UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA

IDENTIFICAZIONE DEI PARAMETRI RETINICI PER L'AUTENTICAZIONE UTENTE

Relatore: Chiar.mo Prof. Carlo Ferrari

Candidato: Alberto Baster

ANNO ACCADEMICO 2009–10

Alla mia famiglia

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

In questa tesi si è voluto studiare e implementare una soluzione all'avanguardia per quanto riguarda il riconoscimento biometrico. Fino ad ora sono stati valutati dei metodi abbastanza veloci in fase di scannarizzazione del soggetto ed elaborazione del modello in esame. Essi risultano essere abbastanza affidabili; tuttavia alcuni sono considerati poco sicuri (impronte digitali o connotati del viso) nella fase di autenticazione dell'utente. Questo perchè tramite chirurgia è possibile modificare fisicamente alcune caratteristiche dell'individuo. L'identificazione retinica è un argomento tutt'altro che nuovo, ma nonostante ciò le prime applicazioni pratiche sono state presentate poco tempo fa. Sicuramente ci sono aspetti negativi derivanti dall'utilizzo di questa tecnica; l'utente infatti preferirebbe interagire con strumenti che non debbano stare a contatto con i propri occhi. Il paper cui si ha fatto riferimento è intitolato 'A Novel Retinal Identification System' ed è stato pubblicato il 21 Febbraio 2008. Il sistema illustrato in tale paper è composto da tre moduli principali che includono la segmentazione dei vasi sanguigni, la generazione di caratteristiche e il pattern matching finale. La ricerca in questa direzione è esplosa questi ultimi anni e sicuramente in futuro ulteriori modelli più efficienti e sicuri verranno a galla.

Capitolo 1

Introduzione

Per migliaia di anni gli uomini hanno istintivamente utilizzato alcune caratteristiche fisiche (come il volto, la voce, il portamento ecc.) per riconoscersi gli uni con gli altri. Verso la fine del XIX secolo, questa idea di partenza fu ulteriormente sviluppata grazie alla scoperta (dovuta agli studi di F. Galton e E. Henry) del carattere distintivo delle impronte digitali, ovverosia del fatto che queste individuano biunivocamente una persona. Subito dopo questa scoperta le polizie di tutto il mondo cominciarono ad acquisire e memorizzare in appositi archivi le impronte di criminali, detenuti e sospetti. Inizialmente le impronte erano registrate su supporto cartaceo, inchiostrando i polpastrelli dei soggetti in questione e realizzando il timbro dell'impronta. In seguito a questa adozione, le forze di intelligence e di pubblica sicurezza perfezionarono le loro tecniche per il rilievo, sulle scene del crimine, delle impronte digitali lasciate dai protagonisti di azioni delittuose. In questi anni, la polizia comincia a fare sempre più affidamento su tecniche di indagine scientifiche, che si affiancano e quelle tradizionali (logica deduttiva) nelle investigazioni. Segni evidenti di questo nuovo approccio scientifico nel condurre le indagini si riscontrano anche in alcuni famosi personaggi della letteratura poliziesca (per tutti, Sherlock Holmes). La scienza biometrica comincia, quindi, a essere impiegata nelle attività giudiziarie e anticrimine, così come in applicazioni inerenti la sicurezza di un numero sempre crescente di persone. Oggi, in piena era digitale, un numero elevatissimo di persone utilizza tecniche di riconoscimento biometrico, non solo nel campo della giustizia, ma anche in applicazioni civili

e militari. Le previsioni di alcuni analisti di mercato affermano che, entro il 2010, la maggior parte degli abitanti della Terra avrà a che fare, episodicamente o in maniera continua, con le tecniche di riconoscimento biometrico.

1.1 Definizioni generali

Le caratteristiche biometriche si possono dividere principalmente in due categorie in base alla loro natura: fisiologiche o comportamentali. Alla prima categoria appartengono attualmente:

- retina/iride;
- mano;
- viso;
- impronte digitali.

Nella seconda categoria appartengono:

- grafia;
- stile di battitura;
- voce.

Al fine di decidere quali siano le caratteristiche migliori che un individuo possa presentare è necessario prendere in considerazione le seguenti proprietà:

- universalità: la caratteristica biometrica da valutare deve essere posseduta da tutti i membri della popolazione;
- unicità: ogni firma biometrica deve differire da quella di ogni altro membro della popolazione;
- invarianza: la firma biometrica non deve variare nelle diverse condizioni di rilevamento e nel tempo;

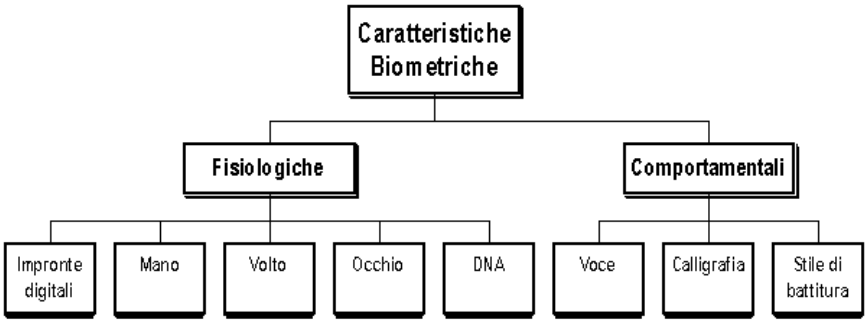


Figura 1.1: Riconoscimenti biometrici

- misurabilità: le caratteristiche devono poter essere misurabili quantitativamente e facilmente ottenibili;
- performance: si riferisce all' accuratezza di riconoscimento raggiungibile e alle risorse necessarie per il conseguimento;
- accettazione: in riferimento al grado di accettazione del sistema di rilevazione biometrica da parte dei membri della popolazione;

1.2 Il riconoscimento retinico

Il riconoscimento biometrico della retina si basa sull' unicità del suo schema vascolare. Tal scoperta risale nel 1935 quando due oftalmologi, Drs. Carleton Simon e Isodore Goldstein scoprirono, mentre stavano studiando alcune malattie dell'occhio, l'unicità del sistema vascolare di ciascun individuo. Successivamente pubblicarono un articolo nell'utilizzo delle figure retiniche per identificare persone in base al loro vessel pattern. Più tardi, nel 1950, le loro conclusioni furono sostenute dal Dr. Paul Tower dallo studio di gemelli identici. Inizialmente pensò che due gemelli dovessere avere probabilmente una simile struttura del sistema vascolare. Gli studi di Tower mostrarono che di tutti i fattori comparati tra i soggetti, i vessel retinici mostrano meno similarità nello scheletro rispetto agli altri parametri.

La retina è localizzata all'interno dell'occhio, nella sua parte posteriore. Uno scanner speciale illumina la retina, attraverso la pupilla, con una luce nell'infrarosso (IR) e memorizza le informazioni sulla riflessione tramite il contrasto rilevato.

La scansione della retina viene considerata un'eccellente e accurata tecnica di identificazione personale; grazie alla sua invulnerabilità è un sistema molto efficace nei casi in cui è richiesta un'assoluta sicurezza nel controllo degli accessi. La tecnologia non è di facile impiego e richiede sia personale esperto, sia la partecipazione dell'individuo da identificare. Viene considerato un metodo piuttosto invasivo, poichè di solito le persone preferiscono evitare un dispositivo che interagisca con i loro occhi, in quanto lo percepiscono come potenzialmente pericoloso. Ciò rappresenterà un limite all'impiego finchè non si riuscirà a realizzare una scansione della retina in maniera più friendly.

Il riconoscimento biometrico della retina funziona in maniera soddisfacente sia in modalità verifica (autenticazione), sia in modalità identificazione. Questa tecnica, rispetto alle precedenti è piuttosto costosa.

1.3 Obiettivo della tesi e software sviluppato

Scopo di questa tesi è creare un sistema per identificare la persona in base al suo pattern retinico.

Su disco verranno memorizzate codifiche in binario delle immagini JPEG analizzate precedentemente. Al momento dell'identificazione della persona, verrà estratto un modello dell'immagine JPEG posta in esame (scansionata) e successivamente convertita in stringa binaria. Il matching avverrà tra stringhe. Se le parole coincidono per un valore percentuale superiore ad una soglia fissata l'utente risulta autenticato. Il tutto può essere riassunto come riportato in figura sottostante.

1.4 La retina

La retina è la membrana più interna dell' occhio.

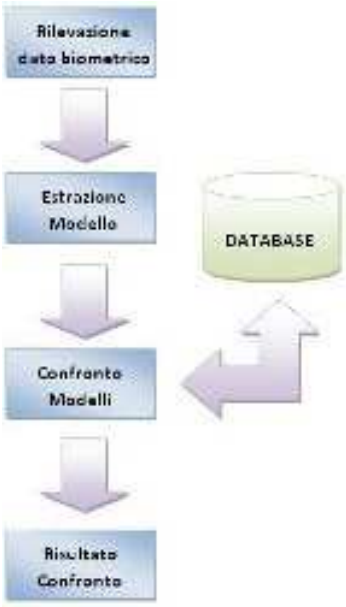
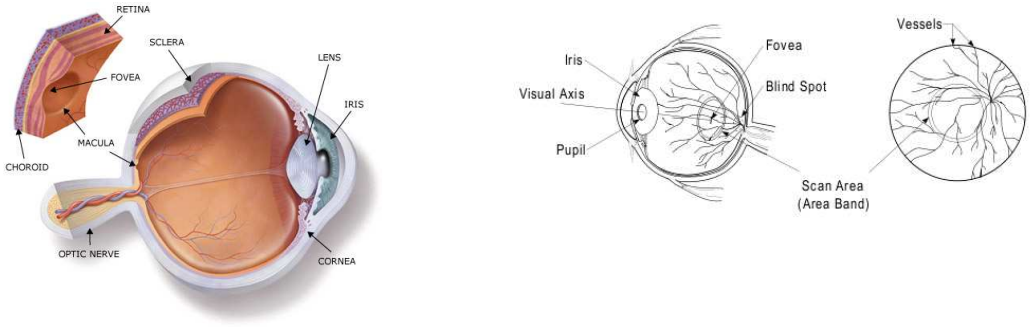


Figura 1.2: Schema riconoscimento biometrico

Con le sue cellule sensibili alle radiazioni luminose (fotorecettori), invia al cervello, attraverso il nervo ottico, le informazioni da interpretare. Tra le cellule che compongono la retina si devono ricordare: i coni, responsabili della visione a colori ma sensibili solo a luci piuttosto intense, e i bastoncelli, che sono particolarmente sensibili a basse intensità di luce ma non ai colori. I coni si suddividono in tre differenti tipologie responsabili della visione dei tre colori primari rosso, verde e blu (ovvero Red Green Blue). Pertanto i coni operano soprattutto in condizione di luce piena, mentre i bastoncelli permettono la visione anche quando la luce è scarsa.

Un fatto abbastanza curioso è la direzione della retina: i fotorecettori sono rivolti verso l'interno dell'occhio e non verso l'esterno, questo per evitare effetti di riflessione interna all'occhio stesso della luce che genererebbe riverberi nell'immagine percepita. Questo significa che la luce, prima di raggiungere un fotorecettore, deve attraversare tutti gli strati di cellule retiniche. La membrana presente posta dietro alla retina è molto ricca di melanina: ciò le permette di assorbire la luce incidente evitando ulteriormente proprio i fenomeni di riflessione.

I coni sono presenti maggiormente in una zona centrale della retina detta fovea.



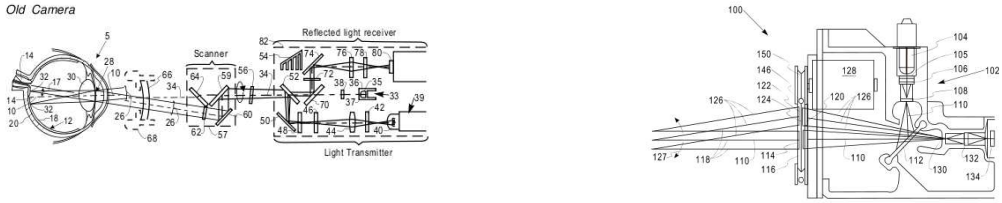
(a) Sezione immagine retinica (b) Vista frontale e laterale retina

Figura 1.3: Immagini retiniche

Questa differente densità di fotorecettori è responsabile della visione nitida nel punto di fissazione e della visione sfumata e poco definita nella zona periferica del campo visivo.

Le cellule neurali presenti nella retina sono stratificate, a partire dai fotorecettori, in: cellule orizzontali, bipolari, amacrine e ganglionari. Gli assoni di queste ultime formano il nervo ottico che si dipana a partire da una zona particolare detta papilla ottica, un'area in cui mancano i fotorecettori. Per ogni occhio infatti esiste un punto in cui non si ha visione (la cosiddetta 'macchia cieca', o in gergo oculistico, Scotoma Fisiologico). Le cellule orizzontali sono, invece, responsabili della comunicazione fra cellule dello stesso strato.

In generale a ogni cellula gangliare fanno capo più fotorecettori; nel caso dei fotorecettori presenti nella fovea si ha una cellula gangliare ogni 1-5 coni o bastoncelli; nel caso dei fotorecettori a cui è dovuta la visione periferica si hanno molti più coni e bastoncelli per ogni cellula gangliare. Quindi, in questo caso l'informazione visiva è frutto di una mediazione di svariate attivazioni di molti fotorecettori.



(a) Old camera (b) New camera

Figura 1.4: Funzionamento interno scanner retinici

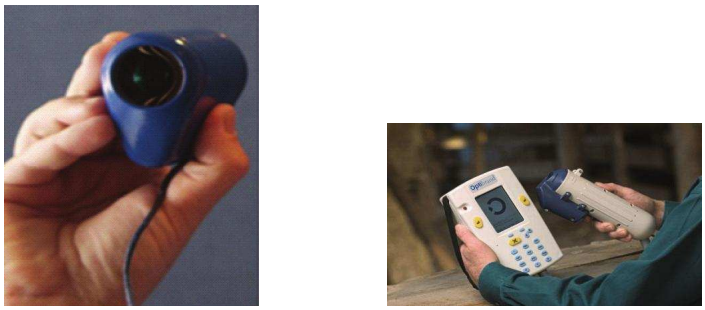
1.5 Rivelatori retinici

Gli oculisti inizialmente usavano strumenti chiamati retinoscopi. Essi emanavano una luce sorgente che veniva indirizzata alla retina del soggetto. Il macchinario quindi riusciva a intrappolare la luce riflessa. La luce che proviene dal retinoscopo è un fascio parallelo così che le lenti dell'occhio possono localizzarlo come un punto nella retina. Essa riflette della luce attraverso le lenti. L'angolo con cui il fascio entra nella retina è lo stesso dell'angolo uscente; questo processo viene chiamato retro-riflessione.

La principale compagnia per la ricerca/sviluppo e produzione di scanner di retina fu la EyeDenfity la quale venne creata nel 1976. I primi tipi di dispositivi furono chiamati scanner di fundus ed erano molto costosi e difficili da operare. Altro problema di tali dispositivi era dato dalla difficoltà di rendere l'utente comodo durante l'acquisizione. Nel 1981 nacque il primo prototito di scanner retinico nel quale veniva utilizzata la radiazione infrarossa per illuminare i vasi sanguigni. L'adozione di queste onde elettromagnetiche è dovuta al fatto che i vasi sanguigni possono assorbire la luce infrarossa più velocemente del resto del tessuto retinico.

Questo strumento, visibile in figura, utilizzava uno specchio rotante. Una sorgente di raggi infrarossi fornisce il fascio di radiazione per scannarizzare il fundus. Un diodo ad emissione di luce produce la radiazione che passa attravereso un filtro spaziale, ed è riflessa dalle lenti. Un filtro di radiazione infrarossa lascia passare solamente elementi con una prefissata lunghezza d'onda.

Le camere ad infrarosso al giorno d'oggi usano una tecnologia differente come mostrata in figura. Il design è molto più semplice. Essa include un disco di scannarizzazione rotante che include lenti multi focali, uno scanner ottico e un codificatore di posizione.



(a) Scanner umano (b) Scanner animale

Figura 1.5: Esempi scanner retinici

Per ottenere l'immagine, la luce infrarossa riflessa viene acquisita ed elaborata dal dispositivo. Lo scanner utilizza in sistema ottico, composto da lenti. Successive ricerche hanno creato dispositivi più semplici, permettendo una riduzione dei costi. L'ultimo scanner fu il ICAM 2001 progettato dalla EyeDentify. Questo prodotto fu tenuto fuori dal mercato a causa degli elevati prezzi. Compagnie come Retina Systems Inc. stanno lavorando per rendere questi dispositivi più user friendly, semplici ed economici.

I blocchi fondamentali che compongono i moderni scanner sono:

- Rivelatore/preamplificatore: Il rivelatore di immagini riceve la luce raccolta dalla camera ad infrarosso. Essa è convertita in tensione da un amplificatore operazione a basso rumore configurato come tran-impedenza. Con una attenta selezione dell'amplificatore operazionale, le componenti primarie del rumore sono effetto delle sole resistenze di feedback. Un secondo amplificatore porterà la tensione ad un valore sufficiente a pilotare un processore.

- Conversione A/D: il segnale grezzo analogico derivato dal blocco precedente può abbracciare almeno due gradi di ampiezza a causa della dimensione della

pupilla incontrata durante la normale operazione di acquisizione retinica. Questo richiede almeno 16 bit di risoluzione.

- Processore di contrasto: la funzione in questo è di ridurre il segnale grezzo in una principale informazione di contrasto. Esso può essere fatto tramite harware o software. Questa operazione rimuove il contenuto variabile o ridondante della forma d'onda acquisita e lascia ovviamente una informazione sufficiente a produrre un identificativo unico dell'occhio.

La variabilità dell'immagine acquisita varia molto poco sotto condizioni corrette. Casi in cui questa instabilità può cresce e produrre quindi errori sono:

- Mancanza di sostenimento del soggetto.
- Condizioni di fascio luminoso fuori dalla portata dello scanner.
- Incorretta distanza oculare dalle lenti della camera.
- Insufficiente dimensione della pupilla.
- Ostruzione o distorsione del collegamento ottico dovuto a:
  1. Camera sporca.
  2. Contatti con le lenti.
  3. Il soggetto non si presta a rimuovere occhiali da vista.
- Ambiente immerso in interferenze luminose.

1.6 Misurazione delle performance

Ci sono essenzialmente tre parametri che permettono di valutare un sistema biometrico e sono: dimensione, velocità e accuratezza. Le dimensioni del modello che si estrae dall'impronta biometrica, intese come dimensioni in byte, hanno significato in relazione al supporto ove verrà memorizzato; tale dato assume importanza soprattutto se si prevede l'utilizzo di dispositivi a memoria limitata come ad esempio le smartcard.

La velocità del sistema è ovviamente una discriminante molto importante in quanto, se il processo di autenticazione impiegasse troppo tempo per ottenere un risultato, diventerebbe inutilizzabile per scopi pratici (si pensi ad esempio all'identificazione dei dipendenti all' ingresso di un' azienda) . L' accuratezza risulta un parametro critico da determinare a causa dell'approccio probabilistico adottato dal sistema e va commisurata al rischio cui si incorre in caso di falsa accettazione. I tipi di errore cui un sistema biometrico può incorrere sono essenzialmente due:

- falsa accettazione (falso positivo): un utente non autorizzato viene autenticato dal sistema perchè la sua 'impronta' risulta abbastanza simile ad un modello precedentemente archiviato. L'indice associato a tale tipo di errore prende il nome di FAR (False Acceptance Rate) ed è misurato come rapporto tra il numero di accessi effettuato da utenti non autorizzati ed il numero di accessi totali;

- falso rigetto (falso negativo) : un utente autorizzato viene rifiutato dal sistema perchè la sua 'impronta' non risulta abbastanza simile al modello con cui è stata comparata. L'indice associato a tale tipo di errore prende il nome di FRR (False Reject Rate) ed è misurato come rapporto tra il numero di accessi rifiutati, da parte di utenti che ne detengono invece il relativo diritto di accesso, ed il numero di accessi totali.

FAR e FRR sono due grandezze strettamente correlate dalla seguente proprietà: al diminuire dell'una cresce l'altra . Ogni sistema biometrico offre la possibilità di regolare il rapporto FRR/FAR e quindi di aumentare o diminuire la sensibilità del sistema.

Definiamo la variabile t come il grado di tolleranza del sistema che serve a definire la bontà del sistema in termini di sicurezza. Con un basso grado di tolleranza si ha un numero elevato di false accettazioni, mentre con un alto grado di tolleranza si ha un numero elevato di falsi rifiuti.

Una volta definito t costruiamo le funzione FAR(t) (monotona non crescente) e FRR(t) (monotona non decrescente) tramite le quali è possibile calcolare ERR (Equal Error Rate) che rappresenta l'errore intrinseco del sistema.

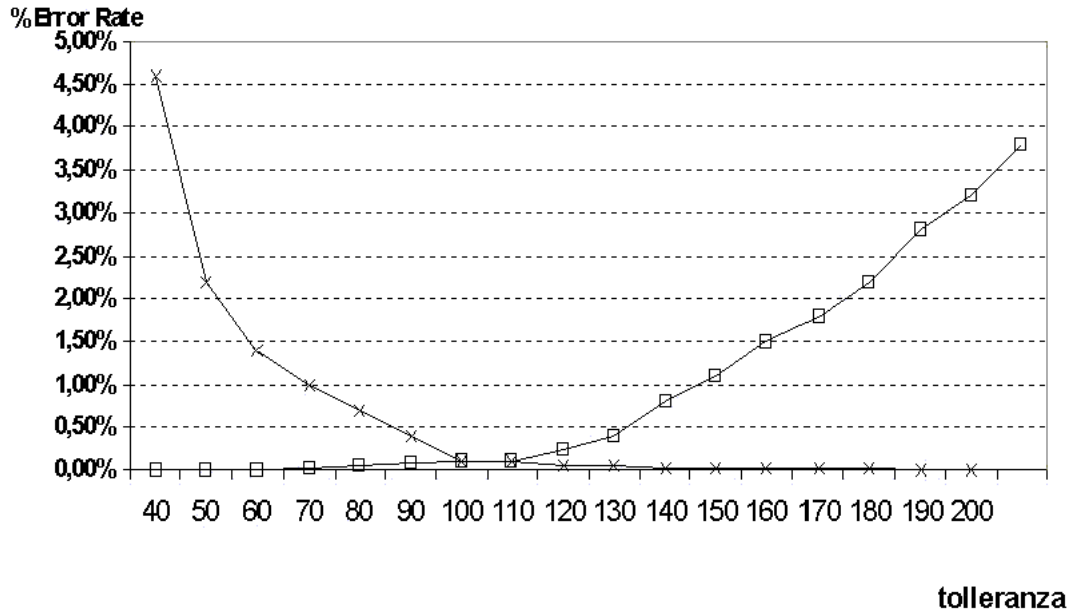


Figura 1.6: errore-tolleranza

1.7 Informazioni biometriche e Giurisprudenza

Le caratteristiche biometriche di un individuo, intese come caratteristica fisica o comportamentale, non trovano una definizione all'interno del mondo giuridico (allo stesso modo non si trovano regolamentazioni precise per le tecnologie che ne fanno uso) ma si fa riferimento a ciò che è proposto dalla sfera scientifica. Parlando a livello generale, l'elemento biometrico, non trova quindi una normativa che lo tuteli in senso stretto, ma si fa riferimento ai parametri costituzionali della salute ed integrità fisica (art. 32 Cost.). In tale prospettiva si possono citare i seguenti articoli:

- art. 13 della Carta Costituzionale che garantisce l'inviolabilità della libertà personale;
- art. 33 della Legge n.833 del 23 dicembre 1978 che esclude la possibilità di accertamenti e di trattamenti sanitari contro la volontà del paziente e non ricorrono i presupposti dello stato di necessità;

- art. 5 del Codice Civile che impedisce gli atti dispositivi del proprio corpo che incidano sull'integrità fisica.

In mancanza di una legale definizione del dato biometrico si ritiene di adottare il concetto espresso dal Gruppo dei Garanti Europei (istituito sulla base dell' art. 29 della direttiva 95/46/CE sulla protezione dei dati personali, e pertanto noto come Working Party Art. 29) nel documento adottato il 1 agosto 2003, laddove si dice che [. . .] i dati biometrici possono sempre essere considerati come informazioni concernenti una persona fisica in quanto sono dati che, per la loro stessa natura, forniscono indicazioni su una determinata persona. Le problematiche relative all'uso di tecnologie basate su sistemi biometrici, sono state dibattute in diversi sedi i cui riferimenti principali sono:

- Unione Europea: il cui Gruppo dei Garanti Europei ha prodotto un documento [17] diventato punto di riferimento in materia;
- Consiglio d'Europa: che si è interessato principalmente dei dati personali trattati tramite smart card [13];
- OECD (Organisation for Economic Co-operation and Development): che ha analizzato le caratteristiche dei sistemi biometrici e le loro relazioni rispetto alla sicurezza e alla privacy [14];
- ICAO (Internation Civil Aviation Organization): che ha proposto l'utilizzo di caratteristiche biometriche nei passaporti e in altri documenti di viaggio leggibili automaticamente al fine di velocizzare il transito dei passeggeri attraverso i controlli nelle aree aeroportuali a fini di sicurezza.

Sono poi da tenere in considerazione i principi sanciti dal codice della privacy dove all'art. 4 lettera b) del D.Lgs. n. 196/2003 definisce il dato personale come qualunque informazione relativa a persona fisica, persona giuridica, ente od associazione, identificati o identificabili, anche indirettamente, mediante riferimento a qualsiasi altra informazione, ivi compreso un numero di identificazione personale. In funzione di tale definizione, è evidente come il trattamento di dati biometrici sia, nella fattispecie, un trattamento di dati personali anche in forma di template, in quanto è sempre possibile

considerarlo come un'informazione relativa ad una persona fisica identificata o identificabile anche attraverso uno o più elementi specifici caratteristici della sua identità fisica. Ai dati biometrici dunque si applicano integralmente i principi del D. Lgs. n. 196/2003 in materia di protezione dei dati personali, fin dalla fase di enrollment. Il trattamento dei dati biometrici può dunque essere considerato lecito solo se tutte le procedure utilizzate, a partire dall'isrizione, vengono effettuate conformemente alle disposizioni di legge. Queste osservazioni sono importanti poichè giustificano da un lato la convenienza di avere i template su un server sicuro e dall'altro su un supporto rimovibile in possesso dell'utente.

Capitolo 2

Modelli di identificazione retinica

In questo capitolo viene presentato inizialmente il modello analizzato e implementato per l'estrazione della struttura dei blood vessel (vasi sanguigni), oltre che dare una breve descrizione di ulteriori template che potrebbero essere presi in esame per risolvere il medesimo compito; futuri progetti potrebbero riguardare l'implementazione di altri modelli.

2.1 Modello basato sulla localizzazione del disco ottico

L'immagine viene acquisita, segmentata in scala di grigi (convertita da rgb a 8 bit). Come si può notare in figura, nell'immagine retinica di partenza è presente una parte più luminisa (disco ottico) che si presenta come una zona chiara di forma approssimativamente circolare situata intorno alla terminazione del nervo ottico. La presenza del disco ottico interferisce con l'estrazione dei vasi; infatti, a causa del forte contrasto tra il disco e i pixel circostanti, il bordo del disco viene generalmente scambiato

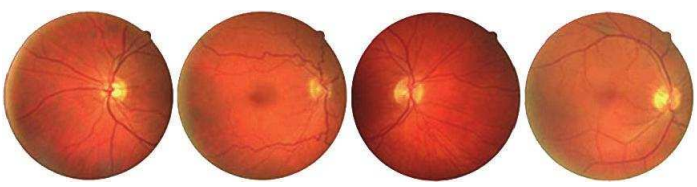
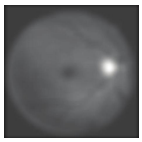


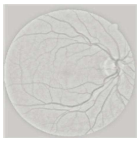
Figura 2.1: Retine umane

(a) Immagine grayscale sorgente

(b) Template utilizzato

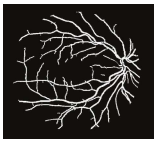


(c) Immagine correlata

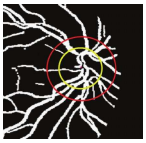


(d) Divisione pixel per pixel

Figura 2.2: Eliminazione disco ottico



(a) Immagine binaria



(b) Anello utilizzato per il filtraggio

(c) Immagine filtrata risultante

Figura 2.3: Filtraggio immagine binaria

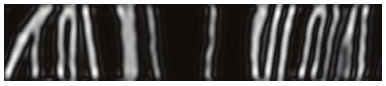
per una vena. Viene quindi eliminato correlando la sorgente con un template di disco ottico preso come modello. Il passo successivo sarà la divisione di pixel per pixel tra la sorgente e l'immagine filtrata.

Si effettua successivamente la conversione da gray scale in binaria. Quindi i possibili valori associati ad ogni pixel risultano essere esclusicamente bianco (0) e nero (1). L'immagine viene ulteriormente filtrata con un anello in modo tale da prelevare solo i vasi contenuti in esso. Il centro della circonferenza dovrà avvicinarsi al punto centrale del disco ottico. Putroppo sono possibili errori che verranno meglio analizzati nel prossimo capitolo. Viene successivamente utilizzata una conversione da coordinate cartesiani a polari.

Figura 2.4: Immagine polare

Per separare le strutture in tre grandezze differenti (grandi, medie e piccole) viene applicata una trasformata wavelet che mi permette di visualizzare l'immagine in differenti scale (scala 1 2 e 3). Se visualizzassimo figure in gradazione differente,

(a) Trasformata wavelet scala 3



(b) Trasformata wavelet scala 2

(c) Trasformata wavelet scala 1

Figura 2.5: Ulteriori elaborazioni

otterremmo delle immagini sfuocate come riportato in figura. Ad una scala maggiore corrisponde sfuocatura maggiore. Il concetto è identico nel caso in cui visualizzo l'immagine a risoluzioni differenti. Per estrarre vasi di dimensione maggiore, sogliamo coefficienti ottenuti dalla trasformata wavelet in terza scala. Per estrarre vasi medi, rimuoviamo quelli grandi dall'immagine polare e ripetiamo la stessa procedura sui coefficienti residui della trasformata wavelet calcolata in scala due. Infine, rimuoviamo vessel medi e grandi dall'immagine polare per ottenere vasi piccoli. In questo modo riesco a ottenere tre recipienti di alberi di vasi sanguigni di dimensione piccoli, medi e grandi.

(a) Vasi grandi

(b) Vasi medi

(c) Vasi piccoli

Figura 2.6: Estrazione vessel di dimensioni differenti

L'immagine convertita precedentemente in binaria mi permette di elaborare la figura d'ora in poi in maniera più semplice. Viene quindi calcolato lo skeleton.

Esso mantiene l'aspetto originario della sorgente, ma minimizza il numero di pixel considerati (possiamo quindi affermare che è una semplificazione dell'immagine di partenza). Viene riportato un esempio.

Otteniamo in questo modo tre icone che contengono esclusivamente figure curve.

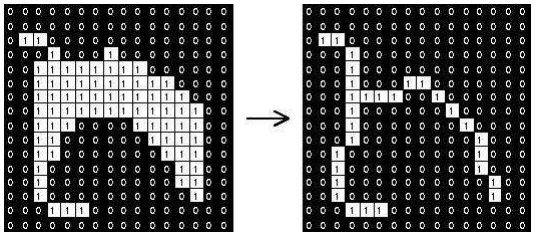


Figura 2.7: Skeleton immagine binaria

Il prossimo passo consiste nel trasformare ciascuna curva in un polinomio di primo grado(una retta).

Figura 2.8: Rappresentazione tramite rette

Questa figura geometrica dovà tenere in considerazione di tutti i valori dei pixel contenuti nella ramificazione i-esima. La pendenza della retta risultante e la sua disposizione nell'asse di coordinate polari verrà descritto da una stringa appartenente ad un alfabeto dato. Il matching tra retine avverrà tra parole. Si riporta schema riassuntivo utilizzato.

Da notare che la stringa ottenuta dalla codifica dell'immagine è imprecisa in quanto dipende dalla posizione dell'i-esima ramificazione nelle sue coordinate polari; infatti ogni volta che viene valutato il centro del disco, non è detto che sia sempre identico. Localizzazioni diverse su stessi individui dipendono da:

- Illuminazioni differenti in fase di acquisizioni successive.
- Presenza di macchie o altro derivanti da retinopatie.

Il matching deve essere svolto quindi su stringhe imprecise. Da qui nasce l'esigenza di utilizzare algoritmi di ricerca approssimati, i quali sono argomentati nel prossimo capitolo.

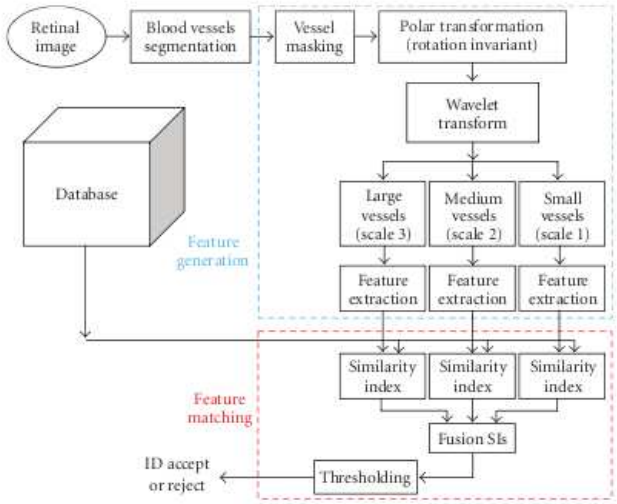


FIGURE 4: Overview of the proposed retinal identification system.

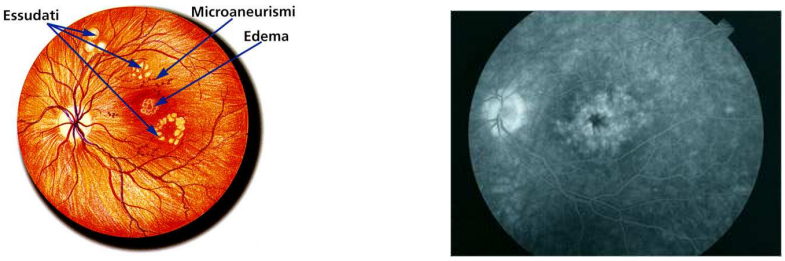
Figura 2.9: Schema utilizzato

2.2 Metodo basato su punti caratteristici

Un ulteriore modello che poteva essere sviluppato, anzichè considerare l'immagine a partire dal disco ottico, è quello in cui si fa un matching tra i punti di incrocio e biforcazione. In questo modo non è più necessaria la ricerca del disco ottico, più difficile da individuare su retine malate (per esempio di diabete) se si utilizzano tecniche di localizzazione del disco come trasformata di hough o basandosi su zone più chiare. In questo caso, come visibile in figura, sono presenti delle chiazze più illuminate (edema). Esistono due tipi di retinopatia diabetica:

- Forma non proliferante. I vasi alterati presentano zone di indebolimento, con dilatazione della parete (microaneurismi) e possono sanguinare producendo emorragie retiniche. Successivamente possono crearsi edema e/o ischemia.

  L'edema si verifica quando dalle pareti alterate dei capillari trasuda del liquido. Questo fluido provoca un rigonfiamento della retina (edema) o l'accumulo di grassi e proteine (essudati duri).



(a) Edema e microaneurismi (b) Edema maculare

Figura 2.10: Malattie retina

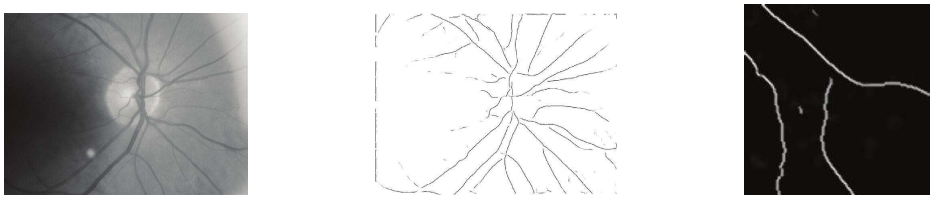
L'ischemia è il risultato dell'occlusione dei vasi capillari; la retina, ricevendo sangue in quantità insufficiente, non riesce a funzionare correttamente. Ciò favorisce il passaggio alla forma proliferante.

- Forma proliferante. Quando i capillari retinici occlusi sono numerosi, compaiono ampie zone di sofferenza retinica (aree ischemiche ed essudati molli o cotton-wool spots). Queste zone di retina sofferente, nel tentativo di supplire alla ridotta ossigenazione, reagiscono stimolando la crescita di nuovi vasi sanguigni. Sfortunatamente questi neovasi sono anomali perchè hanno una parete molto fragile e crescono a caso sulla superficie della retina. Essi sanguinano facilmente, dando luogo a emorragie vitreali, e portano alla formazione di tessuto cicatriziale, il quale, contraendosi progressivamente, può provocare il raggrinzimento e/o il distacco della retina.

Il pattern biometrico può essere così memorizzato su disco considerando solamente punti caratteristici della retina. Il tutto può essere riassunto nel seguente modo:

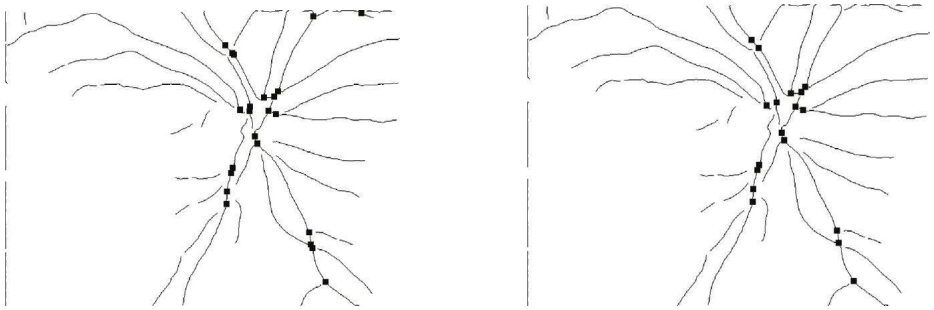
dall'immagine di partenza si estrae una sorta di skeleton iniziale che contiene l'informazione di tutti i punti di ramificazione. Come è possibile notare, sono presenti delle discontinuità nei punti di incrocio e nelle biforcazioni. Questo può essere causato dalla perdita di luminosità che si incontra in alcune regioni dell'immagine. Per ovviare a questo problema, deve essere fatta un'analisi su questi punti critici, e in caso di discontinuità, effettuare un collegamento tra pixel.

Per risolvere il problema delle interruzzioni, si può procedere per step:



(a) Immagine iniziale (b) Skeleton (c) Discontinuitá

Figura 2.11: Estrazione di livelli



(a) Immagine iniziale (b) Immagine filtrata

Figura 2.12: Punti di biforcazione spuri filtrati

- Etichettamento dei segmenti dei vasi sanguigni: viene fatto uno scanning dell'intera immagine alla ricerca di pixel non nulli (valore di soglia superiore ad un certo valore). Se viene trovato un pixel che non ha vicini, l'algoritmo continua.

- Unire punti di discontinuità: ci serviamo dell'algoritmo precedente che etichetta segmenti rappresentanti i vessel. Vengono presi in considerazione i punti finali dei segmenti a coppie e valutato lo smoothing differente tra punti differenti. Vengono accettate coppie di pixel con uno smoothing che si avvicina a 180 gradi. Una volta calcolati i punti da unire, basterà semplicemente unirli con una linea.

- Visualizzare relazioni tra crossover e biforcazioni. Le biforcazioni consentono di stimare l'albero dei vasi sanguigni dalle loro relazioni.

- Una volta calcolate le biforcazioni, è necessario applicare un filtro per eliminare punti spuri. Vengono eliminati quelli troppo vicini al bordo dell'immagine.

- Infine è possibile effettuare il matching tra punti di biforcazione calcolati prece-

dentemente. Da notare che possono imcombere errori quando la stessa immagine di un soggetto è ruotata o traslata rispetto a quella di riferimento e nel caso in cui il numero di punti di biforcazione è differente sullo stesso individuo (a causa di illuminazione differente utilizzata nelle successive scansioni retiniche). Per questi motivi deve essere valutato un modello indipendente da traslazioni e rotazioni, che permetta di effettuare delle prove su immagini imprecise. Il ST (similary tranform) potrebbe essere un'alternativa valida. Oppure si potrebbe utilizzare la trasformata SIFT(Scale-invariant feature transform), utilizzata per descrivere caratteristiche locali di una immagine; questa permette di estrarre punti di biforcazione/incrocio dall'immagine di partenza in modo robusto rispetto a cambiamenti di scala, rumore, illuminazione e distorsioni geometriche locali.

Capitolo 3

Descrizione dettagliata modello

In questo capitolo viene analizzato in maniera approfondita il nostro modello preso in esame. Viene approfondito il nostro template descrivendo dettagliatamente passo passo le varie operazioni utilizzate per processare la nostra immagine retinica di partenza.

3.1 Scala di grigi

La prima operazione è la conversione di immagini con tonalità rbg(Red-Green-Blue) in formato jpeg verso figure da 8 bit (scala di grigi). Rgb indica che ogni punto dell'immagine è rappresentato da un livello rosso, uno verde e uno blu. Dato che ogni tonalità percepita dall'essere umano può essere rappresentata come una combinazione di questi tre colori, le immagini RGB sono immagini complete. Ogni canale ha (ad esempio) 256 livelli di intensità possibili. Questo tipo di conversione riduce le dimensioni del file eliminando le informazioni sulla tonalità dei colori. Inoltre ulteriori elaborazioni sono più semplici da eseguire avendo un'immagine semplice con un solo canale. Da notare, comunque, che la conversione in scala di grigi, comporta la perdita di parte del contenuto informativo di partenza.

```
1

3 // this method is used to filter a rgb image
```

```
public BufferedImage filter(BufferedImage src, BufferedImage dst)
    {
        if (src.getType() != BufferedImage.TYPE_INT_RGB) return src;
        int w = src.getWidth();
        int h = src.getHeight();
        dst = new BufferedImage(w, h, BufferedImage.TYPE_BYTE_GRAY);
        int[] pix = ImageTools.getIntPixels(src);
        byte[] dstpix = ImageTools.getBytePixels(dst);

        for(int y = 0; y < h; y++) {
            int off = y*w;
            for(int x = 0; x < w; x++) {
                int offx = off + x;
                dstpix[offx] = (byte)gray(pix[offx]);
            }
        }
                dst.getRaster().setDataElements(0,0,dstpix);
        return dst;
    }
```

Si riporta inoltre metodo gray

```
// this method calculate short value from int value
 short gray(int c)
    {
        int r = (c & 0xff0000) >> 16;
        int g = (c & 0xff00) >> 8;
        int b = c & 0xff;
        return (short)(0.3*r + 0.59*g + 0.11*b);
    }
```

Da notare la formula di conversione nel metodo gray per passare da tre canali ad uno da 8 bit.

3.2 Correlazione tra immagini

Nel codice implementato si utilizza la correlazione per eliminare il disco ottico (il quale è identificabile nelle immagini retiniche come una zona più luminosa situata nella

parte più a destra della figura). Come operazione successiva si va a rimuovere questa zona più illuminata, tondeggiante e fastidiosa tramite una divisione del contenuto di ciascun pixel per pixel tra l'immagine sorgente e quella filtrata ottenuta nel punto precedente. Come kernel utilizziamo un'immagine che rappresenta il nostro disco ottico, preso da un template salvato su disco. La correlazione rappresenta la misura di similitudine di due segnali come funzione di uno spostamento temporale applicato ad uno di essi. Si calcola la correlazione incrociata per mostrare di quanto y deve essere anticipato per essere reso identico ad x. Quando i due segnali coincidono, il valore di $(x \bigotimes y)$ è massimizzato, poichè quando le forme d'onda sono allineate, esse forniscono solo contributi positivi all'area. Per due segnali di energia x ed y la correlazione incrociata è definita come:

$$R_{xy}(t) = (x \otimes y)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} x^*(\tau)\, y(t+\tau)\, d\tau$$

Da notare che la correlazione incrociata dei segnali x(t) e y(t) è equivalente alla convoluzione di x *(t) e y(t) secondo la formula:

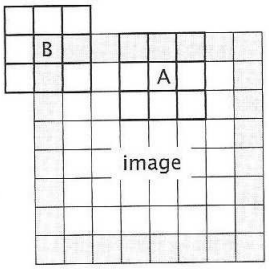
$$x \otimes y = (t \mapsto x^*(-t)) * y.$$

La correlazione incrociata soddisfa la proprietà:

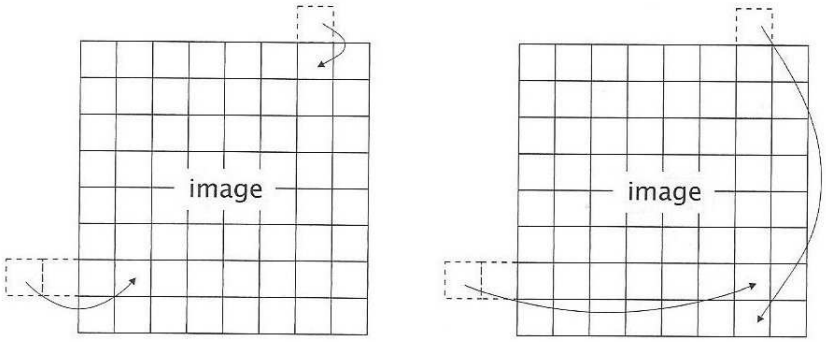
$$\mathcal{F}\{x \otimes y\} = (\mathcal{F}\{x\})^* \cdot \mathcal{F}\{y\}$$

Quindi è possibile calcolare la correlazione tra due segnali applicando la trasformata di Fourier ai due e poi eseguire il loro prodotto. Nel dominio delle frequenza la complessità computazionale risulta essere inferiore rispetto alla classica definizione di correlazione (traslazione nel tempo) che viene effettuata genericamente.

La correlazione quindi è semplicemente una somma tra elementi disposti attorno ad un elemento centrale. Se abbiamo una immagine da 8 bit, eseguendo la somma si potrebbe andare in overflow. Per ovviare a questo inconveniente è possibile fare una normalizzazione dei valori, oppure adottare una rappresentazione del valore dei pixel da 8 a 16 o 32 bit. Un secondo problema concerne la trattazione del bordo dell'imma-



(a) Esempio punti non correlabili



(b) Indice riflesso e indice circolare

Figura 3.1: Problemi e risoluzione in correlazione tra immagini

gine. Non è possibile trattarlo semplicemente perchè il kernel di riferimento non ha valori corrispodenti cui può essere moltiplicato, come riportato in figura. Per un kernel 3x3 ciascun pixel appartenente al bordo non può essere processato. Di conseguenza, con in ingresso un'immagine di grandezza MxN, si avrà come uscita un'immagine di (M-2)x(N-2) pixel. Per un kernel 5x5, in ciascun bordo due pixel non possono essere processati. In questo caso, se in ingresso ho un'immagine di grandezza MxN, in uscita otterò un'immagine di (M-4)x(N-4) pixel. In entrambi i casi otterrò quindi un'immagine più piccola.

Per ovviare a questo inconveniente, possono essere adottate delle soluzioni sviluppate nel software allegato alla tesi:

- Nessun processamento al bordo: la più semplice soluzione consiste semplicemente nell'ignorare quei pixel cui la convoluzione non è possibile. L'immagine di uscita viene inizializzata con tutti pixel a 0 (neri). In uscita otterrò quindi una figura con stesse dimensioni di quella in ingresso, ma vi sarà presente un contorno nero derivante dai pixel del bordo inizializzati precedentemente.

- Copiare pixel di input: un'altra soluzione consiste nel copiare valori di input che non sono possibili da processare nell'immagine di output.

- Troncamento dell'immagine. Le prime due soluzioni non hanno molto senso. Alcune volte è preferibile quindi troncare l'immagine e ottenere in uscita una figura di dimensioni inferiori.

- troncamento del kernel. È possibile troncare il kernel per poter processare anche i pixel del bordo. Ad esempio la matrice 3x3

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  troncata diverrà:

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  Questa tecnica tuttavia aggiunge complessità alla convoluzione.

- Indice riflesso. Dalla formula della correlazione è possibile notare che alcuni problemi vengono dati dal valore della funzione nel punto f(x-j,y-k). Una oppure entrambe le espressioni x-j e y-k andranno fuori range di possibili valori. In una immagine MxN, i valori di riga e colonna consentiti sono: [0,M-1] e [0,Y-1]. Per ciascun passo di computazione, verifichiamo se x-j corrisponde ad un pixel valido, altrimento questo verrà riflesso. La stessa cosa dovrà essere fatta per y-k.

- Indice circolare: un'altro metodo utilizzato per manipolare pixel fuori range è tramite indice circolare. In questo caso l'immagine ripete se stessa continuamente in tutte le direzioni.

Si riporta metodo in cui vengono considerati tutti gli esempi riportati.

```
// this method do convolution of BufferedImage with four different way
public float[] convolve(BufferedImage image) {

  int w = image.getWidth();
  int h = image.getHeight();
  Raster raster = image.getRaster();
  float[] result = new float[w*h];
  float[] coeff = kernel.getKernelData(null);
  int m = width/2, n = height/2;

  float sum;
  int i, j, k, x, y;
  switch (borderStrategy) {

    case REFLECTED_INDEXING:
      for (y = 0; y < h; ++y)
        for (x = 0; x < w; ++x) {
          for (sum = 0.0f, i = 0, k = -n; k <= n; ++k)
            for (j = -m; j <= m; ++j, ++i)
              sum += coeff[i] * raster.getSample(
                refIndex(x+j, w), refIndex(y+k, h), 0);
          result[y*w+x] = sum;
        }
      break;

    case CIRCULAR_INDEXING:
      for (y = 0; y < h; ++y)
        for (x = 0; x < w; ++x) {
          for (sum = 0.0f, i = 0, k = -n; k <= n; ++k)
            for (j = -m; j <= m; ++j, ++i)
              sum += coeff[i] * raster.getSample(
                circIndex(x+j, w), circIndex(y+k, h), 0);
```

```
            result[y*w+x] = sum;
34        }
      break;
36
    case COPY_BORDER_PIXELS:
38    copyBorders(raster, result);
      // fall through and let the NO_BORDER_OP
40    // case do the convolution...

42  default:
      // NO_BORDER_OP
44    for (y = n; y < h-n; ++y)
        for (x = m; x < w-m; ++x) {
46        for (sum = 0.0f, i = 0, k = -n; k <= n; ++k)
            for (j = -m; j <= m; ++j, ++i)
48          {
            sum += coeff[i] * raster.getSample(x+j, y+k, 0);
50

            }
52
          result[y*w+x] = sum;
54      }
      break;
56

  }
58
  return result;
60

}
```

3.3 Localizzazione disco ottico

Prima di proseguire nella trattazione del problema, è emersa la necessità di convertire la nostra immagine da scala di grigi a (profondità) binaria, cosa che semplifica l'analisi della stessa. Tale semplificazione deriva dal fatto che, rispetto ad una figura da 8

bit, i valori dei pixel saranno del dominio bianco/nero. Qualcuno potrebbe chiedersi per quale motivo questa conversione venga fatta ora e non prima. Questa decisione risulta obbligata dal fatto che tale trasformazione fa perdere molte informazioni necessarie per il calcolo della correlazione. Ora, al contrario, siamo arrivati ad uno step in cui possiamo eliminare le informazioni che non sono più utili. L'informazione aggiuntiva infatti complicherebbe solamente la nostra analisi e risoluzione dei problemi successivi.

Ora è necessario localizzare il disco ottico ed estrarre vasi di interesse filtrati secondo un anello di diametro opportuno. Per determinare il centro sono possibili varie soluzioni, tutte prese in considerazione nel codice riportato.

3.3.1 Metodo basato sui pixel attivi

La più semplice consiste nell'osservare l'immagine in scala di grigi privata della zona luminosa a forma di cerchio (secondo la trasformazione precedente). Si può notare come in prossimità del centro in cui dovrà essere tracciato il nostro anello ci sia una ampia concentrazione di pixel di tonalità scura. Quindi il problema è stato risolto semplicemente facendo scorrere lungo tutta l'immagine un filtro di dimensioni quadrate, calcolare la media dei pixel attivi in quella zona e stabilire quindi il punto di concentrazione massima. Da alcune prove effettuate su dataset, questo metodo molto semplice risulta essere abbastanza efficacie.

```
1

3
 // method to calculate sum of value of pixels in subimage
5 public static double sumValue(BufferedImage img) {
        Raster raster = img.getRaster();
7       double sum = 0.0;
        for (int y = 0; y < img.getHeight(); ++y)
9               for (int x = 0; x < img.getWidth(); ++x){
                        if (raster.getSample(x, y,0)== 255)
11                              continue;
```

```
13                          sum += raster.getSample(x, y, 0);

15                  }

17
        return sum ;
19  }
```

3.3.2 Trasformata di Hough

Vogliamo riconoscere il nostro disco ottico in modo differente; la soluzione più intuitiva potrebbe essere:

- Uso un modello (template).
- Il modello scorre sull'immagine e lo si confronta con l'icona sottostante; si cerca quindi di minizzare la funzione di errore definita

$$E(x,y) = \sum_{i,i} |I(y+i,x+j)T(i,j)|$$

Tuttavia questo problema è di difficile risoluzione perchè:

1. gli oggetti subiscono traslazioni
2. variazione di colore
3. rotazioni
4. cambiamenti di scala
5. acquisizioni rumorose

Una delle soluzioni proposte è la trasformata di Hough.

È una tecnica che permette il riconoscimento di configurazioni globali presenti in una immagine (segmenti, curve, forme prestabilite), e si basa sulla trasformazione di

tutti i punti costituenti una immagine, in punti di un nuovo spazio detto spazio dei parametri. La trasformata di Hough, nella sua versione tradizionale, si applica ad immagini binarie, in cui l'informazione associata ad un punto è rappresentata unicamente dalla sua posizione. Ovviamente dovrà essere applicata prima di effettuare la correlazione tra la nostra immagine di partenza e il template, in modo da ottenere una immagine binaria con una sottofigura tondeggiante più chiara (il nostro disco ottico). Essa permette di individuare forme descritte da equazioni analitiche (prima è stata introdotta per le rette, poi per i cerchi, le parabole, ecc.), ed in suo favore giocano caratteristiche come:

1. La HT è parzialmente insensibile alle occlusioni (e al rumore).

2. La HT trasforma il problema della ricerca di una curva nella più semplice ricerca di massimi.

In generale una curva piana è definita in forma analitica tramite un insieme (limitato se è semplice) di parametri. Una equazione lega i parametri alle coordinate cartesiane. Ne consegue che qualsiasi curva può essere descritta dall'equazione

$$f((x,y),(a1,a2,...,an)) = 0$$

dove (x,y) è un punto della curva nello spazio immagine e $(a_1, a_2, ..., a_n)$ è una n-upla di valori che individuano un punto nello spazio dei parametri. Di fondamentale importanza è che un punto nello spazio dei parametri individua univocamente una curva analitica.

Si riportano alcune proprietà della HT.

1. Ogni punto nello Spazio Immagine (SI) corrisponde ad una ipersuperficie (superficie generalizzata) nello Spazio dei Parametri (SP).

2. N punti nello SI appartenenti ad una stessa curva generano n superfici che si intersecano in uno stesso punto in SP.

3. Ogni punto nello SP corrisponde ad una singola istanza della curva nello SI

4. n punti nello SP appartenenti ad una stessa ipersuperficie corrispondono a n curve che si intersecano in uno stesso punto in SI.

Per esempio, se consideriamo una retta con equazione $y = mx + q$, può essere riscritta in

$$f((x,y),(m,q)) = y - mx - q = 0$$

. Fissato un punto (x_i, y_i) nello SI l'equazione

$$q = y_i - mx_i$$

descrive la curva (che rimane ancora una retta).

La circonferenza di centro (α, β) e raggio r è il luogo dei punti caratterizzati dall'equazione

$$(x-\alpha)^2 + (y-\beta)^2 = r^2$$

Se ricerchiamo cerchi di raggio noto, lo SP è bidimensionale e la curva generata da ogni punto nello SI è essa stessa un cerchio e lo spazio dei parametri è generato dalle coppie dalle coordinate del centro (x_c, y_c) Se consideriamo l'equazione parametrica, un cerchio con raggio R e centro (a,b) puo essere descritto da

$$x = a + R\cos(\theta), y = b + R\sin(\theta)$$

Nel caso in cui anche il raggio è incognito, lo spazio dei parametri è tridimensionale. Riscrivendo la circonferenza ottengo:

$$f((c,r),(c_c,r_c),\rho) = (r-r_c)^2 + (c-c_c)^2 - \rho^2 = 0$$

La curva generata in questo caso è un cono.

Da un punto di vista implementativo occorre discretizzare lo SP in celle n-dimensionali. La dimensione delle celle dipende dalla precisione minima richiesta e ogni cella corrisponde ad una istanza (quantizzata) della curva. Dato un punto di contorno nello SI si incrementa ogni cella intersecata dalla superficie generata (processo di voto).

Ogni cella dello SP misura perciò il numero di contributi al riconoscimento della curva corrispondente. Di fondamentale importanza quindi assume il processo di voto dove:

1. I voti si possono 'pesare' in base alla significatività
2. La ricerca dei massimi può essere sufficiente per stabilire la presenza della curva cercata
3. La soglia può essere fissata anche con criteri teorici: valore aspettato e rapporto S/N (rapporto segnale/rumore)
4. La soglia deve comunque rappresentare un compromesso fra il rischio di non rilevare oggetti presenti e quello di ottenere falsi positivi

Un algoritmo generico potrebbe essere il seguente:

```
1  Definisci un accumulatore A(θ,ρ) di dimensioni opportune
2  Inizializza A(θ,ρ) a 0
3  Per ogni punto dell'immagine significativo {
4      incrementa tutti i punti dell'accumulatore corrispondenti
5      }
6  Seleziona i massimi locali dell'accumulatore che corrispondono
   a punti appartenenti alla stessa curva presenti nell'immagine
```

Nel nostro caso, la trasformata richiederebbe una matrice 3D denominata $Acc(\alpha, \beta, r)$ per trovare la posizione del cerchio in quanto non è noto a priori il raggio del disco ottico. Un possibile algoritmo potrebbe essere diviso in vari step:

```
1  ritorna la lista dei parametri (α,β,r) corrispondenti
2  setta la matrice 3D Acc(α,β,r) a 0
3  per tutti i punti(pixel) (u,v) nello spazio immagine {
4      se I(u,v) e un punto di un lato
5      se punto soddisfa equazione circonferenza data
6        incrementa accumulatore di uno
7  }
```

Si riporta un estratto del codice implementato. Da notare che la matrice è 2D in quanto si presuppone di conoscere il raggio del disco da precedenti analisi.

```
1 // Method that returns the accumulator array
  public int[] process() {
3
        // for polar we need accumulator of 180degress * the longest length in the image
5       int rmax = (int)Math.sqrt(width*width + height*height);
        acc = new int[width * height];
7       for(int x=0;x<width;x++) {
                for(int y=0;y<height;y++) {
9                       acc[x*width+y] =0 ;
                }
11      }
        int x0, y0;
13      double t;
        progress=0;
15
        for(int x=0;x<width;x++) {
17              progress+=0.5;
                for(int y=0;y<height;y++) {
19
                        if ((input[y*width+x] & 0xff)== 255) {
21
                                for (int theta=0; theta<360; theta++) {
23                                      t = (theta * 3.14159265) / 180;
                                        x0 = (int)Math.round(x - r * Math.cos(t));
25                                      y0 = (int)Math.round(y - r * Math.sin(t));
                                        if(x0 < width && x0 > 0 && y0 < height && y0 > 0) {
27                                              acc[x0 + (y0 * width)] += 1;
                                        }
29                              }
                        }
31              }
        }
33
  // now normalise to 255 and put in format for a pixel array
35      int max=0;

37      // Find max acc value
```

```
        for(int x=0;x<width;x++) {
39              for(int y=0;y<height;y++) {

41                      if (acc[x + (y * width)] > max) {
                                max = acc[x + (y * width)];
43                      }
                }
45      }

47      //System.out.println("Max :" + max);

49      // Normalise all the values
        int value;
51      for(int x=0;x<width;x++) {
                for(int y=0;y<height;y++) {
53                      value = (int)(((double)acc[x + (y * width)]/(double)max)*255.0);
                        acc[x + (y * width)] = 0xff000000 | (value << 16 | value << 8 | value);
55              }
        }
57      findMaxima();

59      System.out.println("done");
        return output;
61  }


2   // this method is used to calculate max of values

4   private int[] findMaxima() {
        results = new int[accSize*3];
6       int[] output = new int[width*height];

8

        for(int x=0;x<width;x++) {
10              for(int y=0;y<height;y++) {
                        int value = (acc[x + (y * width)] & 0xff);
12

                        // if its higher than lowest value add it and then sort
```

```
14                          if (value > results[(accSize-1)*3]) {

16                              // add to bottom of array
                                results[(accSize-1)*3] = value;
18                              results[(accSize-1)*3+1] = x;
                                results[(accSize-1)*3+2] = y;
20
                                // shift up until its in right place
22                              int i = (accSize-2)*3;
                                while ((i >= 0) && (results[i+3] > results[i])) {
24                                  for(int j=0; j<3; j++) {
                                        int temp = results[i+j];
26                                      results[i+j] = results[i+3+j];
                                        results[i+3+j] = temp;
28                                  }
                                    i = i - 3;
30                                  if (i < 0) break;
                                }
32                          }
                    }
34          }

36          double ratio=(double)(width/2)/accSize;
            System.out.println("top "+accSize+" matches:");
38          for(int i=accSize-1; i>=0; i--){
                progress+=ratio;
40
                drawCircle(results[i*3], results[i*3+1], results[i*3+2]);
42          }
            return output;
44  }
```

3.3.3 Metodo basato sull'incrocio dei vessel

La localizzazione del disco ottico può essere stimata tramite la convergenza dei pixel appartenenti ai vasi sanguigni. Potrebbe essere adottata la logica fuzzy. Si tratta di un

metodo basato sul voto in cui ciascun vessel è modellato da un segmento fuzzy. Sommando i vari contributi, genero una mappa in cui ogni pixel ha un valore proporzionale alla sua forza di convergenza. Il punto con il picco di convergenza più alto è ritenuto essere il centro del disco ottico. Questa soluzione non è stata implementata.

3.3.4 Modello geometrico basato sull'albero vascolare

Questo si basa sull'estrazione della rete vascolare e sul successivo fitting dei pixel corrispondenti ai vasi principali con una parabola, il cui vertice si assume coincidente con il centro del disco ottico. Tale metodo richiede la soluzione di un problema di ottimizzazione con molti minimi locali, per risolvere il quale si intende utilizzare tecniche di ottimizzazione basate sulla swarm intelligence. Anche questo modello non è stato implementato nel codice allegato.

3.4 Estrazioni vasi sanguigni

Una volta determinato il centro del disco ottico, dobbiamo estrapolare vasi sanguigni contenuti in prossimità di un anello costruito appositamente. Si noti come un piccolo spostamento della localizzazione del centro sia causa di immagini filtrate differenti.

```
1 private static Area createRing(int index, double r1, double r2,int width, int height) {
        double x2 = cps[index].x − r2;
3       double y2 = cps[index].y − r2;
        double x1 = cps[index].x − r1;
5       double y1 = cps[index].y − r1;
        Ellipse2D.Double inner = new Ellipse2D.Double(x1, y1, 2∗r1, 2∗r1); // first
7       Ellipse2D.Double outer2 = new Ellipse2D.Double(x2, y2, 2∗r2, 2∗r2); //second
        Rectangle outer1 = new Rectangle(0,0,width,height); //rectangle
9
        Area area = new Area(outer1); //rectangle
11      area.subtract(new Area(outer2)); // subtract first circle

13      return area;
        }
```

3.5 Coordinate polari

Una prima operazione, detta normalizzazione, consiste nel rappresentare l'immagine retinica in un altro dominio (coordinate polari) perchè in questo nuovo modello è possibile effettuare una comparazione tra immagini. L'immagine scannarizzata può infatti variare a causa della distanza di acquisizione differente o per la diversa angolatura di posizionamento dello scanner. Il sistema di coordinate polari è un sistema di coordinate bidimensionale nel quale ogni punto del piano è identificato da un angolo e da una distanza da un punto fisso detto polo. Esso è in corrispondenza biunivoca con un sistema di coordinate cartesiane. Le due coordinate polari r e θ possono essere convertite nelle coordinate cartesiane x e y utilizzando funzioni seno e coseno:

$$x = r\cos\theta y = r\ \mathrm{sen}\,\theta,$$

Viceversa puo essere determinato considerando le seguenti formule: per ottenere la distanza r si usa:

$$r = \sqrt{x^2 + y^2}$$

Per ottenere il valore di nell'intervallo [0, 2) si usa:

$$\theta = \begin{cases} \arctan(\frac{y}{x}) & \text{se } x > 0 \text{ e } y \geq 0 \\ \arctan(\frac{y}{x}) + 2\pi & \text{se } x > 0 \text{ e } y < 0 \\ \arctan(\frac{y}{x}) + \pi & \text{se } x < 0 \\ \frac{\pi}{2} & \text{se } x = 0 \text{ e } y > 0 \\ \frac{3}{2}\pi & \text{se } x = 0 \text{ e } y < 0 \end{cases}$$

```
2 // method to compute parameters

4 public void computeMappingParameters() {
```

```
6       // Determine image dimensions and centre

8       imageRect =
          new Rectangle(sourceImage.getWidth(), sourceImage.getHeight());
10      cx = sourceImage.getWidth() / 2.0f;
        cy = sourceImage.getHeight() / 2.0f;
12
        // Compute r and theta increments
14
        dr = (float) (Math.sqrt(cx*cx + cy*cy) / numRings);
16      dtheta = 360.0f / numSectors;

18      // Compute lookup tables for cos and sin

20      ctheta = new float[numSectors];
        stheta = new float[numSectors];
22      double theta = 0.0, dt = dtheta*(Math.PI/180.0);
        for (int j = 0; j < numSectors; ++j, theta += dt) {
24        ctheta[j] = (float) Math.cos(theta);
          stheta[j] = (float) Math.sin(theta);
26      }

28    }

    // Create a log-polar image
2     public BufferedImage createLogPolarImage() {

4       BufferedImage logPolarImage =
          new BufferedImage(numSectors, numRings, sourceImage.getType());
6
        // Fill first ring (r = 0) with data
8
        int centralValue = sourceImage.getRGB(Math.round(cx), Math.round(cy));
10      for (int j = 0; j < numSectors; ++j) {
          logPolarImage.setRGB(j, 0, centralValue);
12      }

14      // Fill rings with r > 0
```

```
16      int x, y;
        float r = dr;
18      for (int k = 1; k < numRings; ++k, r += dr)
          for (int j = 0; j < numSectors; ++j) {
20          x = Math.round(cx + r*ctheta[j]);
            y = Math.round(cy + r*stheta[j]);
22          if (imageRect.contains(x, y))
              logPolarImage.setRGB(j, k, sourceImage.getRGB(x, y));
24          else
              logPolarImage.setRGB(j, k, 0);
26        }

28      return logPolarImage;

30    }
```

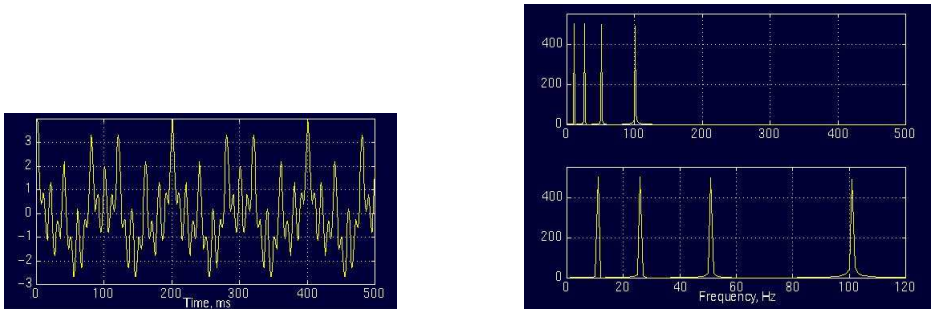
3.6 Trasformata wavelet

Diamo ora una descrizione generale della trasformata wavelet, confrontandola con la trasformata di Fourier e facendo notare le possibili applicazioni pratiche. Partiamo innanzi tutto a descrivere l'analisi tempo-frequenza; lo scopo è rappresentare la generica grandezza $x(t)$ in un dominio differente da quello naturale del tempo, in modo da evidenziare meglio alcune caratteristiche altrimenti difficilmente rilevabili.

La maggior parte dei segnali reali fisici è di tipo non stazionario, cioè con caratteristiche variabili nel tempo. Ad esempio, si può pensare ad una combinazione di sinusoidi con diversi parametri (ampiezze A_i, tempi di inizio t_i, frequenze dominanti f_i, fasi iniziali ψ_i e coefficienti di smorzamento a_i). La seguente formula rappresenta bene un segnale non stazionario.

$$x(t) = \sum_{i=0}^{N} A_i e^{-a_i(t-t_i)} cos(2\pi f_i t + \psi_i) u(t-t_i)$$

Se su questo segnale si effettua una trasformata di Fourier, si ottiene una funzione



(a) Immagine nel tempo (b) Ft segnale originale

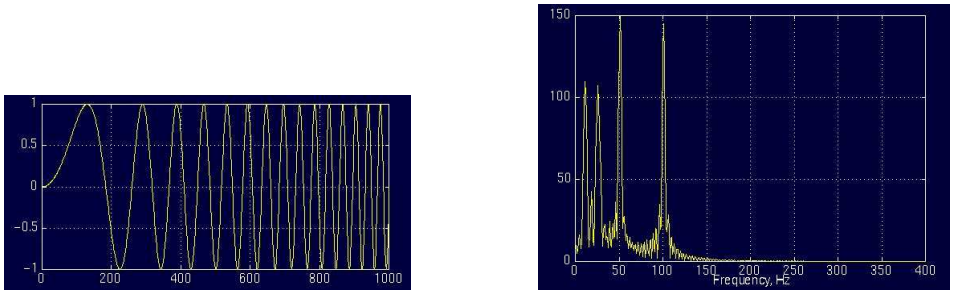
Figura 3.2: Ft segnale stazionario

$X(f)$ il cui modulo fornisce delle chiare informazioni riguardo la presenza delle componenti armoniche f_i e delle rispettive ampiezze A_i. Le informazioni relative agli altri parametri sono inglobate nella fase di $X(f)$; il problema è che tali informazioni sono così mescolate fra loro, specie se il segnale ha molte componenti, da risultare illeggibili. In altre parole, la trasformata di Fourier evidenzia la presenza delle componenti armoniche ma non permette di ricavare facilmente informazioni su quando e come tali frequenze siano effettivamente presenti. Voglio ad esempio convertire il segnale

$$x(t) = cos(2*\pi*10t) + cos(2*\pi*25t) + cos(2*\pi*50t) + cos(2*\pi*100t)$$

Esso è stazionario, perchè ha componenti in frequenza di 10, 25, 50 e 100hz. In figura soprastante è possibile vedere il segnale e la sua trasformata di Fourier. Prendiamo in considerazione ora un'ulteriore segnale. Nell'intervallo 0-300ms ha una frequenza di 100 HZ, dai 300 ai 600ms di 50 HZ, tra 600 e 600ms 25 HZ e infine tra 800 e 1000ms 10 HZ. Questo risulta essere un segnale non stazionario, e in figura è possibile vedere una sua rappresentazione grafica assieme alla sua FT. Come si può notare, la FT ha quattro picchi. Ma essi non corrispondono alle quattro frequenze. Quindi a che intervallo di tempo capitano queste frequenze? A tutti i tempi. In un segnale stazionario, tutte le componenti in frequenza, esistono per tutta la durata del segnale.

Se confrontiamo la Ft del segnale stazionario con quello non stazionario notiamo che i due spettri sono al più identici. Quindi la FT fornisce il contenuto spettrale di un segnale. Non dà informazioni dove queste componenti appaiono nel tempo.



(a) Immagine nel tempo (b) Ft segnale originale

Figura 3.3: Ft segnale non stazionario

Per i segnali non stazionari occorre quindi inserire nella trasformazione una dipendenza dal tempo; il modo più immediato consiste nel rendere locale la FT non operando su tutto il supporto del segnale $x(t)$ ma su porzioni di esso, ottenuto moltiplicandolo per una finestra che trasla nel tempo. Nasce così la trasformata di Fourier a breve termine o Short Time Fourier Transform:

$$STFT_x(\tau, f) = \int_{-\infty}^{+\infty} x(t)g^*(t-\tau)e^{-j2\pi ft}dt = < g(t-\tau)e^{j2\pi ft}, x(t) >$$

Con questa nuova trasformazione si evidenzia lo spettro del segnale all'interno della finestra temporale definita dalla funzione $g(t-\tau)$ e ottenendo quindi un'informazione sul suo contenuto armonico in un intorno dell'istante di tempo τ. Questo approccio, benchè molto semplice, ha un difetto tutt'altro che trascurabile: moltiplicare nel dominio del tempo il segnale $x(t)$ con la funzione finestra $g(t)$ equivale a effettuare la convoluzione dei loro spettri $X(f)$ e $G(f)$ nel dominio della frequenza; la STFT fornisce quindi lo spettro del segnale alterato dalla presenza della finestra. Il vero problema è che tale difetto non è eliminabile; se si vuole ridurre l'effetto di $G(f)$ occorre ridurne la banda Δf, ma ciò implica un aumento dell'ampiezza temporale Δt e quindi una diminuzione della precisione nel dominio del tempo, nel senso che due eventi separati da meno di Δt non sono più discriminabili; viceversa, se si riduce la finestra Δt necessariamente aumenta la banda Δf e quindi si ha una diminuzione della precisione in frequenza. In questo progetto ci serviamo dell'analisi multirisoluzione (MRA) ottenuta dalle wavelet. MRA, come indicato dal suo nome, analizza segna-

li con differenti risoluzioni. Esso è disegnato per dare buona risoluzione temporale e povera risoluzione di frequenza alle alte frequenze e viceversa. Questo approccio ha senso quando il segnale ha alte componenti in frequenza per brevi durate e basse componenti per lunga durata. I segnali che si incontrano in pratica hanno queste caratteristiche. La trasformata wavelet continua fu sviluppata come approccio alternativo alla trasformata di Fourier breve termine, per risolvere il problema della risoluzione. L'analisi wavelet viene fatta allo stesso modo della STFT, nel senso che il segnale è moltiplicato per una funzione, simile a quella utilizzata nella STFT; la trasformazione è computata separatamente per differenti segmenti del segnale nel dominio del tempo. Le principali differenze dalla STFT sono:

- Non è presa la trasformata di Fourier del segnale filtrato con finestra; inoltre nella trasformat di Fourier le frequenze negative relative a picchi corrispondenti a sinusoidi non saranno computati.
- la larghezza della finestra è cambiata in base alla singola componente spettrale.

Per ottenere un'analisi a risoluzione variabile occorre fare in modo che le risoluzioni relative $\Delta t/t$ e $\Delta f/f$ risultino costanti e questo richiede che all'aumentare della frequenza f aumenti in modo proporzionale la banda Δf. A tale riguardo viene in aiuto una proprietà fondamentale della trasformata di Fourier: comprimendo nel tempo una funzione si ottiene una espansione in frequenza del suo spettro, e viceversa

$$\mathcal{F}\left\{x\left(\frac{t}{a}\right)\right\} = |a|X(af)$$

Da questa considerazione nasce l'idea di sostituire l'operazione di modulazione con l'operazione di scalamento, ovvero anziché moltiplicare il segnale per la finestra $g(t)$ ad ampiezza temporale costante ed $\mathcal{F}$-trasformare si esegue direttamente il prodotto scalare con scalamenti e traslazioni di un unico prototipo. Quello che si ottiene prende il nome di trasformata wavelet continua (CWT):

$$CWT_x(a,b) = \int_{-\infty}^{+\infty} x(t)\psi^*_{ab}(t)dt = < \psi_{ab}(t), x(t) >$$

con

$$\psi_{ab}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)$$

Il prototipo $\psi(t)$ prende il nome di wavelet madre; a è il parametro di scalamento, b è il parametro di traslazione.

Il termine wavelet madre prende il nome a causa di due importanti propriettà sotto espresse:

- Il termine wavelet indica una piccola onda. La piccolezza si riferisce al fatto che la finestra di filtro è di lunghezza finita. Il termine onda viene richiamato per il fatto che la funzione è oscillatoria. Il termine madre deriva che le funzioni con differenti regioni di supporto che sono usati nel processo di trasformazione sono derivate della funzione principale. In altre parole, la wavelet madre genera le altre funzioni finestra.

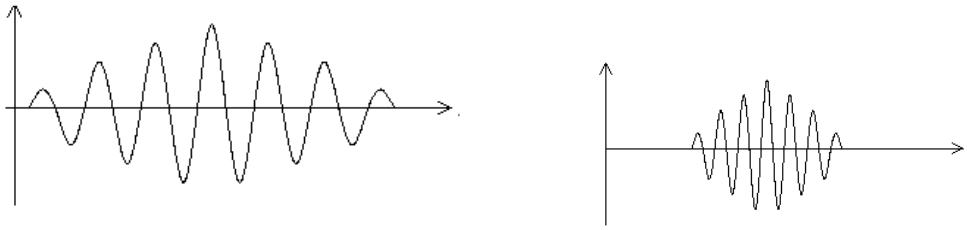
- Il termine translazione è usato con lo stesso significato definito dalla STFT: la finestra è shiftata attraverso il segnale. Inoltre non abbiamo un parametro di frequenza come la STFT; abbiamo invece il parametro di scala che è definito come $1/frequenza$ Si è notato che la STFT è una tecnica di analisi a risoluzione fissa. Si è anche osservato come la causa sia intrinseca nella presenza della funzione finestra $g(t)$ a supporto e a banda limitata, che viene modulata con $e^{j2\pi ft}$ e moltiplicata scalarmente con il segnale $x(t)$.

La denominazione wavelet deriva dal fatto che, graficamente, il prototipo è una funzione che oscilla e si smorza come una piccola onda. Un esempio classico di ondina è riportato in figura.

Si osservi un particolare: valutando la trasformata di Fourier dell'ondina, a meno del parametro di traslazione b, si ottiene

$$\mathcal{F}\left\{\frac{1}{\sqrt{|a|}}\psi\left(\frac{t}{a}\right)\right\} = \Psi(af)$$

essendo



(a) Scala grande (b) Scala piccola

Figura 3.4: Esempi wavelet madre scalati

$$\Psi(f) = \mathcal{F}\{\psi(t)\}$$

Se $\psi(t)$ ha banda Δf centrata in f_0 allora $\psi(\frac{t}{a})$ ha banda $\frac{\Delta f}{a}$ centrata in $\frac{f_0}{a}$: la banda relativa rimane costante.

Per interpretare l'informazione fornita da una rappresentazione tempo-scala, occorre svincolarsi dal concetto di frequenza, poichè confrontando la definizione di trasformata di Fourier con la definizione di trasformata wavelet continua, è immediato notare che il prodotto scalare del segnale $x(t)$ viene ora eseguito con una funzione non periodica e limitata nel tempo: il concetto di frequenza dell'armonica $e^{j2\pi ft}$ viene sostituito con il concetto di scala dell'ondina $\psi_{ab}(t)$. L'analogia è tuttavia immediata: valori piccoli di a significano ondine compresse nel tempo, quindi contenenti armoniche ad alta frequenza; effettuare il prodotto scalare con esse implica ottenere proiezioni che portano informazioni sui dettagli del segnale, cioè su fenomeni rapidamente variabili; d'altra parte, valori grandi di a comportano ondine lentamente variabili, con banda stretta, che invece colgono il comportamento del segnale a lungo termine. Queste fondamentali proprietà sono note come localizzazione temporale (più l'ondina è concentrata nel tempo, migliore è la risoluzione in questo dominio) e come localizzazione spettrale (più l'ondina è concentrata in frequenza, migliore è la risoluzione in questo dominio).

Il parametro scala nell'analisi wavelet è usato in modo simile alla scala usata nelle mappe. Un'alta scala corrisponde a una vista non dettagliata del segnale, e una scala

bassa corrisponde ad una vista dettagliata. Similmente, in termini di frequenza, basse frequenze (alte scale) corrispondono informazioni globali del segnale (viene spaziato l'intero segnale), mentre alte frequenze (basse scale) corrispondono informazioni dettagliate. In applicazioni pratiche, basse scale non ci sono per tutta la durata del segnale; esse appaiono sporadicamente. Alte scale invece permangono per tutta la durata del segnale. Funzione scalamento, come operatore matematico, riguarda la compressione o la dilatazione del segnale. Larga scala corrisponde un segnale dilatato. Se f(t) è una funzione data, f(st) corrisponde ad una funzione contratta o compressa. Se $s > 1$ è espansa; $s < 1$ dilatata. Supponiamo x(t) sia il segnale da analizzare. La wavelet madre è scelta come prototipo di tutte le finestre di processamento. Tutte le finestre che sono usate sono versioni dilatate o compresse e shiftate rispetto alla wavelet originale.

Una caratteristica fondamentale è la proprietà di invertibilità: date due ondine $\psi_1(t)$ e $\psi_2(t)$ con particolari caratteristiche sulle quali non ci si addentra, se si valuta la CWT rispetto a $\psi_1(t)$

$$CWT_{x,\psi_1}(a,b) = \int_{-\infty}^{+\infty} x(t)\psi^*_{1,ab}(t)dt$$

allora il segnale è ricostruibile attraverso la $\psi_2(t)$ e si ha

$$x(t) = \frac{1}{C_\psi}\int_0^{+\infty}\int_{-\infty}^{+\infty} CWT_{x,\psi_1}\psi^*_{2,ab}(t)\frac{dadb}{a^2}$$

dove

$$C_\psi = \int_{-\infty}^{+\infty}\frac{|\Psi_1(\omega)||\Psi_2(\omega)|}{|\omega|}d\omega$$

La definizione di trasformata wavelet continua è utilizzabile nei casi in cui si desideri una valutazione analitica della CWT di un segnale. Tuttavia nella maggior parte dei casi pratici, il segnale $x(t)$ non solo non è noto analiticamente ma risulta quantizzato, cioè è conosciuto esclusivamente in precisi istanti di tempo; se si considera inoltre il desiderio di una valutazione numerica della trasformata, allora essa dovrà essere valutata solo con un numero finito di valori delle variabili a e b.

Nel nostro progetto abbiamo utilizzato la Haar,dove la funzione madre è definita

da:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2, \\ -1 & 1/2 \leq t < 1, \\ 0 & \text{altrimenti.} \end{cases}$$

e la sua funzione padre

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1, \\ 0 & \text{altrimenti.} \end{cases}$$

Se noi abbiamo i campioni si, si+1, si+2, la Haar può essere definita calcolando i coefficienti wavelet:

$$c_i = \frac{s_{i-1} - s_{i+1}}{2}$$

mentre la funzione di scala può essere definita come segue:

$$a_i = \frac{s_{i-1} + s_{i+1}}{2}$$

Ecco il listato del metodo filter.

```
// this method is used to calculate wavelet coefficients
2
public int[] filter(byte[] values, int[] filter) {
4

6           //lazily allocate a buffer
            if (buffer == null) buffer = new int[sizeSqr];
8
            //convert values into integers
10          for (int i = 0; i < filter.length; i++) {
                filter[i] = values[i] & 0xff;
12          }

14          System.out.print(iterations);
            //filter the values
16          for (int i = 0; i < iterations; i++) {
                int[] s;
18              int[] t;
```

```
20                  int length = size >> i;
                    //horizontal processing
22                  s = filter;
                    t = buffer;
24                  int hOffset = length >> 1;
                    for (int y = 0; y < length; y++) {
26                          int sIndex, tIndex;
                            tIndex = sIndex = y * size;
28                          for (int x = 0; x < length; x+=2) {
                                    int a = s[sIndex];
30                                  int b = s[sIndex + 1];
                                    t[tIndex] = a + b;
32                                  t[tIndex + hOffset] = a - b;
                                    sIndex += 2;
34                                  tIndex ++;
                            }
36                  }
                    //vertical processing
38                  s = buffer;
                    t = filter;
40                  int vOffset = (length >> 1) * size;
                    for (int x = 0; x < length; x++) {
42                          int sIndex, tIndex;
                            sIndex = tIndex = x;
44                          for (int y = 0; y < length; y += 2) {
                                    int a = s[sIndex];
46                                  int b = s[sIndex + size];
                                    t[tIndex] = a + b;
48                                  t[tIndex + vOffset] = a - b;
                                    sIndex += size << 1;
50                                  tIndex += size;
                            }
52                  }

54                  length <<= 1;
            }
56
```

```
            //normalize the number of fractional bits before returning
58          normalize(filter);
            return filter;
60      }
```

3.7 Thresholding

Una volta analizzata la nostra immagine in tre scale differenti con la trasformata wavelet multirisoluzione, è necessario utilizzare la tecnica del thresholding (per altro utilizzata anche precedentemente dalla conversione rgb a scala di grigio) per separare le ramificazioni dei vasi sanguigni in base alla loro grandezza. Possono essere classificati sei metodi di segmentazione (da grayscale a binaria). Le stesse tecniche possono essere anche usate per immagini a colori. Un approccio possibile è utilizzare una soglia per ciascuno dei component RGB (per ciascuna banda) e combinarle assieme con un operatore AND.

- metodo basato su istrogramma: vengono analizzati picchi, le valli e le curvature dell'histogramma risultante della nostra immagine.
- metodo basato su clustering: in questo caso i campioni sono raggruppati in due parti appartenenti allo sfondo o al primo piano dell'immagne.
- metodo basato su entropia: utilizza l'entropia delle regioni dello sfondo e dell'immagine, l'entropia incrocio tra sorgente e la figura binarizzata etc.
- metodo basato su attributi: cerca la similarità attraverso l'immagine grayscale e la binaria.
- metodo spaziale: utilizza la distribuzione di probabilità e/o la correlazione tra pixel
- metodo locale: adatta il valore di soglia di ciascun pixel alla caratteristica locale dell'immagine.

3.7.1 Metodo adattivo

Thresholding è chiamato adattivo quando valori di soglia differenti sono usati in diverse regioni dell'immagine. Questo procedimento è anche conosciuto come segmentazione dinamica. Un metodo potrebbe essere il seguente:

1. viene scelto una soglia iniziale T (scelta in maniera random o con qualche altro metodo)
2. vengono creati due insiemi (G1 e G2) definiti come seguente:
   - $G1 = f(m,n) : f(m,n) > T$ (pixels dell'oggetto)
   - $G2 = G2 = f(m,n) : f(m,n) \leq T$ (pixels di sfondo)
3. la media di ciascun insieme viene computata
   - $m1 =$ media di G1
   - $m2 =$ media di G2
4. una nuova soglia viene creata dalla media delle due predenti
   - $T' = \frac{(m1+m2)}{2}$
5. si torna allo step due, utilizzando ora il nuovo valore, ripetendo finchè c'è matching tra il valore vecchio e nuovo.

3.8 Skeleton

Skeletonizzazione fu introdotta per descrivere le proprietà globali di un oggetto e per ridurre l'immagine originale di partenza in un risultato finale (Skeleton). Lo skeleton esprime il processo di restringimento usando la trasformazione Medial degli assi. Quindi un metodo base per la skeletonizzazione è il restringimento. Essa è una tecnica iterativa, che estrae lo skeleton di un oggetto come risultante. Ad ogni iterazione, i pixel dei lati che hanno almeno un punto di sfondo adiacente sono cancallati. Tutti questi

pixel saranno erosi, solamente se la rimozione non modifica la topologia della struttura. Lo skeleton rappresenta quindi la forma di un oggetto, considerando un numero minimo di pixel. Skeleton lavora per figure consistenti di linee (diritte e curve). Non opera per figure che hanno delle forme estese su grande area. La prima definizione di skeleton fu data da Blum nel 1967. Egli definì la funzione assiale mediale (MAF). Il sistema assiale mediale è definito come insieme di punti, che sono il centro di cerchi che possono contenere dentro forme di oggetti. Per rappresentare le forme come skeleton e avere la capacità di ricostruire la figura originale ci potrebbe essere una funzione raggio associata ai punti di skeleton. La MAF di una forma è il luogo dei centri di tutti i dischi massimi contenenti la forma. Maf nella sua forma originale necessita di tempo e spazio. Come se non bastasse una implementazione diretta di questa risulta molto complicata. Per questo motivo, la trasformata continua è convertita in discreta. Una semplice approssimazione è data dal calcolare prima la distanza da ciascun pixel dell'oggetto al più vicino pixel di confine, e poi calcolare il Laplaciano della distanza immagine.

3.8.1 Zhang Suen Thinning

Un altro tipo di algoritmo di Skeletonizzazione (usato nel codice allegato) è lo Zhang Suen. Questo è un metodo parallelo nel quale i nuovi valori sono dipendenti del precedente contenuto di iterazione. L'algoritmo è costruito secondo due sottoiterazioni. Nella prima, il pixel I(i,j) è cancellato se la seguente condizione è soddisfatta:

- Il suo numero di connettività è pari a 1.
- Esso ha almeno due vicini neri e non più di sei.
- Almeno uno di I(i,j+1), I(i-1,j),I(i,j-1) sono bianchi.
- Almeno uno dei pixel I(i-1,j), I(i+1,j), I(i,j-1) sono bianchi.

Nella seconda sottoiterazione le condizioni 3 e 4 cambiano

- Il suo numero di connettività è pari a 1.

- Esso ha almeno due vicini neri e non più di sei.
- Almeno uno di I(i,j+1), I(i-1,j),I(i+1,j) sono bianchi.
- Almeno uno dei pixel I(i+1,j), I(i,j+1), I(i,j-1) sono bianchi.

Se pixel soddisfano queste condizioni saranno cancellati. Alla fine di ciascuna sottoiterazione, se non ci sono più pixel da cancellare, l'algoritmo termina. Si riporta metodo usato.

```
// this method return the skeleton of a bufferedimage
public static BufferedImage zhangSuen(BufferedImage sourceimage){
    int width = sourceimage.getWidth();
    int height = sourceimage.getHeight();
    byte image[][] = BasicOperations.copy(BasicOperations.binarizeImage(sourceimage));

    //byte image[][] = BasicOperations.copy(sourceimage.getBinaryImage());
    int changes1 = 0;
    int changes2 = 1;

    while(changes1!=changes2){
        changes2 = changes1;
        changes1 = 0;
        for(int i=1; i<width-1; i++){
            for(int j=1; j<height-1; j++){

                byte p1 = image[i][j];
                byte p2 = image[i][j-1];
                byte p3 = image[i+1][j-1];
                byte p4 = image[i+1][j];
                byte p5 = image[i+1][j+1];
                byte p6 = image[i][j+1];
                byte p7 = image[i-1][j+1];
                byte p8 = image[i-1][j];
                byte p9 = image[i-1][j-1];

                int nonZero = BasicOperations.nonZeroNeighbours(i,j,image);

```

```
                    if(2 <= nonZero && nonZero <= 6){
30                      int pattern01 = BasicOperations.timesPattern01(i,j,image);

32                      if(pattern01 == 1){

34                          if(p2*p4*p6==0 && p4*p6*p8==0){
                                image[i][j] = 0;
36                              changes1++;
                            }
38
                            if(p2*p4*p8==0 && p2*p6*p8==0){
40                              image[i][j] = 0;
                                changes1++;
42                          }//3
                        }//2
44                  }//1
                }
46          }
        }
48

50      sourceimage = GetImageFromMatrix2(image);
        return sourceimage;
52  }
```

3.9 Ricostruzione ramificazioni

Il prossimo step consiste nell'analizzare i singoli alberi di ramificazione appartenenti a ciascun contenitore e memorizzarli in un apposito array unidimensionale. In questo modo saranno più semplici prossime elaborazioni di interesse. Si può notare come questa funzione sia simile a quella presente in photoshop denominata filtro ad estrazione. Esso offre un metodo per isolare un oggetto in primo piano ed eliminarne lo sfondo su un livello. Un semplice algoritmo utilizzato trova la radice dell'albero p-esimo a partire dal basso e ricostruisce la ramificazione verso l'alto. Verranno analizzati tutti pixel presenti nella coordinata (x,y-1) (se con y si identifica la ordinata del pixel k-esimo)

. Dovrà essere memorizzata la posizione in cui pixel successivo è nero. Supponiamo quindi di avere il punto richiesto nella posizione P1(x-1,y-1). Verrà aggiornato il vettore con la nuova coordinata.

```
1 if (matrix[x−1][y−1] ==0 ) // if next pixel with coordinates x−1,y−1 = black

3 {
        coordinates[g++]= new Point(x−1,y−1);
5       x=x−1;
        y=y−1;
7 }
```

Da notare che è necessario considerare se sono presenti punti di biforcazione. In questo modo verrà analizzata una funzione aggiuntiva che restituisce vero in caso affermativo; questa informazione addizionale verrà utilizzata in fase di codifica. Se sono presenti punti critici e quindi più figure curve formano l'albero i-esimo di partenza, le separo per poterle elaborare separatamente.

3.10 Rappresentazione sistema vascolare con polinomi di primo grado

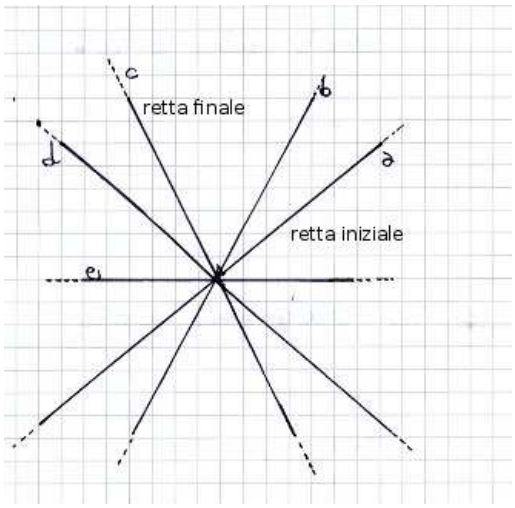


Figura 3.5: Esempio polinomi

L'idea successiva è di descrivere un sistema vascolare (di forma qualsiasi) tramite polinomio di primo grado, considerando il valore di tutti i pixel intermedi. L'idea di partenza è di prendere un polinomio iniziale descritto dall'equazione lineare $ax + by + c = 0$; la retta passerà per due punti di partenza, che consideriamo essere il primo e l'ultimo del nostro array unidimensionale (dove si sono memorizzati i pixel costituenti la nostra singola figura). In base al contenuto dei pixel, otteniamo una nuova retta inclinata di un certo valore rispetto all'originaria. Tale inclinazione dipende ovviamente dal valore trovato in ciascun pixel. Si potrebbe visualizzare questo funzionamento a effetto tergicristallo. L'inclinazione di tal retta sarà uno dei parametri fondamentali con cui verrà codificata successivamente la nostra immagine.

Da notare, nel codice allegato, il modo in cui la retta viene inizializzata nei punti radice e coda dell'array di punti analizzato. Ad ogni iterazione viene valutata la nuova direzione del pixel aggiornato rispetto a quella precedente; in base ad essa viene calcolato un valore intero. Nell'esempio riportato si presuppone che il prossimo punto ricercato abbia coordinata x,y-1. Quindi il nuovo pixel si trova in una posizione di 90^o rispetto al precedente. Il secondo punto della coordinata della retta finale terrà conto del contenuto di questo valore delta.

```
Point primaryroot = newcoordinates[0];
Point primarytail = newcoordinates[l−1];
[..]

if (coordinates[index].y == root.y && coordinates[index].x<root.x) {
        direction=90;
        delta= caldelta(direction,olddirection);
        olddirection=90;
        root.y=coordinates[index].y;
        root.x=coordinates[index].x;

}
```

3.11 Definizione alfabeto e conversione in stringhe

A questo livello siamo in presenza di immagini in coordinate polari, composte esclusivamente da rette inclinate di una certa pendenza. Un esempio di possibile figura da analizzare ed elaborare è la seguente.

Figura 3.6: Immagine da codificare in stringa

Il passo successivo sarà la codifica di tali polinomi; quindi dobbiamo definire il nostro alfabeto di partenza. Il metodo adottato consiste nel prendere in considerazione la posizione della retta i-esima nell'asse in coordinate polari e la sua inclinazione. Definiamo ora un linguaggio con cui si intende un insieme di stringhe di lunghezza finita costruite sopra un alfabeto finito, cioè sopra un insieme finito di oggetti tendenzialmente semplici che vengono chiamati caratteri, simboli o lettere. Un linguaggio su Σ è un insieme $L \subseteq \Sigma$

Quindi possiamo definire un alfabeto che definisce due linguaggi: uno per descrivere stringhe che tengono conto della posizione nell'asse x delle coordinate polari, ed uno per esprimere parole riguardo la pendenza della retta (supposto che essa sia al massimo di 180 gradi). Definiamo quindi $\Sigma = \{a_0, a_1, \ldots, a_n\}$ che sarà composto da 10 elementi; definiamo inoltre L_1 che contiene stringhe del primo tipo, e L_2 del secondo.

Sono quindi entrambi linguaggi finiti, dove $|L_1|$ = 360 elementi e $|L_2|$= 180.

La stringa finale sarà composta dalla concatenazione delle due precedenti, convertite in binario (per il motivo che si è trovato più semplice lavorare con questo tipo di codifica).

Un ulteriore fondamentale parametro per la rappresentazione è l'informazione se una figura presenta punti di biforcazione. Viene utilizzato un bit aggiuntivo settato ad uno in caso affermativo, zero in caso contrario. Da notare tuttavia che in questo modo potremmo ottenere delle stringhe di lunghezza differente in base al numero di alberi trovati nel contenitore i-esimo. Quindi un primo matching tra stringhe avverrà sulla loro lunghezza.

3.12 Matching tra stringhe e possibili soluzioni

Nel nostro problema dobbiamo effettuare il matching tra parole non precise in quanto otterremo codifiche diverse in base ad una localizzazione del disco ottico nella stessa persona ma in immagini differenti non sempre coincidente. Quindi lo stesso individuo potrà essere codificato con parole leggermente differenti. Occorrerà quindi introdurre una soglia sopra (sotto) alla quale il confronto tra stringhe è significativo. Date due stringhe A e B, la loro distanza, d(A, B), è il minimo costo di una sequenza di operazioni di trasformazione che trasforma A in B. Se non esiste nessuna sequenza di trasformazioni per passare da A in B il costo è $+\infty$. Descriviamo quindi tre differenti distanze di uso pratico.

1. Distanza di Hamming la quale

   - consente solo sostituzioni.
   - è simmetrica.
   - se $|A| = |B|$, $0 \leq d(A,B) \leq |A|$.

   Per esempio, se X = aaaccd e Y = abccd , d(X, Y) = 2 perchè ci vogliono almeno due sostituzioni.

2. Distanza di Levenshtein (denominata anche edit distance), che non è altro di una versione modificata di Hamming. Essa presenterà le seguenti caratteristiche:

   - consente solo sostituzioni, cancellazioni e inserzioni.
   - è simmetrica.
   - se $|A| = |B|$, $0 \leq d(A,B) \leq |A|$.

   Ad esempio, se X = aaaccd Y = abccd , d(X, Y) = 2 perchè ci vogliono almeno una sostituzione e una cancellazione.

3. Episode distance, che presenta le seguenti proprietà:

   - consente inserzioni.

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Figura 3.7: Edit Distance

- non è simmetrica.
- è possibile che A e B non facciano match tramite trasformazione, quindi può essere $d(A,B) = |B| - |A|$ oppure $d(A,B) = +\infty$

  Ad esempio, se X = aaaccd Y = abccd , d(X, Y) = 2 perchè ci vogliono due inserzioni.

Definiamo ora in maniera più precisa il concetto di somiglianza tra stringhe. Data una distanza, d, e una soglia di errore e, può essere definita una relazione di somiglianza tra stringhe $\sim_e$ con $A \sim_e B \Leftrightarrow d(A,B) \leq e$

Essa estende la relazione di uguaglianza tra stringhe; non è in genere transitiva e può non essere simmetrica. Una volta calcolata la distanza tra due stringhe, si può definire l'allineamento ottimale come un allineamento in cui la edit distance tra le due stringhe risultanti sia minimo.

3.12.1 Implementazione algoritmo di Levenshtein

Diamo ora una definizione induttiva di edit distance.

D[i, j] = edit distance tra A[1.. i] e B[1.. j]

base

Se j = 0, allora D[i, 0] = i Se i = 0, allora D[0, j] = j

Step	Descrizione
1	Setta n come dimensione di s. Setta m come dimensione di t Se n=0 ritorna m ed esci. Se m =0 ritorna n ed esci. Costruisci una matrice di dimensione mxn
2	inizializza la prima riga da 0..n inizializza la prima colonna da 0..m
3	Esamina ciascun carattere di s (i da 1 a n)
4	Esamina ciascun carattere di t (j da 1 to n)
5	Se s[i]= t[j] il costo è 0 Se s[i] $\neq$ t[j] il costo è 1
6	Setta la cella d[i,j] della matrice uguale al minimo di a. la cella d[i-1,j]+1 b. la cella sinistra d[i,j-1]+1 c. la cella diagonale sinistra superiore d[i-1,j-1]+cost
7	Dopo che le iterazioni (3,4,5,6) sono completate, la distanza è nella cella d[n,m]

Tabella 3.1: Algoritmo Levenshtein

caso induttivo: $i, j \neq 0$

$D[i,j] := min(D[i-1,j-1] + f(i,j), D[i-1,j] + 1, D[i,j-1] + 1)$

dove $f(i,j) = 0$, se $A(i) = B(j)$, $f(i,j) = 1$, se $A(i) \neq B(j)$

Come descritto precedentemente, l'algoritmo di Levenshtein permette di calcolare il più piccolo numero di operazioni di modifica necessarie per modificare una stringa e ottenerne una nuova. Il metodo più usato è tramite programmazione dinamica. Una matrice viene inizializzata misurando nella cella (i,j) la distanza tra il i-esimo carattere-prefisso di dimensione uno e il j-esimo prefisso dell'altra parola. La matrice sarà riempita dall'angolo superiore sinistro fino a quello in basso a destra. Ogni salto orizzontale o verticale corrisponde ad un inserimento o ad una cancellazione. Il costo per queste operazioni è unitario. Il salto diagonale può costare uno se due caratteri nella riga e colonna non fanno match, zero in caso in caso contrario. Ciascuna cella minimizza il costo locale. In questo modo il numero risultante nell'angolo destro in basso rappresenta la distanza di Levenshtein tra le due parole cercate.

Si riportano vari step utilizzati. Da notare che in ingresso ho due stringhe denominate s e t.

3.12.2 Allineamento ottimo

Come visto precedentemente, per quantificare la similarità tra due stringhe è possibile calcolare la edit distance. L'ultimo passo consiste nel computare l'allineamento ottimale. Un allineamento di due stringhe è un modo di mettere una stringa sopra l'altra, per illustrare come le parti delle stringhe sono collegate. Date due stringhe s e t, un allineamento si ottiene spostando parole modo che i caratteri delle stringhe risultanti possono essere messi in una corrispondenza biunivoca tra loro. Consideriamo il seguente esempio:

PAROLA X	polite	polite	polite
ALLINEATA Y	p.late	.plate	pla.te
d(X,Y)	2	2	2

Tabella 3.2: Allineamento ottimo

All'allineamento ottimo corrisponderà quindi distanza 2.

Per il calcolo, si procede a ritroso nella procedura di programmazione dinamica vista precedentemente. In D[n, m] è memorizzata la distanza minima tra A e B. Occorre riattraversare tutta la tabella dal basso verso l'alto e verso sinistra ricostruendo le operazioni da eseguire per trasformare la stringa A in B con costo D[n, m]. Quindi i vari step sono:

1. da D[n, m] si passa al minimo tra D[n-1, m-1], D[n-1, m] e D[n, m-1]: se il minimo è D[n-1, m-1], occorre eseguire una sostituzione se D[n-1, m-1] $\neq$ D[n, m], altrimenti non si fa nulla;

2. Se il minimo è D[n-1, m], occorre eseguire una cancellazione in B, se il minimo è D[n, m-1], occorre eseguire una inserzione in B;

3. Si ripete la stessa cosa a partire dalla cella corrispondente al minimo così ottenuto.

Capitolo 4

Conclusioni

4.1 Librerie grafiche usate e utilizzo programma

Nello sviluppo dell'interfaccia grafica si sono usate le JFC. È l'abbreviazione di Java Foundation Classes; esso ingloba un gruppo di caratteristiche per costruire interfacce utente grafiche e aggiungere ricche funzionalià ed iteratività agli applicativi java. Le Swing sono API potenti e flessibili che vengono utilizzate nel nostro prodotto. Esse sono composte da 18 package. Si può dire quindi che JFC ingloba swing.

Si riporta nelle prossime pagine una tabella in cui sono definite tutte le sue funzionalità.

La prima cosa da fare è predisporre il sistema ovvero è necessario installare tutti i

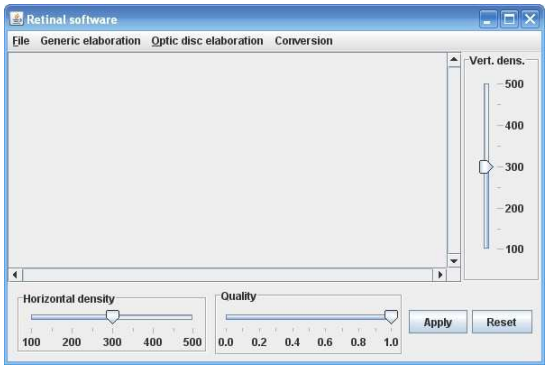


Figura 4.1: Software retina

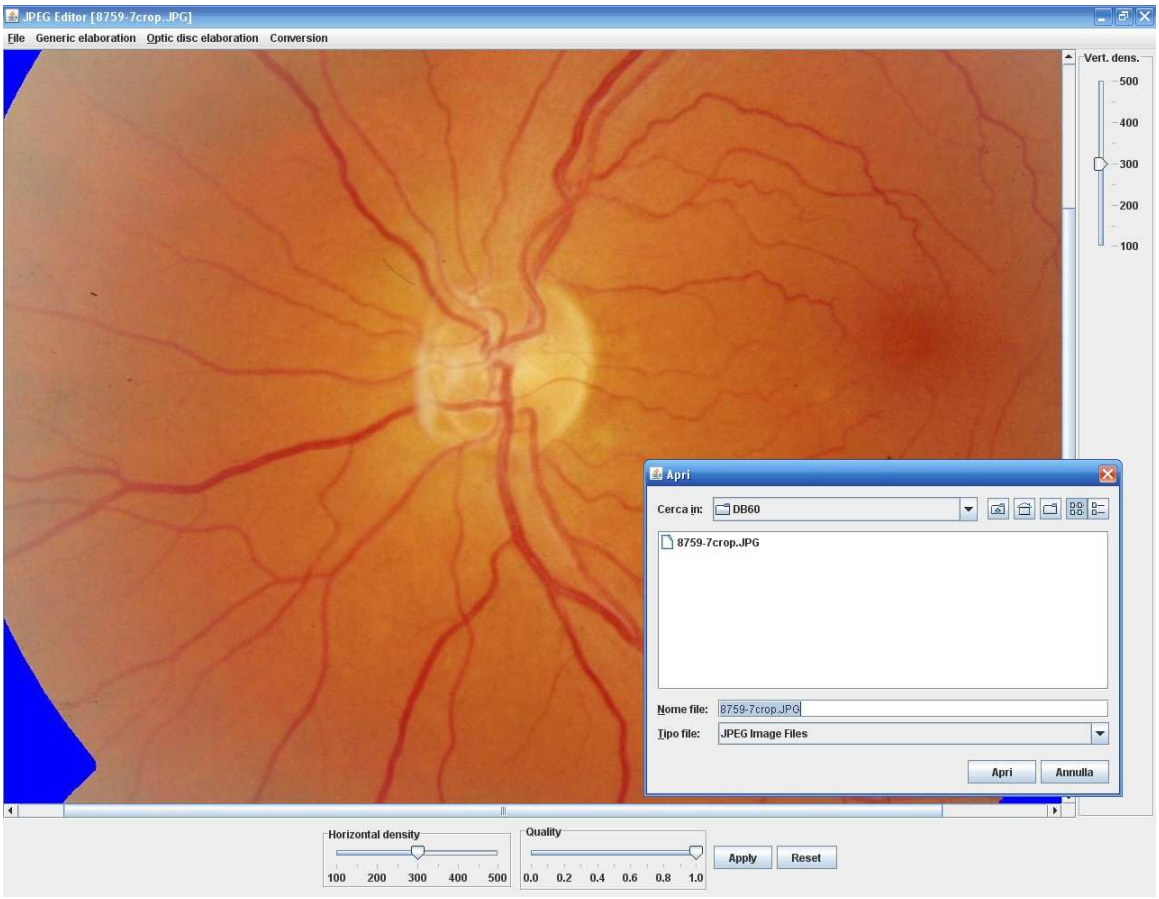


Figura 4.2: Funzioni software

Caratteristica	Descrizione
Componenti GUI swing	Include molti elementi come bottoni, tabelle. Molti componenti hanno funzionalità interne di ordinamento, stampa e Drag and Drop.
Supporto per interfaccie grafiche	L'aspetto delle applicazioni è gestito da una interfaccia ad alto livello, consentendo delle scelte soggettive sull'apparenza visiva. Per esempio, lo stesso programma può sia usare l'aspetto di java che di Windows.Addizionalmente, la piattaforma java supporta lo stile delle GTK+.
Accessibilità API	Mette a disposizione tecnologie assistenti come display Braille, per dare informazioni dall'interfaccia utente.
Java 2D API	Attiva lo sviluppatore a incorporare immagini di alta qualità 2D, testo, applet. Java 2D include API di estensione per generare e spedire output di alta qualità verso dispositivi esterni.
Internazionalizzazione	Consente a sviluppatori di costruire applicazioni che possono interagire con utenti del mondo nella loro propria cultura e lingua. Si possono creare applicativi che accettano del testo in migliaia di lingue con differenti caratteri come giapponese,cinese, coreano.

Tabella 4.1: JFC

javax.accessibility	javax.swing.plaf	javax.swing.text
javax.swing	javax.swing.plaf.basic	javax.swing.text.html
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.html.parser
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.text.rtf
javax.swing.event	javax.swing.plaf.synth	javax.swing.tree
javax.swing.filechooser	javax.swing.table	javax.swing.undo

Tabella 4.2: Swing Package

software necessari per il corretto funzionamento. Ne consegue che per l'esclusivo utilizzo del prodotto finito è sufficiente l'installazione del pacchetto jre-1.5.x last version, altrimenti se si ha la necessita di modificare il codice sorgente si è obbligati a installare il pacchetto jdk-1.5.x.

Per eseguire il programma principale, basta lanciare comando $''java - retina''$ nel package look. Si sono utilizzate esclusivamente librerie standard, quindi non è necessario installare nulla di aggiuntivo. Si riportano inoltre alcuni screenshot catturati durante il funzionamento del software.

4.2 Sviluppi futuri

Ulteriori aggiunte potrebbero essere lo sviluppo del sistema descritto tramite agenti mobili. I classici modelli client-server che sono stati, e lo sono tuttora, cardine fondamentale di questa branca di ricerca, non riescono a rispondere in modo esaustivo a tutti i quesiti e a tutte le problematiche che si possono incontrare in tali ambiti, ed è proprio per questo motivo che diviene necessario effettuare una ridefinizione dei requisiti di dinamicità, scalabilità, fault-tolerance, adattabilità e sicurezza richiesti dalle nuove applicazioni. Risulta quindi necessario introdurre nuovi paradigmi di programmazione incentrati sulla mobilità, sia della computazione (mobile computing) sia del codice (mobile code), ovvero sulla possibilità di descrivere un programma che possa essere spedito senza cambiamenti ad una serie di calcolatori eterogenei ed eseguito con la stessa semantica su ognuno di essi; usualmente ci si riferisce al codice mobile come a software in grado di viaggiare su una rete molto differenziata, attraversando vari domini di protezione, e che può essere eseguito automaticamente all'arrivo a destinazione. I vantaggi di tale approccio sono molteplici: fra i più importanti citiamo l'efficienza (ripetute interazioni con un server remoto possono essere soppresse spedendo il processo direttamente al client in modo che vi interagisca localmente), semplicità e flessibilità (il mantenimento di una rete può essere molto più semplice quando le applicazioni sono su un server, ed i client, autonomamente, quando necessario, le installano sul proprio sito, scaricandole on demand) ed infine l'occupazione di spazio (scaricare il codice quando necessario, permette di ridurre la duplicazione del codice sui vari si-

ti). In tale scenario ha assunto una notevole importanza un paradigma, detto ad agenti mobili, il quale ha riscosso notevole interesse soprattutto per la facilità con la quale esso permette di realizzare applicazioni distribuite. Tale paradigma prevede lo sviluppo di entità software autonome, chiamate appunto agenti, capaci di muoversi spontaneamente fra gli host di una rete ed interagire con le risorse disponibili su tali nodi ed eventualmente con altri agenti. Si comprende subito che è fondamentale modellare correttamente le interazioni cui tali agenti possono venire sottoposti, in modo da renderle affidabili e sicure. La sicurezza gioca un ruolo preponderante nello sviluppo di tale tipo di applicazioni, in quanto solo con un adeguato modello di sicurezza si potrà favorirne lo sviluppo. È da sottolineare che comunque l'ottenimento e la gestione di uno specifico livello di sicurezza è un compito non banale, specialmente per ciò che riguarda l'alta dinamicità di interazione cui gli agenti sono sottoposti.

Bibliografia

[1] C. Beccari, Latex, Hoepli, 1981;

[2] Wilhelhelm Burger, Mark J. Burge, Principles of Digital Image Processing, Fundamental Techniques, Springer

[3] Wilhelhelm Burger, Mark J. Burge, Principles of Digital Image Processing, Core Algorithms,Springer

[4] Robert Buzz Hill Retina Identification

[5] Computer Vision : a modern approch

[6] R.M. Haralick and L.G. Shapiro. Computer and robot vision. Addison Wesley, 1992.

[7] Programming in Java Advanced Imaging

[8] Hadi Farzin, Hamid Abrishami-Moghaddam, and Mohammad-Shahram Moin, A Novel Retinal Identification System, 21 February 2008.

[9] M. Ortega, M. G. Penedo, J. Rouco, N. Barreira, and M. J. Carreira, Retinal Verification Using a Feature Points-Based Biometric Pattern

[10] Enrico Grisan, Automatic Analysis of Retinal Images: Retinopathy Detection and Grading, 3rd February 2005

[11] M. Foracchia, E. Grisan, and A. Ruggeri, Detection of optic disc in retinal images by means of a geometrical model of vessel structure, IEEE Transactions on Medical Imaging, October 2004.

[12] Marco Tranquillin: Sistema sicuro di autenticazione biometrica ad agenti mobili su Jade.

[13] Eric N. Mortensen, Hongli Deng, Linda Shapiro, A SIFT Descriptor with Global Context

[14] Ali Can, Hong Shen, James N. Turner, Howard L. Tanenbaum, and Badrinath Roysam, Member, Rapid Automated Tracing and Feature Extraction from Retinal Fundus Images Using Direct Exploratory Algorithms

[15] V. Bevilacqua S. Cambò L. Cariello G. Mastronardi, A combined method to detect retinal fundus features

[16] V. Bevilacqua, G. Mastronardi, A. Colaninno, and A. D'Addabbo. Retina images processing using genetic algorithm and maximum likelihood method. November 2004

[17] Alauddin Bhuiyan, Baikunth Nath, Joselito Chua and Ramamohanarao Kotagiri, blood vessel segmentation from color retinal images using unsupervised texture classification

[18] S.Kother Mohideen Dr. S. Arumuga Perumal, Dr. M.Mohamed Sathik, Image De-noising using Discrete Wavelet transform

[19] Alberto Martin and Sabri Tosunoglu, image processing techniques for machine vision

[20] C. Marino, M. G. Penedo, M. Penas, M. J. Carreira,F. Gonzalez, Personal authentication using digital retinal images

[21] Adam Hoover, Michael Goldbaum , Locating the Optic Nerve in a Retinal Image Using the Fuzzy Convergence of the Blood Vessels

[22] D. Gusfield, Algorithms on Strings, trees, and sequences, Cambridge University Press, 1997.

Ringraziamenti

Volevo ringraziare in primo luogo il mio relatore, il Prof. Carlo Ferrari per l'aiuto ed i preziosi consigli che mi sono stati dati non solo durante la stesura di questo mio lavoro di tesi ma durante tutta la mia carriera universitaria ed al Prof. Grisan per utili suggerimenti durante il lavoro svolto. Un ringraziamento particolare lo devo ovviamente ai miei genitori, che mi hanno sempre sostenuto ed incoraggiato, soprattutto nei momenti di difficoltà, ed il raggiungimento dei miei risultati è solamente merito dei loro grandi sforzi e sacrifici.