviso tanti momenti, dai meno impegnativi come un semplice pranzo a quelli più seri e delicati come la preparazione di esami o lo sviluppo dei tanti progetti svolti insieme. Senza di loro probabilmente non avrei imparato il significato di lavoro di squadra.

Un enorme ringraziamento va ovviamente ai miei genitori, per quanto mi sono stati vicini, per tutto l'aiuto dato, per avermi sempre fornito i consigli migliori e per avermi sostenuto, sia dal lato meramente economico che quello umano. Non avrei mai potuto compiere questo percorso senza di loro.

Ringrazio infine chiunque mi sia stato vicino in qualunque modo, anche solo con un messaggio di conforto o per sapere semplicemente come stavo.

Alessandro

Bibliografia

[1] Examples of test oracles.
http://www.testingeducation.org/k04/OracleExamples.htm.

[2] Gabor filter. https://en.wikipedia.org/wiki/Gabor_filter.

[3] Minutiae based matching. http://www.griaulebiometrics.com/.

[4] INCITS. Finger minutiae format for data interchange. Technical report, 2004.

[5] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer, 2009.

Elenco delle figure

Elenco delle tabelle