UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

Progettazione e sviluppo di una nuova applicazione per il riconoscimento di minuzie in un'impronta digitale.

Design and development of a new application for minutiae extraction from a fingerprint.

Relatore:

Prof. Aldo Franco Dragoni

Candidato:

Alessandro Muscatello

Anno accademico 2020/2021

Indice

1. Introduzione

1.1. Scopo della tesi

Questa tesi ha lo scopo di mostrare quali passi sono stato compiuti per l'implementazione di un nuovo algoritmo di riconoscimento ed estrazione delle minuzie delle impronte digitali. L'algoritmo si basa sul lavoro di Devansh Arpit e Anoop Namboodiri [1], pubblicato nel 2011, il quale introduce una serie di miglioramenti all'algoritmo originale di Dario Mario e Davide Maltoni [2]. Le tecniche più usate per l'estrazione di minuzie dalle impronte digitali prevedono una binarizzazione dei valori di grigio secondo un valore di soglia e successivamente il thinning delle creste. Lo scopo originale di Maio e Maltoni era quello di analizzare direttamente l'immagine in scala di grigi ed effettuare un tracciamento delle creste dell'impronta senza usare tecniche di filtraggio.

L'algoritmo che è stato sviluppato da Arpit e Namboodiri presenta degli importanti miglioramenti rispetto a quello precedente di Maio e Maltoni. In particolare, l'utilizzo del filtraggio con l'algoritmo CLAHE, il filtro con la tecnica in frequenza, l'utilizzo di parametri dinamici per la direzione e la larghezza delle ridge permettono un miglior inseguimento e riconoscimento delle creste con la conseguenza di ottenere un numero inferiore di minuzie spurie al termine dell'esecuzione del programma, pur mantenendo l'idea di fondo di agire direttamente sui valori di grigio dell'immagine.

Infine, questa tesi ha preso spunto dal lavoro compiuto dalla Dott.ssa Silvia Paolucci nella sua Tesi [3] "Progettazione e Sviluppo di un'applicazione di supporto alla dattiloscopia", nella quale ha mostrato la sua implementazione dell'algoritmo di Maio e Maltoni.

1.2. Storia degli studi delle impronte digitali

La dattiloscopia, ovvero lo studio delle impronte digitali, vede gli inizi in tempi recenti, intorno alla metà del XVII secolo. Nel 1858 l'ufficiale William Herschel in servizio nel Bengala, in India, riuscì per la prima volta ad utilizzare un'impronta di una mano per certificare l'autenticità di un documento, come fosse una firma. Già dall'anno successivo iniziò a raccogliere le impronte di indice e pollice dei suoi amici e conoscenti, finché non riuscì anche ad utilizzarle come tecnica di autenticazione solo per un anno, tra il 1877 e il 1878, per le ricevute delle pensioni governative. La sua raccolta di un ventennio gli permise di capire una delle caratteristiche più importanti delle impronte: l'immutabilità nel tempo. Quando Herschel ritornò in Inghilterra il suo successore non capì l'importanza dell'utilizzo delle impronte digitali, tanto che ne terminò l'impiego.

Nel frattempo, Henry Faulds, medico missionario in Giappone, iniziò i propri studi sulle impronte e fu il primo a pubblicare nel 1880 sulla rivista *Nature* il primo articolo che spiegava come si potessero utilizzare le impronte digitali per scopi di investigazione. Lo stesso anno Faulds inviò una lettera allo scienziato Charles Darwin nel tentativo di interessare un personaggio con una certa influenza pubblica circa l'importanza dello studio delle impronte. Qualche mese dopo che l'articolo fu pubblicato Herschel rispose rivendicando la paternità di questa tecnica che aveva messo in pratica già 20 anni prima. Darwin ormai vecchio non aveva la forza, anche a causa di una malattia, di seguire gli studi di Faulds; per cui inviò la lettera a suo cugino Sir Francis Galton che non se ne interessò fino al 1884 quando nel suo tentativo di creare la «razza con caratteristiche fuori dal comune» aveva iniziato a raccogliere i dati fisici e psicologici che avrebbero permesso di definire questa media evolutiva.

Il 25 maggio 1888 Galton tenne una conferenza alla *Royal Institution* descrivendo come Herschel avesse utilizzato le impronte digitali per mettere fine alle discussioni sull'autenticità degli atti notarili e parlando solo rapidamente dell'articolo di Faulds. Citando in questo modo i due interventi, Galton fece credere che Herschel fosse stato il primo ad aver pubblicato l'idea.

Le ricerche di Faulds furono comunque fondamentali per il lavoro di Galton, in particolare il suo sistema di classificazione delle impronte che utilizzava le lettere per distinguere alcuni tratti particolari delle impronte. Successivamente Galton riuscì anche a dimostrare che anche se confrontate le impronte di due gemelli, sebbene lo schema generale fosse simile, i "dettagli" delle impronte non corrispondevano mai. Questi dettagli verranno chiamati da Galton minuzie (*fingerprint minutia*).

Nel 1892 Galton pubblicò *Finger Prints,* in cui descrisse esaurientemente ogni aspetto della sua ricerca, facendo compiere alla scienza delle impronte digitali un grande passo in avanti. Galton fornì una prova sistematica della base scientifica delle impronte digitali, senza la quale non avrebbero mai potuto raggiungere il pubblico riconoscimento.

Infine, è degno di nota il lavoro svolto da Sir Edward Henry che nel 1899 realizzò un sistema di classificazione che semplificò molto l'identificazione che, al tempo, era realizzata manualmente da esperti. Il suo lavoro sarà la base per l'attuale sistema di classificazione che ancora oggi presenta alcuni tratti originari.

2. Le impronte digitali

2.1. Le caratteristiche biometriche

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

1) Universalità: tutte le persone devono possedere tale caratteristica biometrica
2) Unicità: la caratteristica deve essere sufficientemente distinguibile dagli altri individui
3) Permanenza: la caratteristica deve rimanere invariata nel tempo
4) Misurabilità: la caratteristica deve poter essere misurabile quantitativamente

Si possono distinguere due categorie di caratteristiche biometriche di un essere umano: fisiologiche e comportamentali.

Alcune caratteristiche comportamentali possono essere la firma o la voce di un individuo. In questi casi però lo stato fisico ed emotivo della persona possono farle variare notevolmente ed in poco tempo; quindi, non sono indicate come sistema di riconoscimento.

Le principali caratteristiche fisiologiche che possono essere utilizzate come identificatore biometrico di un essere umano che soddisfano i requisiti sono: le impronte digitali, la forma del viso, l'iride o la retina. I sistemi basati sull'iride o la retina, pur essendo sicuri e veloci, sono di difficile applicazione, mentre quelli basati sulle impronte digitali sono molto diffusi.

L'uso delle caratteristiche biometriche per il riconoscimento della persona presenta numerosi vantaggi rispetto ai tradizionali sistemi di autenticazione basati sull'uso di PIN o password poiché mentre questi ultimi 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 delle caratteristiche biometriche.

2.2. Anatomia

L'epidermide, ovvero lo strato più superficiale della cute, presenta dei rilievi che corrispondono alla sporgenza delle sottostanti papille del derma. Questi rilievi sono separati tra di loro da solchi e, insieme, formano i dermatoglifi, i quali danno origine a un

sistema di segmenti costituenti le impronte digitali. Queste strutture possono essere trovate su tutto il palmo delle mani e sulla pianta dei piedi.

La formazione delle impronte avviene durante la gestazione del feto e il loro sviluppo è univocamente determinato sia dal codice genetico, sia dallo specifico ambiente in cui esse si formano. Per questo anche due gemelli monozigoti, pur possedendo lo stesso codice genetico, avranno impronte digitali differenti. Inoltre, anche in caso di danneggiamenti superficiali della cute, l'impronta andrà a crescere esattamente come allo stato della nascita. Solo in caso di un danneggiamento profondo si andrà a modificare l'impronta, che acquisirà la nuova "caratteristica" per il resto della vita.

2.3. Struttura e classificazione delle impronte

Osservando le impronte digitali (Figura 1) si può notare come siano costituite da un flusso orientato di creste dette *ridge lines* e di valli chiamate *valleys* che, insieme, formano un complesso disegno detto *ridge pattern*.

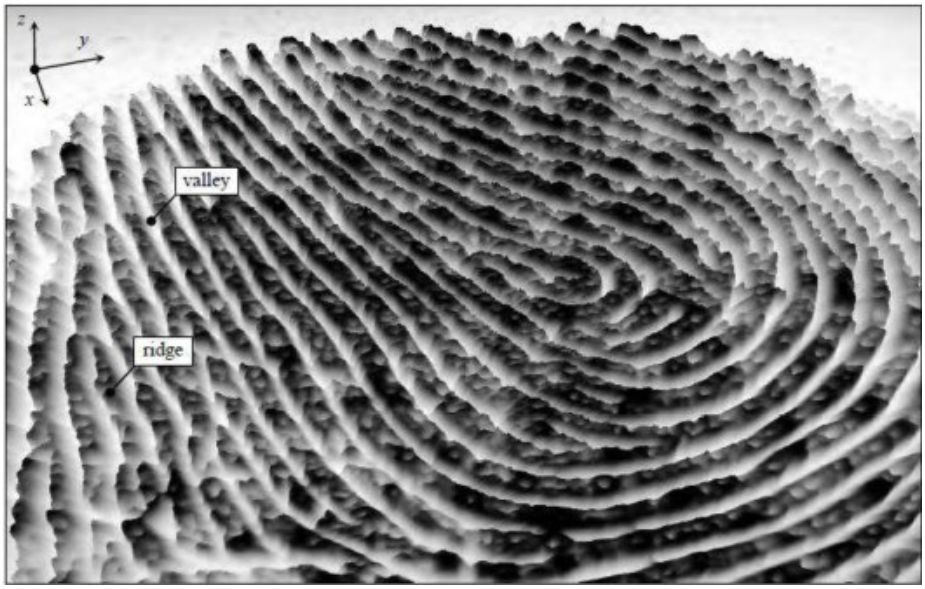

Figura 1: immagine 3D di un'impronta digitale

Per la descrizione di un'impronta si può procedere secondo tre livelli gerarchici:

A livello globale (Livello 1) si può notare come ogni impronta sia caratterizzata da delle forme distintive strutturali dove le ridges hanno una elevata curvatura. Queste regioni sono chiamate singolarità (Figura 2Figura 2), e sono essere classificate in tre tipi: cicli (*loop*), delta e spirali (*whorl*). Si chiama, inoltre, punto di *core* il "punto più a nord della ridge line più interna".

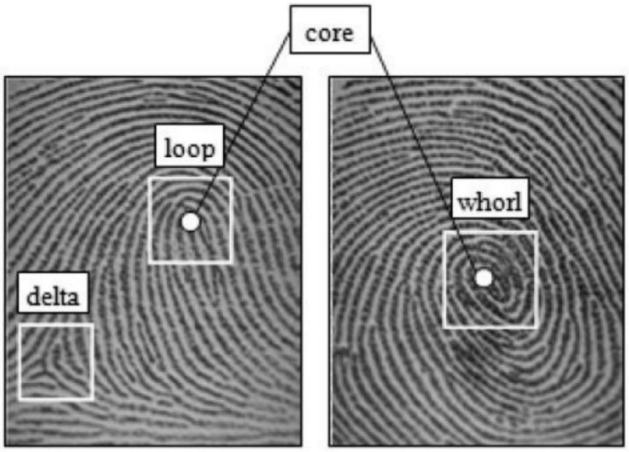

Figura 2: esempi di singolarità

Al Livello 2 possiamo identificare quelle che sono le minuzie (*minutiae*). La minuzia, dal nome stesso, è una particolarità dell'impronta in riferimento a come le ridges possano essere discontinue ed interrompersi improvvisamente (*ridge ending*), oppure possa dividersi in due nuove ridges (*bifurcation*). Sebbene l'FBI consideri solo queste due come uniche minuzie, in realtà esiste una classificazione anche per le loro combinazioni; le sette combinazioni più comuni sono riportate in Figura 3.

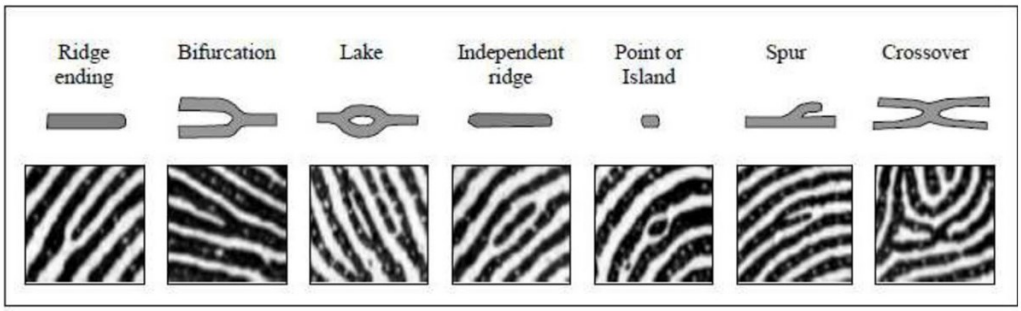

Figura 3: le sette tipologie più comuni di minuzie

Infine, al Livello 3 si indentificano i dettagli più profondi che possono essere estratti da un'impronta. Ne fanno parte gli attributi di dimensione e forma del contorno, i pori sudoripari infine tagli o pieghe naturali della pelle.

3. Descrizione del nuovo metodo di Ridge Following

3.1. Miglioramenti introdotti

I metodi tipici per l'estrazione di minuzie da un'impronta digitale sono solitamente composti da cinque fasi: miglioramento dell'immagine, binarizzazione mediante un valore di soglia detto threshold, creazione dello scheletro dell'impronta mediante thinning dell'immagine binarizzata, estrazione delle minuzie, e post processing.

Il miglioramento dell'immagine viene solitamente eseguito con dei filtri di Gabor e dei filtri nel dominio delle frequenze. Questi processi utilizzano un filtraggio pixel-per-pixel che tengono conto di un intorno di essi ma non dell'immagine nel suo complesso. Ciò rende robusti questi filtraggi nel senso che ogni punto filtrato non è affetto dal rumore delle regioni limitrofe. Nascono però dei problemi dal momento in cui questi metodi non sono in grado di risolvere ambiguità presenti in regioni particolarmente rumorose, risultando in delle formazioni di minuzie spurie nell'estrazione finale.

I metodi di binarizzazione sono stati largamente esplorati passando dall'approccio più semplice con un valore generale di threshold *t* e settando il valore di grigio in output pari a 1 se il grigio originale era maggiore del valore di threshold, 0 altrimenti; fino ad arrivare a tecniche che distinguono ridge e valleys secondo il segno della massima curvatura normale della superficie dell'impronta tridimensionale.

Le tecniche di thinning sono lo step immediatamente successivo alla binarizzazione. Essi consentono di ridurre la larghezza delle ridge fino ad un pixel, ottenendo così lo scheletro dell'impronta. Moltissimi approcci sono disponibili in letteratura, fatto dovuto all'utilizzo di queste tecniche anche in circostanze come l'estrazione dei caratteri in applicazioni OCR, analisi di manoscritti, vettorizzazione e disegno di mappe. Sfortunatamente, gli algoritmi di thinning sono fortemente soggetti ad aberrazioni e irregolarità dovute alla precedente fase di binarizzazione e che dunque porterà alla formazione di *spikes* nello scheletro che a loro volta porteranno alla formazione di minuzie spurie.

Il metodo presentato da Maio e Maltoni ha proposto una nuova tecnica di estrazione delle minuzie da un'impronta introducendo l'idea di tracciare le creste (*ridges*) direttamente sull'immagine in scala di grigi e conseguentemente individuare le minuzie che si incontrano durante il tracciamento senza effettuare le precedenti fasi di binarizzazione e thinning. L'obiettivo era quello di risolvere questi principali problemi:

i. Molte informazioni vengono perse durante la fase di binarizzazione.
ii. Binarizzazione e thinning richiedono molto tempo di calcolo.

iii. Vengono introdotte un alto numero di minuzie spurie.

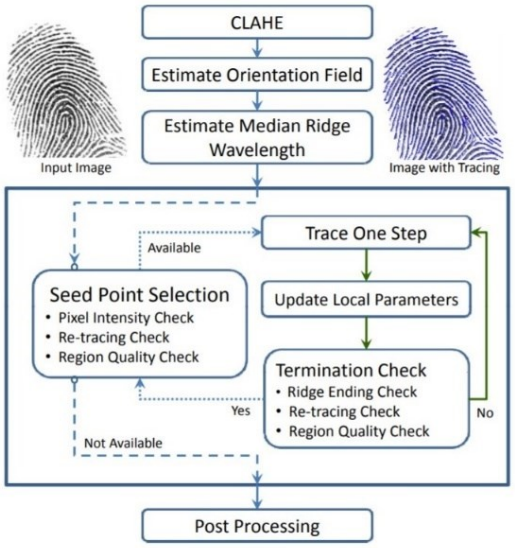

Figura 4: Step generali dell'algoritmo di Arpit e Namboodiri

Il nuovo metodo proposto da Arpit e Namboodiri [1] permette di superare alcuni limiti di quello suggerito da Maio e Maltoni [2]: gli step di alto livello sono molto simili, le differenze sono come questi step vengono effettivamente implementati (vedi Figura 4). In particolare, viene migliorato:

i. il *seeding* dei pixel da cui iniziare il tracing;
ii. la determinazione della direzione delle ridges durante il tracciamento: nello specifico vengono tenuti in considerazione i naturali disturbi come pori, variazione della larghezza della ridge, frequenza variabile delle ridge…
iii. terminazione del tracciamento quando viene incontrata una minuzia o una zona con forte rumore.

3.2. Spiegazione dell'algoritmo

3.2.1. Fasi preliminari

La prima operazione consiste nel filtrare l'immagine in base alla media e alla varianza dei valori di grigio per ottenere una immagine normalizzata che verrà utilizzata per il calcolo dell'orientation field.

La seconda fase preliminare dell'algoritmo prevede il calcolo dell'*orientation field* (Figura 5 a destra). Questo servirà per determinare la direzione iniziale del tracciamento una volta calcolato un seeding point. L'algoritmo utilizzato per ottenere l'orientation field è l'*LMS Orientation Estimation Algorithm* [3].

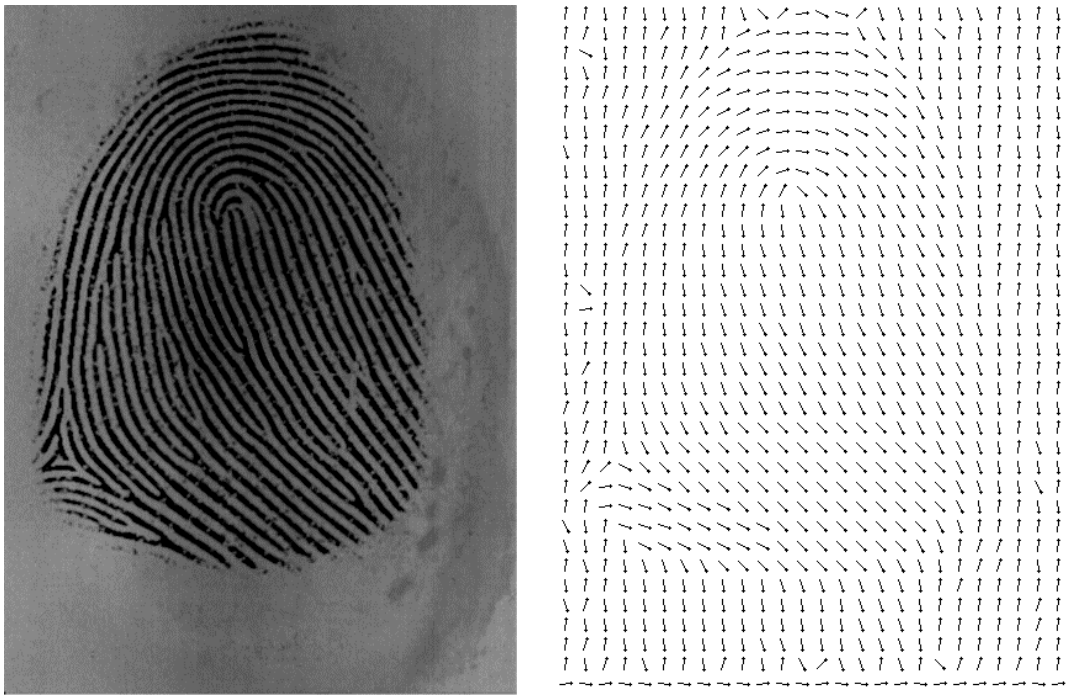

Figura 5: a sinistra l'immagine dell'impronta, a destra il corrispondente Orientation Field

L'LMS consiglia di ottenere l'orientation field dividendo l'immagine in blocchi di 16×16 pixel e stimare la direzione in ciascun blocco.

Per ognuno di questi blocchi viene quindi calcolata la larghezza media delle ridges presenti in essi, e per farlo verranno sfruttate alcune proprietà della trasformata di Fourier.

Per le successive letture sull'immagine si andrà ad utilizzare l'immagine filtrata attraverso un algoritmo CLAHE (*Contrast Limited Adaptive Histogram Equalization*). Questa tecnica di filtraggio è una comune operazione nell'image processing e permette di allargare l'istogramma dei livelli di grigio; in questo modo si ottiene un'immagine con un miglior contrasto, necessario all'algoritmo per distinguere le zone di forte rumore.

3.2.2. Algoritmo di tracciamento

Il processo di tracciamento ha come scopo quello di ottenere una singola linea continua per ogni ridge. Una volta ottenuto un seeding point (vedi Determinazione dei *seeding points*), è possibile iniziare il tracciamento della ridge: inizialmente seguendo la direzione precalcolata dall'orientation field, poi adattando dinamicamente la direzione ad ogni passo. Il tracciamento si muove su tutta la lunghezza della ridge registrando i punti che vengono attraversati, in modo da ottenere una rappresentazione digitale sia delle creste che delle minuzie rilevate.

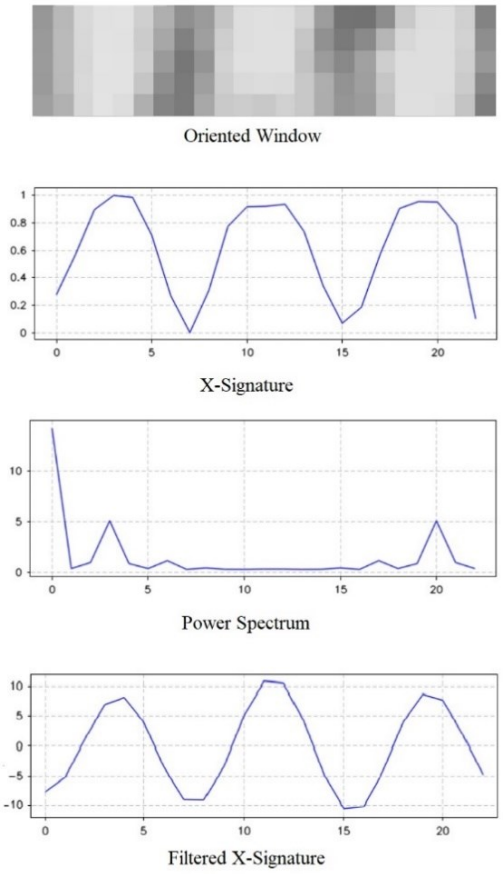

Figura 6: Oriented Window seguita dalla sua X-Signature, Power Spectrum della X-Signature ed infine X-Signature filtrata.

Il metodo adottato da Maio e Maltoni per tracciamento dei punti successivi utilizzava un approccio "step-and-correct" dato che veniva selezionato un punto ad una distanza fissa dal precedente e poi veniva eventualmente modificata la selezione per trovare il massimo locale. La principale miglioria apportata in questa fase è quella di utilizzare una *Oriented Window* che analizzi preventivamente la regione adiacente all'ultimo punto tracciato in direzione del tracciamento, con una distanza e delle dimensioni dinamicamente adattate alla larghezza locale della ridge.

La Figura 6 mostra una oriented window estratta, quindi la x-signature calcolata, che verrà poi analizzata attraverso il suo Power Spectrum, ed infine viene ricavata la x-signature filtrata. Da quest'ultima può essere identificato il picco che individua la nuova direzione di tracciamento. Questo filtraggio è fondamentale per permettere la selezione di punti anche in zone dove sono presenti dei pori della pelle che non permetterebbero all'algoritmo di effettuare un corretto inseguimento delle linee.

Un altro importante utilizzo dell'analisi spettrale è quello per la determinazione del valore di rumore presente all'interno di una zona. In questo modo si possono escludere dal seeding e dal tracciamento i punti che appartengono a queste zone.

3.2.3. Determinazione dei *seeding points*

Per iniziare il tracciamento delle ridges è prima necessario trovare un pixel che sia su una esse e in una regione in assenza rumore. Queste condizioni sono cruciali in quanto l'eventuale scelta di un pixel in una zona particolarmente rumorosa o al di fuori della cresta da analizzare porterebbe ad un incorretto tracciamento e dunque alla formazione di ridges spurie ed indesiderate nel risultato. Per assicurarsi che un punto possa essere selezionato come seed point bisogna: confrontare il suo valore di grigio con un valore di soglia (threshold), controllare il relativo Power Spectrum (come illustrato precedentemente) ed infine controllare che il punto non sia già stato tracciato in uno step precedente del tracciamento.

Per tenere traccia di tutti i pixel che vengono analizzati e selezionati dall'algoritmo di tracciamento viene usata un'immagine che ha la stessa dimensione di quella originale sulla quale vengono memorizzati questi punti. Ad ogni step l'immagine ausiliaria viene costantemente aggiornata per permettere un corretto tracciamento.

3.2.4. Condizioni di terminazione

Infine, le condizioni di terminazione sono necessarie per determinare quando interrompere l'inseguimento di una ridge. L'interruzione del tracciamento può avvenire per diverse cause:

i. l'entrata in una zona che presenta un rumore troppo elevato tale per cui è impossibile continuare il tracciamento;
ii. il caso in cui venga selezionato un punto che si trova su una ridge già tracciata. In questa circostanza verrà registrata una biforcazione sul punto selezionato;
iii. il raggiungimento della terminazione di una cresta. In questo caso verrà registrata una terminazione sul punto selezionato.

4. Implementazione dell'algoritmo

4.1. Linguaggio di programmazione e librerie

In questo capitolo si vogliono spiegare i passi fondamentali e le tecniche usate per implementare l'applicazione. Il linguaggio di programmazione che si è deciso di utilizzare è C++, e le librerie di supporto per lo sviluppo sono:

- OpenCV (ver. 4.5.2) per la manipolazione delle immagini;
- CvPlot (ver. 1.2.1) per il debug e visualizzazione dei grafici.

La scelta è ricaduta su di esse per la loro semplicità di utilizzo, il che ha permesso una rapida implementazione del programma. Di contro l'essenzialità, in particolare quella di OpenCV, si è rivelata essere una difficoltà; ad esempio, come spiegato successivamente nel corrente capitolo, non esiste una funzione di libreria che permetta l'estrazione di finestre orientate dalle immagini.

Fino al passo 4.4 (compreso) l'implementazione fa riferimento al paper *"Fingerprint image enhancement: Algorithm and performance evaluation."* [3]. Dal successivo step in poi si farà riferimento al paper "*Fingerprint feature extraction from gray scale images by ridge tracing, 2011 International Joint Conference on Biometrics (IJCB)*" [1].

4.2. Normalizzazione dell'immagine

La prima fase dell'algoritmo consiste in una normalizzazione dei grigi dell'immagine per permetterne una migliore manipolazione in fase di ottenimento della Oriented Field [3]. L'immagine in scala di grigi IMG è definita come una matrice di dimensione $M \times$ N dove $IMG(i,j)$ è il valore di grigio nel pixel (i,j). I passi per ottenere l'immagine normalizzata sono:

1) Calcolare la media M e la varianza VAR dell'immagine:

$$M(IMG) = \frac{1}{N^2}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} IMG(i,j),$$

$$VAR(IMG) = \frac{1}{N^2}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\big(IMG(i,j) - M(IMG)\big).$$

2) Calcolare l'immagine normalizzata g definita come:

$$h(i,j) = \begin{cases} M_0 + \dfrac{\sqrt{VAR_0(IMG(i,j)-M)^2}}{VAR} & se\ IMG(i,j) > M \\ M_0 - \dfrac{\sqrt{VAR_0(IMG(i,j)-M)^2}}{VAR} & altrimenti, \end{cases}$$

dove M_0 e VAR_0 sono i coefficienti desiderati per il filtraggio. $M_0 = 100$ e $VAR_0 = 1000$ sono i valori che sono stati utilizzati durante i test.

L'operazione di normalizzazione modifica i valori di grigio tenendo conto di tutta l'immagine (*pixel-wise operation*), ciò permette di ridurre le variazioni tra dei valori di grigio tra ridges e valli e quindi di migliorare la riuscita del prossimo step.

4.3. Orientation Field

Per poter calcolare l'orientation field si è fatto riferimento all'algoritmo l'*LMS Orientation Estimation Algorithm* [3].

Sia $h(i,j)$ l'immagine normalizzata sulla quale calcolare l'orientation field, i principali step sono:

1) Suddividere l'immagine in blocchi di 16×16 pixel.
2) Calcolare i gradienti $\partial_x(i,j)$ e $\partial_y(i,j)$ per ogni pixel dell'immagine. In questo caso è stato scelto di utilizzare l'operatore Sobel.
3) Stimare l'orientazione locale $\theta(i,j)$. Una stima per ogni blocco centrato nel pixel (i,j).

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{2h_{xy}}{h_{yy} - h_{xx}}\right)$$

dove

$$h_{xy} = \sum_{u=i-8}^{i+7}\sum_{v=j-8}^{j+7} \partial_x(u,v)\partial_y(u,v),$$

$$h_{xx} = \sum_{u=i-8}^{i+7}\sum_{v=j-8}^{j+7} \partial_x(u,v)^2,$$

$$h_{yy} = \sum_{u=i-8}^{i+7}\sum_{v=j-8}^{j+7} \partial_y(u,v)^2.$$

Matematicamente rappresenta $\theta(i,j)$ rappresenta la direzione che è ortogonale alla direzione massima dello spettro di Fourier della window. Le dimensioni della matrice θ risulteranno essere:

$$\dim(\theta) = (righe\ di\ h\ /\ 16) \times (colonne\ di\ h\ /\ 16).$$

4) A causa delle imperfezioni presenti sull'immagine, θ potrebbe non risultare in un campo vettoriale continuo: per questo è stato necessario applicare un filtro passa-basso che consente di ottenere il campo continuo $O(i,j)$.

Per applicare il filtro è necessario prima calcolare le due componenti lungo x e y:

$$\Phi_x(i,j) = \sum_{u=i-2}^{i+2} \sum_{v=i-2}^{i+2} W(u,v) \cos(2\theta(i-u, j-v)),$$

$$\Phi_y(i,j) = \sum_{u=i-2}^{i+2} \sum_{v=i-2}^{i+2} W(u,v) \sin(2\theta(i-u, j-v)).$$

Dove W è una matrice che contiene un filtro passa-basso bidimensionale, con unità integrale, di dimensione 5×5. (Vedi Figura 7)

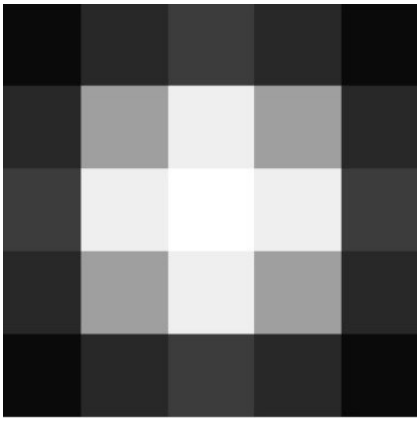

Figura 7: Filtro gaussiano bidimensionale

5) Infine, si può ottenere l'orientazione locale con

$$O(i,j) = \frac{1}{2} \tan^{-1}\left(\frac{\Phi_y(i,j)}{\Phi_x(i,j)}\right).$$

La matrice O è quella che contiene l'Oriented Field che è possibile vedere nella Figura 5 a destra.

4.4. Oriented Window, x-signature e larghezza locale delle ridge

Per ogni blocco 16×16 è ora possibile estrarre la oriented window locale, basandoci sull'orientazione precalcolata, dalla quale ottenere la x-signature, ovvero la media dei valori di grigio lungo l'altezza della window. Un chiaro esempio è illustrato nella Figura 8.

A livello implementativo l'estrazione dell'oriented window in realtà viene saltato poiché si procede con la lettura diretta della x-signature dalla matrice h.

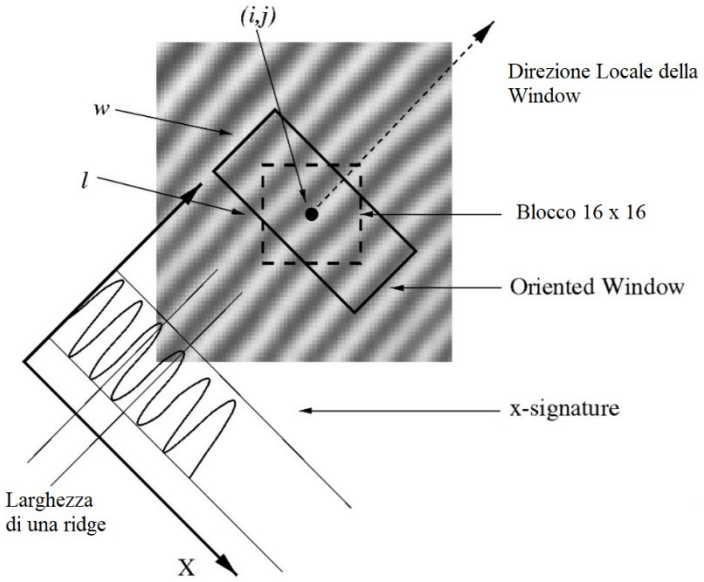

Figura 8: Estrazione di una oriented window e relativa x-signature

Sia g l'immagine dalla quale estrarre le x-signature, i principali step sono:

1) Dividere h in blocchi di 16×16 pixel
2) Calcolare direttamente la x-signature $X[0], X[1], \dots, X[l-1]$ per ogni blocco senza estrarre la oriented window:

$$X[k] = \frac{1}{w}\sum_{i=0}^{w} h(u,v), \qquad k = 0, 1, \dots, l-1,$$

con

$$u = i + \left(d - \frac{w}{2}\right)\cos O(i,j) + \left(k - \frac{l}{2}\right)\sin O(i,j),$$

$$v = j + \left(d - \frac{w}{2}\right)\sin O(i,j) + \left(\frac{l}{2} - k\right)\cos O(i,j),$$

$$w = 16,$$

$$l = 32,$$

dove w e l sarebbero le dimensioni di altezza e larghezza della window che avrebbe dovuto essere estratta.

Questo metodo di calcolo della sommatoria, però, soffre di un importante problema: la natura stessa dell'immagine è quella di avere dei pixel che sono discretizzati, per cui non è possibile selezionare valori di posizione dei pixel frazionari. Questo comporta una difficoltà nell'estrazione di finestre che sono ad angoli diversi da multipli di 90°, e quindi della quasi totalità delle window. Per cercare di risolvere il

problema in fase implementativa, si è deciso di adottare la funzione di libreria *LineIterator* che permette, dati due punti di un'immagine, di estrarre tutti i punti della linea che li congiunge. I punti di inizio e fine linea vengono selezionati in questo modo:

$$start(xStart, yStart),$$
$$end(xEnd, yEnd)$$

dove

$$xStart = (16j) + (k - 15)\sin O(i,j) - (-9)\cos O(i,j),$$
$$yStart = (16i) + (k - 15)\cos O(i,j) + (-9)\sin O(i,j),$$
$$xEnd = (16j) + (k - 15)\sin O(i,j) - 9\cos O(i,j),$$
$$yEnd = (16i) + (k - 15)\cos O(i,j) + 9\sin O(i,j).$$

In questo modo si estraggono 32 linee verticali per ogni window sulle quali si possono calcolare le relative 32 sommatorie per i valori della x-signature. Tra i punti di inizio e fine linea, però, non è comunque garantito che vengano selezionati esattamente 16 punti e che non vengano selezionati gli stessi punti più volte nel momento in cui si creano due linee adiacenti. Durante le successive fasi di determinazione dei seeding point e del tracciamento si è deciso di adottare un'ulteriore tecnica di estrazione delle x-signature mediante una traslazione e una rotazione dell'immagine.

Nell'ipotesi che non ci si trovi in una regione di forte rumore ed in assenza di minuzie, la x-signature mostrerà un'onda di forma sinusoidale che avrà la stessa frequenza delle ridge.

3) Calcolare il numero di ridge presenti nella window. Per ottenerlo si è deciso di applicare la trasformata discreta di Fourier (DFT) sulla x-signature grazie alla quale, dato un insieme finito di valori, si ottengono i coefficienti di una combinazione lineare di sinusoidi complesse ordinate al crescere della frequenza. Si prende poi l'indice ($maxIndex$) relativo al coefficiente di valore assoluto più alto per determinare la frequenza dominante della x-signature. Il numero di ridge locale è dato da $n = floor\left(\frac{maxIndex}{2}\right)$, dove $floor$ è la funzione per l'approssimazione per difetto.
4) Calcolare la larghezza media stimata delle ridges in pixel per ogni blocco: $\lambda_m(i,j) = \frac{colonne\ di\ \Omega(i,j)}{n(i,j)}$. Questo dato sarà poi utile nelle successive fasi.

Si vuole infine sottolineare che le dimensioni delle matrici O e λ_m saranno pari a $(righe\ di\ h\ /\ 16) \times (colonne\ di\ h\ /\ 16)$.

4.5. Filtraggio CLAHE

L'algoritmo CLAHE permette di modificare, allargandolo, l'istogramma delle frequenze di un'immagine. Questo produce l'effetto di aumentare il contrasto dell'immagine per permettere una migliore distinzione tra i livelli di chiaro e di scuro di ridges e valleys. Per l'implementazione si è deciso di utilizzare la funzione già presente all'interno della libreria di OpenCV. Si ottiene dunque la nuova matrice

$$g = CLAHE(IMG).$$

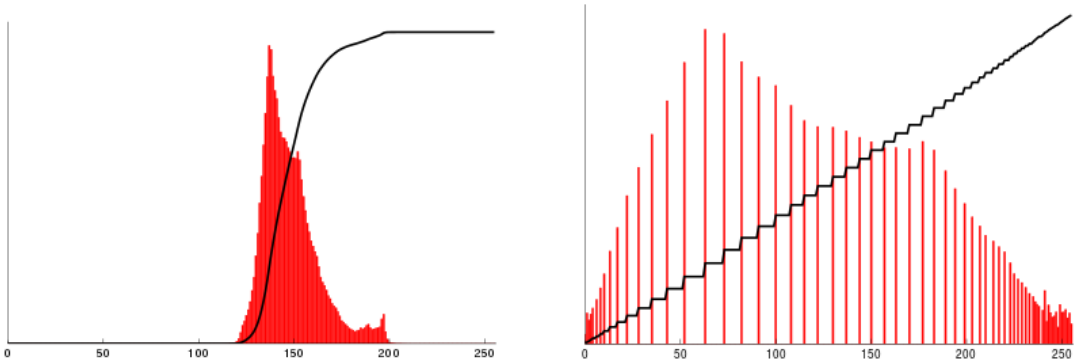

Figura 9: Istogramma di un'immagine prima e dopo l'applicazione dell'algoritmo CLAHE (in nero il valore cumulativo)

Per l'implementazione si è deciso di utilizzare la funzione già presente all'interno della libreria OpenCV.

4.6. Determinazione dei seeding points

Mentre le fasi precedenti possono essere considerate come preliminari all'algoritmo vero e proprio, la scelta di un seed point è la prima innovazione introdotta da Arpit e Namboodiri. Un *seed point* è un pixel dell'immagine che si trova su una ridge dell'impronta ed in una regione in assenza di rumore. La scelta del seed point è una fase cruciale perché un'eventuale selezione errata porterà alla formazione di segmenti spuri nel risultato finale.

Come già menzionato nell'introduzione, un seed point deve soddisfare le seguenti condizioni:

- Pixel Intensity Check: controlla se il punto è sufficientemente scuro;
- Retracing Check: controlla che il punto non è stato già precedentemente tracciato;
- Region Quality Check: controlla che la regione non sia troppo rumorosa.

4.6.1. Pixel Intensity Check

Dato che l'immagine è stata filtrata con l'algoritmo CLAHE, come primo check è possibile effettuare un semplice controllo di soglia. Un punto può iniziare qualificarsi come seed point se $g(i,j) < \alpha$, dove α è il valore di threshold. Nei test è stato utilizzato $\alpha = 25$.

4.6.2. Retracing Check

Per controllare che un punto non sia stato già tracciato si crea una seconda immagine I $(M \times N)$ della stessa dimensione dell'immagine originale che, all'avvio del programma, è interamente settata a 0. Ad ogni passo dell'algoritmo di tracciamento questa immagine verrà continuamente aggiornata con i nuovi punti tracciati e con il tratto di linea che li congiunge; in questo modo sarà possibile sia evitare la selezione di un seed point già segnato, sia, durante il tracciamento, rilevare le biforcazioni.

In questa fase si è deciso di adottare una tecnica diversa per l'estrazione delle oriented window poiché il metodo precedentemente descritto nelle fasi preliminari non permette una corretta estrazione dei pixel a causa della naturale discretizzazione presente nelle immagini; in più, la libreria OpenCV non ha una vera e propria funzione per ritagliare una porzione ruotata di un'immagine. Infatti, questa seconda implementazione prevede una traslazione ed una rotazione dell'immagine g ed I, per cui si avrà l'oriented window da estrarre al centro dell'immagine già ruotata, e quindi più facile da manipolare. I passi per l'implementazione e per effettuare il check sono:

1) Traslare l'immagine g in modo che al centro dell'immagine ci sia il pixel (i,j) sul quale estrarre l'oriented window. I valori per la traslazione sono:

$$traslazione\ lungo\ x = \frac{lunghezza\ dell'immagine}{2} - i,$$
$$traslazione\ lungo\ y = \frac{larghezza\ dell'immagine}{2} - j.$$

   Si ottiene così l'immagine traslata g'.
2) Ruotare in senso orario g' di $\theta - 90°$, in modo da ottenere le ridges locali in direzione verticale e dove $\theta = O\left(floor\left(\frac{i}{16}\right), floor\left(\frac{j}{16}\right)\right)$.
3) Estrarre al centro dell'immagine g' una window Ω di dimensioni $\lambda_m \times \lambda_m$, dove λ_m è la larghezza media delle ridge nel blocco 16×16 nel quale è contenuto il pixel (i,j).

4) Binarizzare la window Ω rispetto alla sua intensità media (τ) in modo da ottenere le ridge ad 1 e le valli a 0.

$$\Omega'(l,m) = \begin{cases} 1, & \Omega(l,m) < \tau \\ 0, & altrimenti. \end{cases}$$

5) Similarmente estrarre un'altra oriented window Ω_t (step da 1 a 3), centrata nel pixel (i,j), di dimensioni $\lambda_m \times \lambda_m$ dalla matrice I.
6) Moltiplicare la matrice Ω' ed Ω_t elemento per elemento ($\Omega' .* \Omega_t$).
7) Sommare tutti gli elementi della matrice risultate per ottenere lo scalare $T = sum(\Omega' .* \Omega_t)$.

Una ridge si considera già attraversata se $T > 0$, quindi il punto (i,j) viene scartato, altrimenti si passa al successivo Check.

Il metodo di estrazione delle window spiegato nei passi precedenti sarà quello usato in tutte le fasi successive per i motivi sopra descritti.

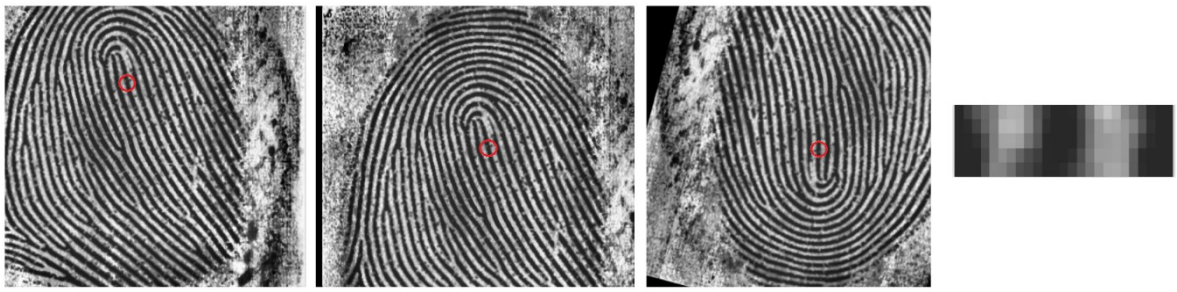

Figura 10: traslazione, rotazione e taglio dell'immagine per l'estrazione dell'oriented window. Il punto sul quale avviene l'estrazione è segnato con un cerchio.

4.6.3. Region Quality Check

Se un punto ha superato i due controlli precedenti, si passa ad analizzare la rumorosità della regione ad esso circostante:

1) Estrarre una oriented window Ω centrata sul pixel (i,j) di dimensioni $5\lambda_m \times \lambda_m$ $(w \times l)$ ed orientata in direzione delle ridges.
   In questo caso si è deciso di non utilizzare la larghezza consigliata dal paper per la larghezza della window dato che, nel passo 4), sarà necessario calcolare il Power Spectrum della finestra e, nel caso si abbia una larghezza non sufficiente, non sarà possibile calcolare questo valore con precisione. Per questo la larghezza è stata portata da $2.5\lambda_m$ a $5\lambda_m$.

2) Ricalcolare i valori d'intensità di grigio della window estratta in modo da invertire i bianchi con i neri. Le ridges saranno quindi chiare e le valli scure.

$$\Omega'(l,m) = f - \Omega(l,m),$$
$$\forall (l,m) \in \Omega,$$

dove f è il numero dei valori di grigio ammessi (in un'immagine a 8 bit pari a 256 valori).

3) Calcolare la x-signature $X[0, 1, \dots, 5\lambda_m]$ della window Ω' dove:

$$X[i] = \frac{1}{\lambda_m} \sum_{j=0}^{\lambda_m} \Omega'(\mathrm{j}, \mathrm{i}).$$

Quindi normalizzarla in modo da ottenere $X'[i] \in [0,1]$.

4) Calcolare il Power Spectrum $P[k]$ dove $k = 1$ indica la prima armonica, $k = 2$ indica la seconda armonica…

5) Calcolare il numero di ridge presenti nella window come spiegato precedentemente.

6) Calcolare l'*Harmonic Ratio* (HR) definito come:

$$HR = \frac{max(P[n-1], P[n], \dots, P[n+3])}{max(P[0], P[1], \dots, P[n-2])},$$

dove n è il numero totale di ridge presenti nella window.

7) Se un pixel ottiene un valore di $HR > \beta_1$ allora può essere considerato un seed point. Il valore di β_1 che è stato usato nei test è di 1.3. Nel caso in cui il pixel venga scartato verrà selezionato un nuovo pixel dell'immagine g sul quale verranno sottoposti tutti i Check precedentemente descritti.

Il paper [1] non giustifica in maniera esauriente l'utilizzo dell'Harmonic Ratio come tecnica per la selezione dei seeding point, ed inoltre non fornisce alcun valore di default a β_1. Gli autori si sono limitati solo a scrivere che, dalle loro osservazioni, hanno notato che le regioni rumorose presentano picchi nelle frequenze più basse piuttosto che in quelle alte e che quindi un rapporto tra il picco all'interno della regione di frequenze accettabili e il picco delle frequenze più basse fornisce una buona stima della rumorosità della regione. L'HR è stato dunque interpretato come una "misura del rumore": più questo valore è alto, più la frequenza ammissibile è dominante sul rumore, e ciò porterebbe all'accettazione del pixel come seed point.

4.7. Tracciamento

Il tracciamento è la fase in cui, a partire da un seeding point selezionato, si cercano tutti i punti della ridge a cui appartiene. Durante il processo verranno create due stringhe di punti: una lungo la direzione (θ_o) stimata nell'Orientation Field e l'altra nella direzione opposta. Queste due stringhe saranno poi unite quando si verifica una condizione di stop per formare un'unica ridge continua. La larghezza della stringa verrà dinamicamente ricalcolata per far fronte alle variazioni locali. I passi del tracciamento sono:

1) Selezionare un punto di test in direzione θ_o distante dal seed λ.

$$x_{test} = x_{seed} * \lambda * \cos(\theta_o),$$
$$y_{test} = y_{seed} * \lambda * \sin(\theta_o).$$

   In questo caso λ e θ_o sono la larghezza della ridge nel punto e la direzione della oriented window che sono stati calcolati allo step precedente del tracciamento. Come primi valori di λ e θ_o si usano i valori precalcolati nella fase preliminare.
2) Estrarre una oriented window Ω centrata sul pixel di test (i, j) di dimensioni $2.5\lambda \times \lambda$ $(w \times l)$ con un angolo θ_o.
3) Ricalcolare i valori d'intensità del grigio della window estratta in modo da invertire i bianchi con i neri. Le ridges saranno quindi chiare e le valli scure.

$$\Omega'(l, m) = f - \Omega(l, m),$$
$$\forall (l, m) \in \Omega,$$

   dove f è il numero dei valori di grigio ammessi (in un'immagine con uno spazio di colore di 8 bit è pari a 256 valori).
4) Calcolare la x-signature $X[0, 1, \dots, \lambda]$ della window Ω' come mostrato precedentemente e normalizzarla in modo da ottenere $X'[i] \in [0,1]$.
5) Calcolare il Power Spectrum $P[k]$ dove $k = 1$ indica la prima armonica, $k = 2$ indica la seconda armonica…
6) Calcolare il numero di ridge presenti nella window come mostrato precedentemente.
7) Calcolare l'*Harmonic Ratio* (HR) definito come:

$$HR = \frac{max(P[n-1], P[n], \dots, P[n+3])}{max(P[0], P[1], \dots, P[n-2])},$$

   dove n è il numero totale di ridge presenti nella window.

8) Se $HR < \beta_2$ allora ripetere i passi da 1 a 7 aumentando la larghezza della oriented window da 2.5λ a 4.5λ. Nel paper non viene specificato nessun valore per β_2; nei test è stato scelto $\beta_2 = 0.3$. Nel caso in cui il valore di HR sia ancora sotto al valore di soglia terminare la stringa.
9) Se il test ha successo, effettuare la trasformata discreta di Fourier (DFT) alla x-signature. Dato un segnale normalizzato con $n\ (> 1)$ cicli, la trasformata mostrerà un picco sulla *n-esima* armonica. Eliminare le armoniche al di fuori del range $[n-1, n+3]$, ed effettuare quindi la trasformata inversa per ottenere la x-signature filtrata X_{new}.
10) Calcolare la distanza Δc in pixel tra il centro della oriented window e la ridge che si sta tracciando. Per farlo si cerca la distanza tra il massimo locale in X_{new} e il centro della window in modo che:

$$X_{new}[c + \Delta c - 1] < X_{new}[c + \Delta c] > X_{new}[c + \Delta c + 1]$$
$$X_{new}[i] < X_{new}[c], \forall i : \ c < i < c + \Delta c.$$

La prima disequazione è utilizzata per esprimere la condizione di massimo locale del punto $X_{new}[c + \Delta c]$, mentre la seconda per assicurarsi che non si salti erroneamente da una ridge all'altra durante il tracciamento.
11) Calcolare $\theta_c = \left(\frac{\Delta c}{0.5\lambda}\right)$. Il nuovo angolo corretto per il tracciamento è $\theta_t = \theta_o + \theta_c$.
12) Il nuovo nodo della stringa è dato da:

$$i_{new} = i + 0.5\lambda \cos(\theta_t),$$
$$j_{new} = j + 0.5\lambda \sin(\theta_t).$$

Marcare sull'immagine I' tutti i punti che si trovano sul percorso tra il precedente ed il nuovo nodo (esso compreso).

In questo step la libreria OpenCV mostra uno dei suoi difetti che si era in parte già presentato quando si aveva utilizzato la funzione *lineIterator*: nel momento in cui si richiede di disegnare una linea, i valori di spessore che si possono utilizzare sono puramente interi ed inoltre per valori pari la funzione non disegna bene la linea a causa dell'assenza di anti-aliasing. Tale difetto si accentua se i punti di inizio e fine linea formano un angolo diverso da $\frac{n\pi}{2}$.
13) In ultimo calcolare la larghezza in pixel della ridge locale e archiviarla per il ciclo di step successivo.

14) Ripetere gli step precedenti finché non si verifica una condizione di stop.

4.8. Condizioni di terminazione

Sono le condizioni per cui interrompere il tracciamento. Una condizione è già stata implementata ovvero il caso in cui si finisca in una zona particolarmente rumorosa durante il tracciamento. Per questa situazione è stato implementato il controllo dell'Harmonic Ratio.

La seconda condizione di stop è quella in cui l'algoritmo selezioni un punto che si trova su di una ridge già tracciata. In questo caso è semplicemente necessario controllare attraverso l'immagine ausiliaria I' se il punto era già stato selezionato.

L'ultimo check da implementare, ovvero quello del riconoscimento di una terminazione, è il seguente: per ogni punto (i, j) selezionato

1) Estrarre una oriented window Ω centrata (i, j) di dimensioni $\lambda \times \lambda$ $(w \times l)$.
2) Binarizzare la window Ω rispetto alla sua intensità media (τ) in modo da ottenere le ridge ad 1 e le valli a 0.

$$\Omega'(l, m) = \begin{cases} 1, & \Omega(l, m) < \tau \\ 0, & altrimenti. \end{cases}$$

3) Calcolare l'RF sulla colonna $(c + \Delta c)$

$$RF(RidgeFraction) = \frac{1}{\lambda} \sum_{m=0}^{\lambda} \Omega'(c + \Delta c, m),$$

dove Δc è il valore di distanza in pixel del picco della x-signature locale rispetto al centro della window, calcolato durante la fase di tracciamento.

In caso si abbia $RF < \beta_3$ il tracciamento viene interrotto ($\beta_3 = 0.3$ è il valore suggerito).

5. Risultati ottenuti e conclusioni

5.1. Problemi riscontrati

Come già accennato, durante la fase implementativa ci sono stati diversi problemi riguardanti l'utilizzo di OpenCV che non si è rivelata la miglior soluzione per la manipolazione delle immagini. Ad esempio: l'estrazione delle oriented window, sfruttando la funzione LineIterator, (Figura 11) tralascia alcuni pixel che avrebbero dovuto essere valutati ed invece valuta più volte pixel che sono già stati selezionati. In questo caso il suo utilizzo rimane comunque giustificato poiché nonostante l'errore "concettuale" la funzione è un buon compromesso tra tempi di esecuzione e precisione dei risultati.

Figura 11: estrazione di alcune oriented window con diverse angolazioni con LineIterator

Sebbene i risultati riportati nel paper [1] riferiscono di un buon miglioramento rispetto all'algoritmo presentato da Maio e Maltoni, in particolare per quello che riguarda la gestione delle zone ad alto rumore e la presenza di pori nella pelle, il programma che è si è qui voluto implementare sulla base del nuovo algoritmo non ha prodotto gli stessi risultati. Infatti, si può notare in Figura 12 come il tracciamento non sia mai molto preciso e, in presenza di minuzie, l'inseguimento della ridge termina prematuramente. Una delle cause maggiori è, nella fase di tracciamento, la formula per calcolare la larghezza λ della ridge attuale che viene sovrastimata. Ciò si ripercuote sullo step successivo nel momento in cui viene stimata la variazione di angolo di tracciamento θ_c che anch'essa risulterà sovrastimata, portando la ricerca del nuovo punto al di fuori della ridge tracciata.

Figura 12: estrazione delle ridges da un'impronta digitale generata dal programma SFinGe

5.2. Conclusioni e sviluppi futuri

In definitiva l'implementazione eseguita sull'algoritmo non è ancora in grado di seguire correttamente le linee delle creste e quindi di individuare le biforcazioni e le terminazioni. Il suggerimento che si può dare per uno sviluppo futuro è quello di implementare l'algoritmo con librerie che possano manipolare meglio le immagini. Inoltre, è sicuramente necessaria un'azione di post-processing dell'immagine estratta dove poter unire le terminazioni di stringhe che si "affacciano" tra loro in modo da creare un'unica stringa continua. Ciò permette di evitare la lettura errata di terminazioni di stringa spuri.

Infine, durante lo sviluppo ci si è posti l'interrogativo se tentare un approccio completamente diverso al problema: ad esempio, cercando di sviluppare un'applicazione di riconoscimento delle minuzie basata su algoritmi di intelligenza artificiale che, in parte, non soffrirebbe del problema del thinning degli algoritmi procedurali.

6. Bibliografia

[1] D. Arpit and A. Namboodiri, *"Fingerprint feature extraction from gray scale images by ridge tracing"*, 2011 International Joint Conference on Biometrics (IJCB), 2011, pp. 1-8, doi: 10.1109/IJCB.2011.6117533.

[2] D. Maio and D. Maltoni. *"Direct gray-scale minutiae detection in fingerprints"*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(1):27–40, Jan. 1997.

[3] L. Hong, Y. Wan, and A. K. Jain. *"Fingerprint image enhancement: Algorithm and performance evaluation"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8):777–789, Aug. 1998.

[4] D. Maltoni, D. Maio, A. Jain and S. Prabhakar. *"Handbook of Fingerprint Recognition"*, London: Springer London, 2009.

[5] S. Paolucci, *Progettazione e Sviluppo di un'applicazione di supporto alla dattiloscopia.* Università Politecnica delle Marche, 2019.