UNIVERSITÀ DEGLI STUDI DI CAMERINO

SCUOLA DI SCIENZE E TECNOLOGIE

Corso di Laurea in Matematica e Applicazioni
(Classe LM-40)

Applicazione della triangolazione di Delaunay e dell'Edit Distance per il confronto di impronte digitali

Tesi di Laurea in Analisi Numerica

Relatori:
Prof. Riccardo Piergallini
Prof. Pierluigi Maponi

Laureanda:
Nadia Petracci

ANNO ACCADEMICO 2017-2018

Indice

Introduzione

L'identificazione delle persone è sempre stata un elemento centrale della lotta alla criminalità; infatti essa è di particolare importanza sia negli studi civili e sia, soprattutto, negli studi penali per l'individualizzazione dei rei e riconoscimento dei cadaveri. Fin dall'antichità sono state utilizzate molteplici tecniche, come ad esempio, incisioni su tavolette di argilla, marchi impressi sulla fronte o lacerazioni delle orecchie. Mano a mano che la scienza si è evoluta, anche i metodi di identificazione si sono sviluppati e soprattutto migliorati nel corso degli anni. Un primo passo è stato fatto già agli inizi del XIX secolo, con lo sviluppo della fotografia, ma il primo strumento veramente efficace ed attualmente ancora in uso è la *dattiloscopia*, ovvero il riconoscimento delle persone per mezzo delle loro impronte digitali.
La dattiloscopia ha un ruolo fondamentale per quanto riguarda aspetti di sicurezza; sia aspetti forensi, come l'analisi e la documentazione della scena del crimine, ma anche per la gestione dell'accesso in determinati luoghi tramite porte regolate da serrature biometriche. L'importanza dell'utilizzo di impronte digitali si fonda sull'unicità e sull'immutabilità nel tempo dei *dermatoglifi* formati dalle creste papillari sui palmi delle mani o sulle piante dei piedi. Infatti, tutti gli oggetti che vengono toccati, presentano tracce di impronte digitali, ciò è dovuto al sudore o alle secrezioni che escono dai pori della pelle, i quali fungono come una specie di inchiostro per le impronte. Perciò, a seconda del luogo in cui è stata rinvenuta, un'impronta può contribuire ad incolpare o anche scagionare una persona per un presunto reato.
Al giorno d'oggi, le impronte digitali, vengono usate comunemente anche per differenti sistemi di chiusure (porte, lucchetti, chiavistelli) o per bloccare o sbloccare il proprio smartphone o personal computer. Tutto ciò consente, infatti, di selezionare persone che possono accedere a determinati servizi o in determinate aree e fungere così come sistemi di difesa (antifurto), individuando eventuali trasgressori. Infatti, le tecniche di identificazioni sono finalizzate a identificare un individuo sulla base delle sue peculiari caratteristiche comportamentali o fisiologiche.

Con questa tesi ho voluto esplorare ed approfondire il riconoscimento biometrico, per poi concentrare l'attenzione sul confronto di impronte digitali appartenenti al database "*NIST Special Database 4 Gray Scale Images of Fingerprint Image Groups(FIGS)*".

Un'impronta digitale è costituita da un insieme di linee, chiamate *ridge lines* e da *valli*, che a volte si intersecano o si interrompono, formando un disegno detto *ridge pattern*. A partire dal ridge pattern possono essere estratte ulteriori informazioni, come ad esempio le minutiae. Quest'ultime, costituiscono un fattore importante per la discriminazione delle impronte, in quanto restano identiche per tutta la vita di una persona; esse sono i punti in cui si ha un comportamento anomalo delle ridge line. Sono state individuate sette tipologie di minutiae, ma principalmente vengono considerate solo due classi: le terminazioni (punti in cui una ridge line s'interrompe bruscamente) e le biforcazioni (punti in cui una ridge line si divide in due o più rami). L'idea è quella di associare una struttura topologica alle minutiae.
Inizialmente, nel mio lavoro, vengono estratti i punti minutiae, memorizzando per ciascuna le proprie coordinate (x,y), che verranno utilizzate per la costruzione di triangoli minutiae di Delaunay. Questi triangoli hanno un buon potere discriminatorio, infatti sono gli unici a soddisfare le proprietà della triangolazione di Delaunay:

- l'unicità;
- può essere calcolata in modo efficiente in tempo O(N);
- il rumore ha un'influenza solamente locale.

La triangolazione consiste nella suddivisione di una regione dello spazio in sottoregioni rappresentanti triangoli. Dato un insieme S di punti $p_1, ..., p_n$ si può calcolare la triangolazione di Delaunay partendo dal diagramma di Voronoi. Quest'ultimo suddivide lo spazio in regioni, attorno a ciascun punto, in modo tale che tutti i punti nella regione attorno a p_i , siano più vicini a questo punto, rispetto a qualsiasi altro punto di S. Dato il diagramma di Voronoi, la triangolazione di Delaunay può essere costruita collegando il centro di ogni coppia di regioni di Voronoi adiacenti. Tale triangolazione prende il nome da *Boris Delaunay* per il suo lavoro su questo argomento nel 1934.
Una volta costruiti i grafi tramite la triangolazione di Delaunay, essi vengono confrontati e per ciascuna coppia di impronte digitali, dopo il matching finale, si otterrà il valore dell'Edit Distance. L'Edit Distance quantifica quanto dissimili siano due grafi, contando il numero minimo di operazioni richieste per trasformare un grafo in un altro.

Ciascuna operazione di edit presenta un costo; tali operazioni sono le seguenti:

- sostituzione di un lato con un altro;
- cancellazione di un vertice;
- cancellazione di un arco;
- inserimento di un vertice;
- inserimento di un arco.

In questo elaborato vengono utilizzate le due applicazioni appena osservate, la triangolazione di Delaunay e l'Edit Distance, per effetturare il matching finale tra le impronte digitali. Ovviamente, più il valore dell'Edit Distance tra i grafi di due impronte digitali sarà minore, maggiore è la probabilità che le due impronte appartengano alla stessa persona.
In particolare sono state svolte le seguenti attività:
Nel ***Capitolo 1*** è stato introdotto il riconoscimento biometrico, osservando nel dettaglio le caratteristiche biometriche e soffermando principalmente l'attenzione sulle impronte digitali, sia per quanto riguarda il loro aspetto storico, che l'aspetto anatomico.
Nel ***Capitolo 2*** ho introdotto il concetto di Triangolazione di Delaunay e Diagramma di Voronoi, osservando la loro dualità e soprattutto evidenziando le principali caratteristiche che rendono tale triangolazione preferibile rispetto ad altre.
Nel ***Capitolo 3*** è stata messa in evidenza l'applicazione della Triangolazione di Delaunay sulle impronte digitali, focalizzando inizialmente l'attenzione su come avviene l'estrazione di punti minutiae, per poi costruire su tali punti la triangolazione di Delaunay.
Il ***Capitolo 4*** presenta l'applicazione dell'Edit Distance, mostrando la sua importanza nel confronto non solo di grafi, ma anche e soprattutto di stringhe o alberi. Alla fine di tale capitolo viene osservata nel dettaglio l'applicazione chiamata GED, Graph Edit Distance, che viene appunto utilizzata per confrontare due grafi.
Infine, nel ***Capitolo 5***, vengono mostrati i risultati ottenuti dall'applicazione della funzione GED, implementata su MATLAB, tramite tabelle, che mettono in evidenza i valori dell'Edit Distance tra i grafi di coppie di impronte digitali.

Capitolo 1

Biometria e sistemi biometrici

1.1 Introduzione alla Biometria

Il riconoscimento biometrico "*Biometric Recognition*", o semplicemente la biometria, dal greco *bios* (“vita”) e metron (“*misurazione*”), è la scienza che studia come stabilire l’identità di un individuo a partire da attributi fisici, chimici, o comportamentali.
I sistemi di sicurezza biometrici possono utilizzare una vasta gamma di caratteristiche fisiche e comportamentali. [1]

Le principali caratteristiche biometriche **fisiologiche** sono:

- *Impronte digitali*: grazie alle loro caratteristiche, le impronte digitali sono gli identificatori biometrici più diffusi.
- *DNA*: è il codice che contiene le informazioni genetiche di una persona. Molto usato in applicazioni forensi, poiché ogni individuo possiede un codice genetico diverso (ad eccezione dei gemelli monozigoti). È generalmente isolato dal sangue, dalla pelle, dalla saliva, dai capelli e da altri tessuti e fluidi biologici, per identificare i responsabili di atti criminosi, come delitti o violenze.

Il processo utilizzato è il *fingerprinting* genetico: tale tecnica consiste nel comparare la lunghezza delle sezioni variabili del DNA ripetitivo, che possono essere molto diverse tra un individuo e l'altro.

- *Geometria della mano e delle dita*: alcune caratteristiche della mano umana (come ad esempio la lunghezza delle dita) sono relativamente invarianti e specifiche (ma non uniche) di ciascun individuo.

- *Volto*: è una delle caratteristiche biometriche più accettate perché è il metodo più comune che utilizzano gli esseri umani nelle loro interazioni quotidiane.

- *Iride*: la tessitura dell'iride di un occhio umano è assunta ormai come una caratteristica distintiva di ogni individuo e di ogni occhio. La tecnologia di riconoscimento dell'iride è estremamente precisa e veloce su immagini di iride ben catturate ad alta risoluzione.

Le principali caratteristiche biometriche **comportamentali** sono:

- *Voce*: il timbro vocale di una persona è una caratteristica biometrica distintiva. Tuttavia non è molto peculiare da permettere l'identificazione di un individuo in grandi database, in quanto è influenzata anche da fattori come la salute, lo stress e lo stato emotivo di una persona.

- *Firma*: il modo con il quale una persona firma è una caratteristica distintiva di un individuo. Oggi è la forma più accettata come metodo di verifica dell'identità nelle transazioni commerciali. La firma può variare nel tempo ed è influenzata dalle condizioni fisiche ed emotive degli individui.

Nell'ingegneria elettronica, l'uso di caratteristiche biometriche ai fini del riconoscimento presenta numerosi vantaggi rispetto ai tradizionali sistemi d'autenticazione basati sull'uso di PIN e password, che possono facilmente essere dimenticati dai legittimi proprietari, ceduti ad altri, o rubati da utenti non autorizzati. Proprio per questi motivi i sistemi biometrici sono ritenuti molto più affidabili dei sistemi tradizionali, vista la difficoltà di falsificazione di molte caratteristiche biometriche. I vantaggi del riconoscimento biometrico sono: la praticità dell'utente (ad esempio il ritiro del denaro presso un bancomat senza carta o PIN), maggiore sicurezza (ad esempio solo una persona autorizzata può entrare in una struttura), maggiore efficienza (ad esempio un sovraccarico ridotto rispetto alla manutenzione della password del computer).

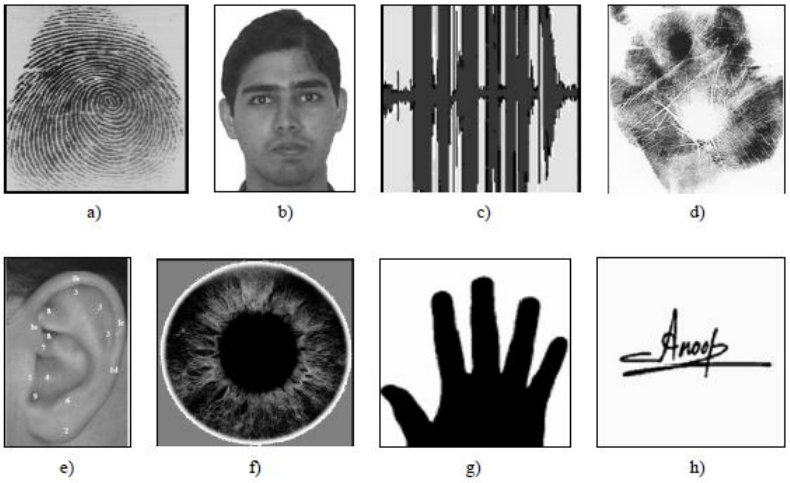


*Figura 1.1: **a)** Fingerprint **b)** Viso **c)** Voce **d)** Palmprint **e)** Orecchio **f)** Iride **g)** Geometria della mano **h)**Firma*

Qualsiasi tratto anatomico o comportamentale umano può essere utilizzato come identificatore biometrico per riconoscere una persona, purché soddisfi i seguenti requisiti:

1. *Universalità*: tutte le persone possiedono tale caratteristica biometrica.
2. *Unicità*: la caratteristica di ogni persona è sufficientemente diversa da quelle delle altre persone.
3. *Permanenza*: la misura biometrica deve rimanere sostanzialmente invariata nel tempo.
4. *Misurabilità*: la caratteristica biometrica deve essere misurabile quantitativamente.
5. *Prestazioni*: l'accuratezza del riconoscimento e le risorse richieste per ottenerla devono essere compatibili con l'applicazione.
6. *Accettabilità*: gli individui della popolazione che utilizzerà il sistema di sicurezza dovranno essere propensi a sottoporre il proprio tratto biometrico.
7. *Aggiramento*: consiste nella difficoltà di imitare o riprodurre il proprio tratto per aggirare il controllo offerto dal sistema.

Tali caratteristiche si riferiscono a un sistema ideale e gli indicatori devono solo essere ammissibili, cioè essere scelti in modo che si avvicinino sufficientemente alle caratteristiche ideali.
Durante la progettazione di un sistema biometrico è necessario considerare altre questioni quali velocità, robustezza e precisione di riconoscimento. È inoltre opportuno esaminare il fattore di elusione come indicatore della facilità con la quale è possibile ingannare il sistema con metodi fraudolenti.
Alcune serie televisive, che ruotano attorno alle indagini di polizia scientifica, hanno contribuito ad aumentare l'attenzione da parte dell'opinione pubblica sull'importanza del lavoro dei servizi scientifici e soprattutto sulla procedura d'identificazione per mezzo delle impronte digitali, le quali a differenza di quanto si vede in queste serie, sono molto impegnative.

1.2 Sistema Biometrico

Un sistema biometrico è sostanzialmente un meccanismo in grado di raccogliere dati da un individuo, estrarre le caratteristiche d'interesse e confrontarle con i modelli già presenti nella banca dati [2] . In base ai risultati del confronto il sistema reagirà in maniera positiva o negativa.
I sistemi biometrici assumono nomi diversi secondo l'ambiente applicativo in cui operano:

- *Sistema biometrico di verifica*: esegue un confronto tra la caratteristica biometrica di una persona con quella del suo modello già memorizzato nel sistema. Viene effettuato un confronto "uno a uno" per verificare che l'identità che l'individuo dichiara di possedere sia effettivamente corretta.

- *Sistema biometrico d'identificazione*: compie un riconoscimento di un soggetto cercando fra tutti i modelli dell'intero database, all'interno del quale viene condotta una ricerca "uno a molti" per stabilire, se nota, l'identità dell'individuo.

Si può suddividere un sistema biometrico in moduli logici [1] :

- *Modulo di Enrollment*: durante questa fase la caratteristica biometrica viene acquisita da un lettore biometrico che produce una rappresentazione digitale della stessa. Viene effettuato un controllo per verificare la qualità della rappresentazione prima che venga processata ulteriormente da un estrattore di caratteristiche ("*feature extractor*") che genera un template da memorizzare nel database di sistema.

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

- *Modulo d'identificazione*: l'utente non deve fornire alcun PIN o username poiché il sistema confronta i dati biometrici acquisiti con tutti i template presenti nel database. Il sistema può riconoscere l'identità dell'utente oppure non riconoscerla, segnalando un errore.

A seconda del dominio dell'applicazione un sistema biometrico potrebbe funzionare sia come sistema online che come sistema offline.

- *Sistema online*: richiede che il riconoscimento sia eseguito velocemente e con una risposta immediata (ad esempio un'applicazione di accesso alla rete del computer). Le caratteristiche biometriche sono catturante usando dei live scanner e sia la fase di enrollment che la fase di decisione sono del tutto automatiche.

- *Sistema offline*: non richiede che il riconoscimento abbia una risposta istantanea. È propria di sistemi semi-automatici che richiedono l'intervento umano nelle varie fasi (ad esempio un operatore che scansiona manualmente un'impronta precedentemente acquisita).

Tutto ciò è riassunto nella seguente ***Figura 1.2***:

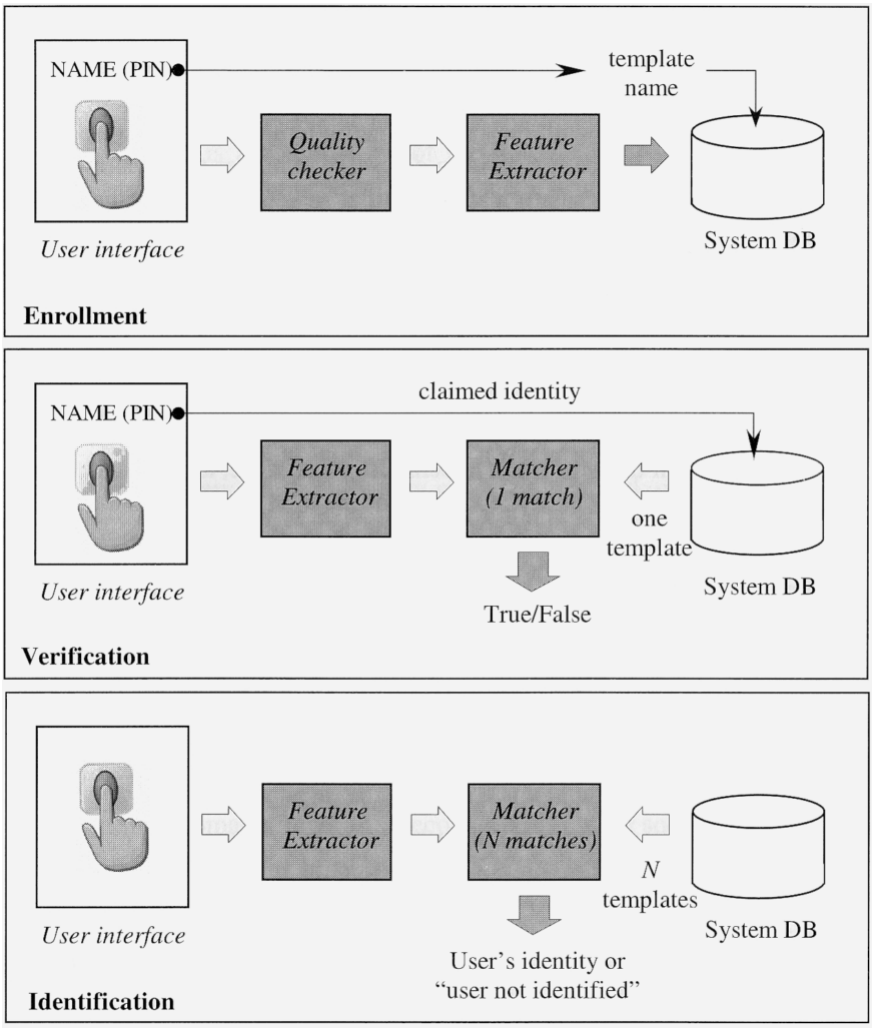


Figura 1.2: *Processi di Enrollment, verifica ed identificazione. Per ogni procedura sono schematizzate le operazioni compiute dai singoli moduli.*

1.3 Le impronte digitali

Tra i principali identificatori biometrici si collocano senza dubbio le impronte digitali. Il maggiore utilizzo del riconoscimento basato su impronte digitali si trova in ambito commerciale, governativo, civile e domini finanziari. I sistemi di riconoscimento delle impronte digitali, se correttamente implementati, offrono maggiore sicurezza, convenienza ed efficienza rispetto a qualsiasi altro mezzo d'identificazione. Tali sistemi di riconoscimento hanno già sostituito le password, i Pin o le carte in moltissime applicazioni. Ciò ha portato a una riduzione del numero di furti e ad una maggiore protezione della privacy. Grazie alla continua crescita e allo sviluppo di questa nuova tecnologia, sta aumentando l'interazione tra i mercati, le tecnologie e le applicazioni. È dunque chiaro che il riconoscimento basato sulle impronte digitali avrà un'influenza sempre più profonda sulle nostre vite quotidiane.

1.3.1 Cenni storici

Da studi archeologici è emerso come le impronte digitali fossero già usate nel 500 a.C. come forma d'identificazione personale. Tra le rovine di Ninive (città dell'antica Mesopotamia situata sulle rive del fiume Tigri nell'odierno Iraq) sono state ritrovate 25000 tavole di argilla. Queste tavolette, che riguardavano transazioni commerciali e atti ufficiali, erano incise in caratteri cuneiformi e riportavano, oltre al nome dell'autore anche un'impronta delle sue unghie. Su numerose tavolette, insieme alle impronte delle unghie, sono impresse anche parti delle impronte dei polpastrelli e delle creste papillari dell'autore.
Si presume quindi che tali civiltà fossero già consapevoli dell'unicità del disegno papillare, come dimostra la scelta di utilizzare le impronte per «siglare» le tavolette. [1] In Cina, nel 247 a.C. l'imperatore Qin Shin Huang, utilizzava le impronte digitali, incise nell'argilla, per apporre il proprio sigillo sui documenti.
Durante la dinastia Tang (618–907 d.C.) la carta e la seta fecero la loro comparsa in Cina. Le impronte digitali e palmari impresse sui documenti ne confermavano l'autenticità. Durante questo periodo, il poeta cinese Kia Kung-Yen descrisse in modo esplicito il metodo «hua-chi» (impronta digitale) utilizzato per la prevenzione degli scambi d'identità. Anche in Europa sono stati rinvenuti reperti di documenti siglati con impronte digitali risalenti all'epoca dell'Impero Romano. In Persia, nel XIV° secolo, osservando diversi documenti ufficiali che erano stati siglati con impronte, un ufficiale notò che non esistono due impronte esattamente uguali.

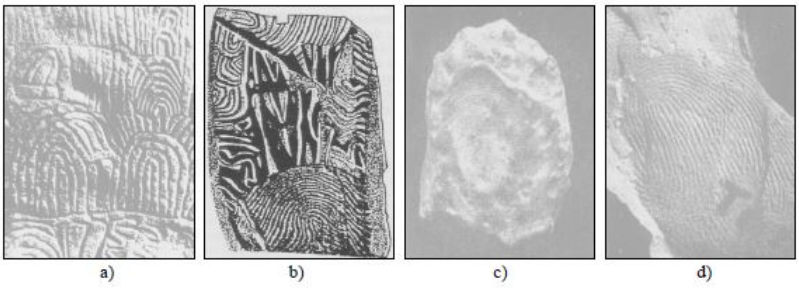


Figura 1.3: *Esempi di incisioni archeologiche di impronte digitali.* ***a)****Scultura Neolitica* ***b)*** *Pietra situata nell'isola di Goat Island (2000 a.C.)* ***c)*** *Sigillo di argilla Cinese (300 a.C.)* ***d)****Impronta su lampada palestinese (400 d.C.)*
Mentre le impronte sulle incisioni neolitiche e le pietre erette di Goat Island non possono essere utilizzate per stabilire l'identità, ci sono prove sufficienti per suggerire che il sigillo in argilla cinese e le impronte sulla lampada palestinese sono stati utilizzati per indicare l'identità degli individui.

La *dattiloscopia*, lo studio delle impronte digitali, ha però origini più recenti. Nel 1665 Marcello Malpighi, professore di anatomia all'Università di Bologna, scrisse uno dei primi documenti scientifici riguardante le creste cutanee, il "*De externo tactus organo anatomica observatio*", ma non enfatizzò l'importanza delle impronte ai fini dell'identificazione personale. Neanche vent'anni più tardi, nel 1684 il botanico e fisico inglese Nehemiah Grew pubblicò un suo studio sistematico sulle creste, il solco e la struttura dei pori nelle impronte digitali. In seguito nel 1788, l'anatomista tedesco J.C.A. Mayer individuò alcune caratteristiche ricorrenti delle impronte digitali, sostenendo anche la loro unicità da individuo a individuo. Nel 1823 fu fatta una prima classificazione delle impronte, ideata dal professore di anatomia Jan Evangelista Purkyne, basandosi sulla struttura generale delle creste. Egli descrisse nella sua tesi nove pattern riscontrabili nelle impronte, ma non fece riferimento all'importanza delle impronte ai fini del riconoscimento.
Nel 1856, William Herschel, nel suo lavoro di magistrato a Jungipoor,(India), utilizzò le impronte nei contratti degli abitanti del luogo, tenendo in considerazione la credenza locale secondo la quale il contatto personale con il documento sarebbe stato maggiormente vincolante rispetto a una semplice firma. Inoltre Herschel si convinse che l'impronta potesse confermare o negare l'identità, grazie alle caratteristiche di unicità dell'impronta e di persistenza nel corso della vita [3] .
Nel 1880, Henry Faulds, medico scozzese e il sovrintendente del reparto di chirurgia dell'ospedale di Tsukiji Hospital a Tokio, (Giappone), pubblicò un articolo sulla rivista scientifica "*Nature*", nel quale sosteneva l'individualità delle impronte digitali e ne suggerì un loro possibile utilizzo nel campo forense.

Il Dr. Faulds chiese a Charles Darwin un supporto scientifico e Darwin, ormai in età avanzata, segnalò suo cugino, Sir Francis Galton, un antropologo inglese.
Galton, il cui interesse principale era l'uso delle impronte come un ausilio nello studio dell'ereditarietà razziale, pubblicò nel 1892 un libro, intitolato "*Fingerprints*" nel quale introdusse la nozione di minuzia e suggerì il primo semplice sistema di classificazione di impronte digitali, basato sul raggruppamento dei pattern in *arches, loops e whorls*.
Galton scoprì presto che le impronte non avevano alcuna utilità nella determinazione della storia genetica dell'individuo, ma provò scientificamente la loro individualità e persistenza.
Nel 1892 Vucetich, ufficiale di polizia argentino, raccolse i primi file di impronte basandosi sulla classificazione di Galton ed effettuò la prima identificazione di un criminale.
Nel 1904 egli pubblicò "*Dactiloscopia comparada*", libro tutt'ora usato in molto paesi di lingua spagnola.
In Inghilterra e nel Galles l'introduzione delle impronte per l'identificazione dei criminali cominciò nel 1901, sulla base delle teorie di Francis Galton riviste in seguito da Edward Richard Henry (che divenne poi il commissario capo della polizia metropolitana di Londra).
Nel giugno 1900 fu pubblicato il sistema di classificazione di Galton-Henry e adottato da numerosi uffici di polizia in diversi stati.
Il primo caso d'identificazione di persone mediante l'utilizzo delle impronte digitali risale al 1902 nel *Civil Service Commission* di New York.
Nel 1924 l'FBI (*Federal Bureau of Investigation*) creò una divisione d'identificazione delle impronte digitali con un database di 810000 schede di impronte digitali.
Nel 1971 Moenssens A., in "*Fingerprint Techniques, Chilton Book Company*" riassunse i principi biologici delle impronte digitali:

1. Le creste e i solchi epidermici sono differenti per ogni impronta digitale.

2. I vari tipi di configurazioni delle strutture papillari sono variabili solo all'interno di limiti ben individuati, che consentono lo sviluppo di una classificazione sistematica.

3. Le impronte papillari sono invariabili per un dato polpastrello.

Il primo principio costituisce il fondamento del riconoscimento delle impronte digitali, mentre il secondo costituisce il fondamento della classificazione delle impronte digitali.

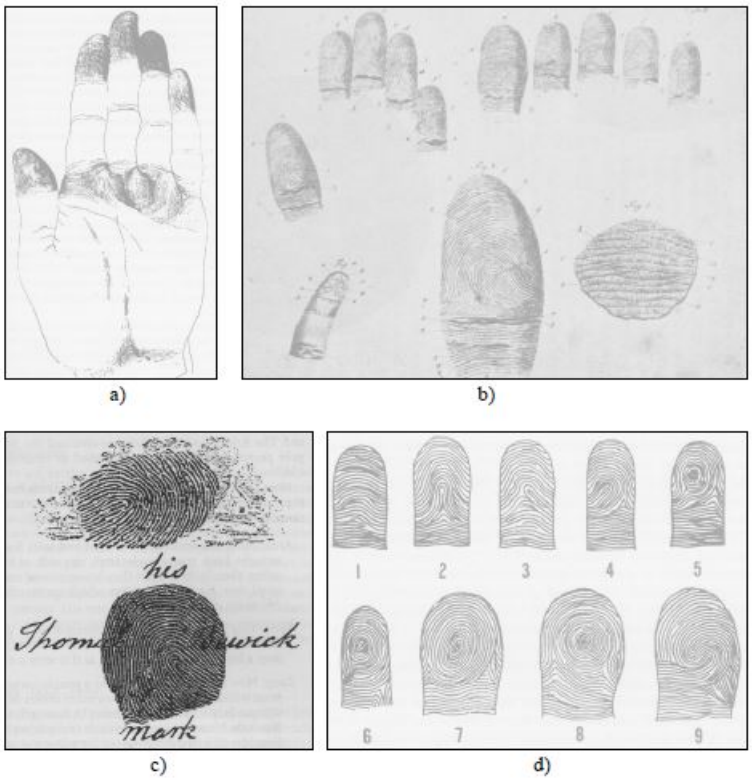


Figura 1.4: *a) Dermatoglifi estratti da Grew b) Prelievi di Mayer di impronte digitali c) Marchi di Thomas Bewick d) Nove modelli illustrati nella tesi di Purkinjk*

Negli anni ’60 l’FBI sviluppò il primo sistema automatico di identificazione delle impronte: AFIS (*Automated Fingerprint Identification System*). L’AFIS è un sistema hardware e software costituito da terminali a disposizione delle singole unità di polizia scientifica, o dei RIS dei Carabinieri, che hanno la possibilità di connettersi, via rete telematica, alla Banca dati del Casellario Centrale d’Identità, che contiene le informazioni biometriche dei singoli soggetti per essere identificati ai fini preventivi o giudiziari.
Tali sistemi permettono l’archiviazione di un gran numero di impronte digitali in un database e la rapida effettuazione dei confronti per l’identificazione delle impronte (matching) tramite opportuni algoritmi.

1.3.2 Anatomia delle impronte digitali

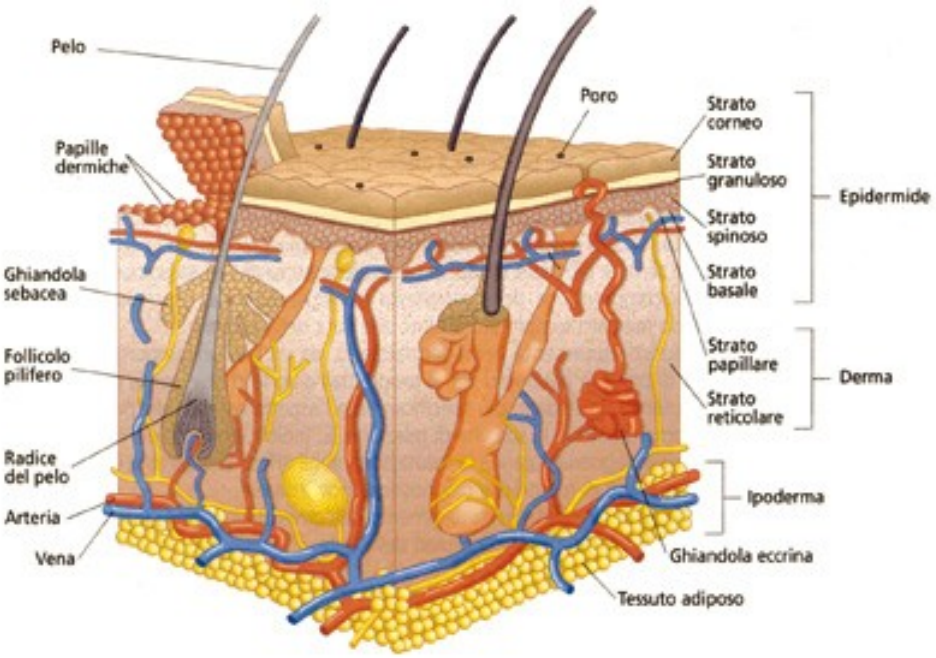


Figura 1.5: *Organizzazione anatomica della cute*

L'apparato tegumentario ricopre l'intera superficie corporea e svolge numerose funzioni, che sono: regolazione della temperatura corporea, funzione protettiva, sensoriale e difesa immunitaria. Ha due elementi principali: la *pelle*, che è costituita da epidermide e derma e poi gli *annessi cutanei* rappresentati da peli, unghie e varie ghiandole esocrine multicellulari [4]. La *cute* o pelle è il rivestimento più esterno del corpo di un vertebrato. Nei mammiferi e in particolare nell'uomo, è l'organo più esteso dell'apparato tegumentario e protegge i tessuti sottostanti (muscoli, ossa, organi interni). Prevalentemente è liscia a livello macroscopico, ma presenta delle irregolarità naturali, che a volte possono presentare natura patologica. Essa è formata da due strati: l'*epidermide* (strato più superficiale) e il *derma* (strato più in profondità). Lo strato di tessuto connettivo che unisce l'epidermide al derma è chiamato *strato basale*. Qui gli elementi sono di forma chiaramente cubica (prismatici) dotati di tonofilamenti e tonofibrille. Il processo di mitosi che si svolge in questo strato è simile a quello che si svolge nei tubuli seminiferi del didimo: per mitosi si producono due cellule una delle quali diverrà lamella cornea muovendosi verso la superficie, mentre l'altra resterà in loco per dare vita in futuro ad altre cellule.
Il derma è suddiviso in due strati: lo strato papillare e lo strato reticolare. Lo strato papillare è lo strato più superficiale del derma, che prende rapporto con la struttura dell'epidermide; la parte dermica della struttura giunzionale è detta *papilla dermica*

(una sporgenza conica rivolta verso l'epidermide).
Le papille dermiche sono più alte nella pelle dei polpastrelli delle dita e questo è un fattore determinante per la formazione delle impronte digitali. Infatti proprio in corrispondenza di queste sporgenze coniche, sulla superficie più esterna della cute si formano dei rilievi, che danno origine ad un'alternanza di creste e solchi chiamata *dermatoglifo*. Un'impronta digitale non è altro che la traccia lasciata da un dermatoglifo. Spesso vengono associate le impronte digitali ai polpastrelli, ma in realtà tutto il palmo della mano è in grado di lasciarne: nel caso in cui si abbia l'impronta di tutta la parte interna della mano, ci si riferisce con i termini di impronta palmare (*palmprint*). Analogamente vale anche per l'epidermide della parte del piede a contatto con il suolo, per cui si parla di impronta del piede (*footprint*) [2] .
Le impronte digitali sono completamente formate nel feto a circa 7 mesi di gestazione e fanno parte del fenotipo di un individuo, cioè l'insieme delle caratteristiche morfologiche e funzionali di un organismo. Lo sviluppo di un fenotipo è determinato in modo univoco dalla combinazione di un genotipo specifico e di un ambiente specifico. La micro- diversità delle condizioni ambientali nelle immediate vicinanze di ciascun polpastrello caratterizza la formazione dei più minuti dettagli della loro superficie. Anche due gemelli monozigoti, pur possedendo lo stesso codice genetico, hanno impronte digitali diverse. La principale funzione delle impronte digitali sull'epidermide è di aumentare l'attrito quando si afferrano gli oggetti.
Il processo con cui si sviluppano le increspature della pelle avviene attraverso la formazione di piccole isole attorno ai pori, che piano piano si sviluppano sino a formare le creste. Le impronte digitali restano immutate nel corso del tempo, dalla nascita di un individuo fino alla sua morte. Con l'avanzamento dell'età la pelle subisce un cambiamento, ma la struttura che interessa per la classificazione resta invariata. Le increspature si danneggiano quando l'epidermide è ferito, ma questa mutazione è temporanea, infatti l'epidermide si rigenera completamente. Se è danneggiato anche il derma, allora si formerà una cicatrice e il danno diviene permanente; la lesione è però localizzata solo sulla cicatrice, mentre la restante struttura resta intatta.

Analizzando la struttura delle impronte digitali è facile osservare che sono costituite da un flusso orientato di creste ("*ridge lines*") e valli ("*valleys*"), che formano un disegno complesso chiamato "*ridge pattern*" [1] .

Figura 1.6: *Ridge Pattern*

Studiando la struttura di un'impronta ad un *livello globale* si può notare come le creste assumano forme particolari caratterizzate da elevate curvature. Queste regioni, chiamate *singolarità* o *punti singolari*, possono essere classificate in tre tipi: cicli ("*loop*"), delta e spirali ("*whorl*"). Queste regioni sono caratterizzate da forme specifiche, rispettivamente simili a $\cap$, Δ e O.

Figura 1.7: *Singolarità*

Si definisce:

- *Punto di core*: approssimativamente il centro di uno schema di creste. Di punti di core ce ne può essere al massimo uno. Nella pratica consiste nel punto più estremo della curvatura della cresta più interna che forma la spirale. Le creste di un loop sono tutte a forma di una "u" ***(Figura 1.8)***;
- *Punto di delta*: definito come il punto su una cresta e coincidente o il più vicino possibile ad un punto di divergenza: quest'ultimo individua un punto in cui due creste, che correvano pressoché parallele, si dividono. Il delta viene definito anche come il punto al centro di una regione triangolare, che si trova normalmente in basso a destra o sinistra, nella quale confluiscono creste da direzioni diverse.

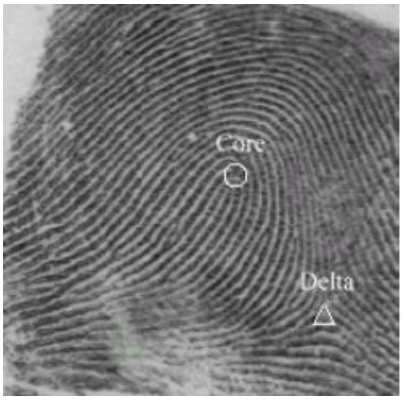


Figura 1.8: *Punti di Delta e Core*

Inoltre le singolarità di tipo whorl possono essere pensate come singolarità degeneri, formate dalla sovrapposizione di due loop.

A un *livello locale* è possibile individuare altre importanti caratteristiche, chiamate minuzie ("*minutiae*"), che rappresentano delle discontinuità legate alla struttura creste/valli. Nel 1892 Sir Francis Galton classificò le minuzie e osservò che esse rimangono identiche per tutta la vita di un individuo e per questo rappresentano un tratto distintivo delle impronte digitali e sono, infatti, le caratteristiche maggiormente utilizzate nel riconoscimento delle impronte.

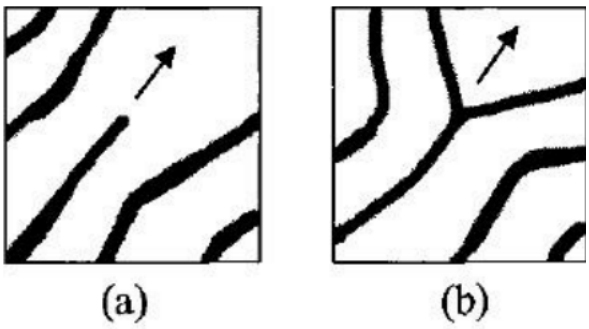


Figura 1.9: Minutiae a) Terminazione b) Biforcazione

Sono state individuate ben sette principali tipologie di minuzie, ma in applicazioni pratiche è piuttosto difficile capire la differenza tra un tipo e un altro. Per questo motivo l'FBI attualmente considera solo due classi di minuzie: le *terminazioni* e le *biforcazioni*. Le terminazioni sono i punti in cui una ridge line s'interrompe bruscamente, mentre le biforcazioni sono i punti in cui una ridge line si divide in due o più rami. Le biforcazioni e le terminazioni sono minuzie *duali*, cioè ad una biforcazione di una valle corrisponde una terminazione di una cresta e viceversa. Il numero di minuzie, la loro posizione e il loro orientamento rappresentano una caratteristica altamente distintiva che consente di distinguere un'impronta da un'altra.
Se l'immagine di un'impronta viene acquisita ad alta risoluzione (1000 dpi) è possibile identificare i pori della pelle che appaiono come piccoli puntini in ogni cresta. Il numero, la posizione e la forma di questi pori sono caratteristiche molto distintive e alcuni sistemi di riconoscimento sono basati proprio sul confronto di queste peculiarità.

1.3.3 Classificazione delle impronte digitali

Occorre moltissimo tempo per effettuare l'identificazione di un'impronta digitale su un vasto database. Perciò gli studiosi hanno cercato nuovi metodi di classificazione per le impronte in modo da comparare solo impronte della stessa classe.
Nel 1900 Edward Henry raffinò un precedente studio di Galton, introducendo uno schema di classificazione che tuttora viene adottato in tutto il mondo.
Sulla base del tipo e della posizione dei punti singolari, il sistema di Galton-Henry divide le impronte in cinque classi:

- *Arch*: In questi schemi le creste corrono da una parte all'altra dell'immagine senza formare curve a "u". Sono caratterizzati dall'assenza di punti di delta, ma qualora ve ne fosse una parvenza basta controllare che nessuna cresta "ricurvante" intersechi il segmento tra delta e core. Si possono distinguere due tipi di impronte ad arco:

  - *plain arch*: le creste tendono a fluire attraverso l'immagine con curvature molto smussate.
  - *tented arch*: qui sono presenti curvature significativamente strette che conferiscono all'immagine la forma di una tenda. Sono impronte ad arco triangolare, in cui le creste formano un angolo o una piega al centro con la presenza di un delta al di sotto di un loop.

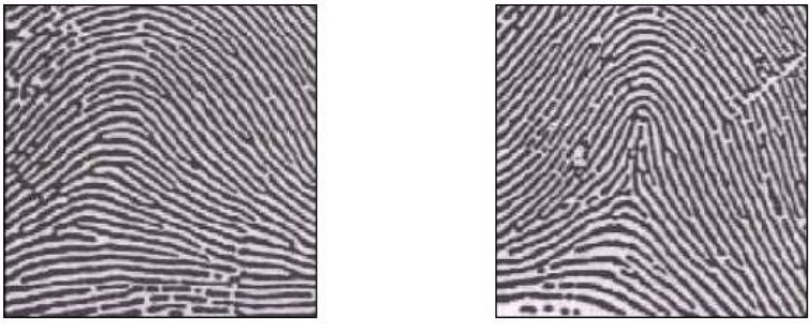


Figura 1.10: *Esempi di plain Arch e Tented Arch in un'impronta digitale*

- *Loop*: Sono caratterizzati da più creste che attraversano la linea immaginaria tra punto di core e punto di delta e formando una "u", ritornano approssimativamente nella direzione da cui sono venute; i loop sono infatti caratterizzati dall'avere esattamente un punto di loop e uno di core. I pattern di loop si possono a loro volta suddividere in destri e sinistri, o più correttamente in *"ulnar loop"* e

"radial loop" se si conosce anche la mano di cui fanno parte. Nel loop sinistro, le creste che lo formano provengono e ritornano verso la direzione sinistra, perciò, se il dito a cui appartengono si trova nella mano destra, tendono verso la direzione del pollice e quindi verso l'osso del radio, mentre se appartenessero alla mano sinistra, si affaccerebbero verso il mignolo e quindi verso l'ulna. Se non si conosce la mano è comunque sufficiente osservare l'impronta per stabilire semplicemente se il loop è destro o sinistro.
Per quanto riguarda i radial loop, rappresentano quelli meno comuni e si trovano solitamente sul dito indice.

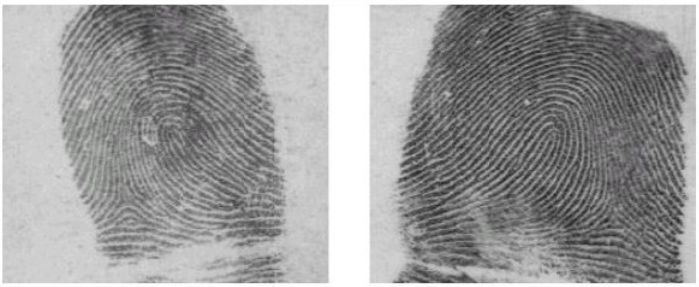


Figura 1.11: *Esempi di Right Loop e Left Loop*

- *Whorl*: le impronte sono rappresentate da una figura chiusa o spirale (come la sovrapposizione di due loop) e due delta. Come i loop, anche i whorl si possono suddividere in ulteriori sottoclassi:

  - *Plain loop.*
  - *Central pocket.*
  - *Double loop.*
  - *Accidental.*

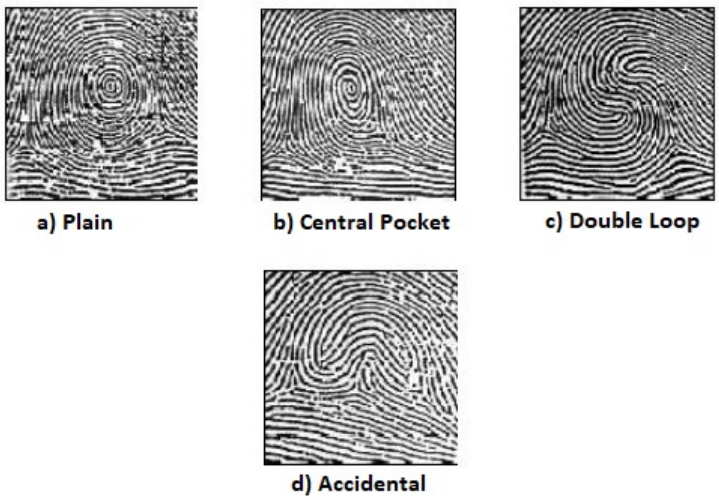


Figura 1.12: Esempi di Whorl

Se l'immagine contiene più di due delta allora è un *"accidental loop"*, se invece vi sono due delta, allora ricadono negli altri tre gruppi. Per quanto riguarda i primi due, tracciando un'ideale linea tra i due punti di delta, si osserva se questa interseca l'area delle creste che formano un cerchio: se la risposta è positiva allora si tratta di un *"central pocket loop"*, se negativa, di un *"plain loop"*. Per quanto riguarda invece il *"double loop whorl"*, la zona circolare è formata da due completi e distinti loop che "girano" uno dentro l'altro.

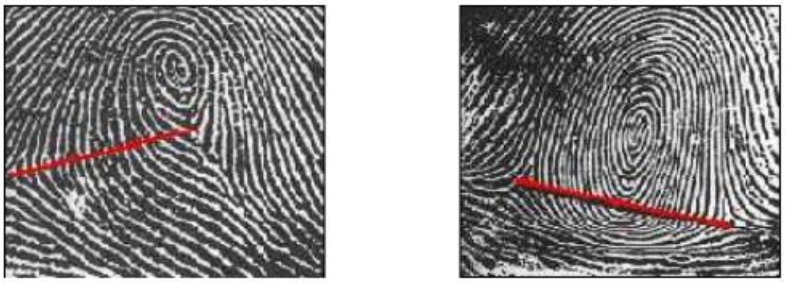


Figura 1.13: Nella prima figura si può osservare un Central pocket loop, nella seconda figura un Plain loop

Si può osservare che la zona a forma di delta presente negli arch in generale, non individua un punto di delta, dato che è formata dalle stesse creste ricurvanti: in un loop tale punto è creato invece da due flussi di creste ben distinti uno dall'altro.
Le cinque classi qui menzionate whorl, right loop, left loop, arch (o plain arch), e tented arch, vengono utilizzate dal *National Institute of Standards and Technology* (NIST), come *benchmark* per i propri sistemi di classificazione automatica, dei quali rientra per esempio il programma Pcasys, un prototipo dimostrativo di cui è disponibile anche il codice sorgente (pubblico dominio) e basato su reti neurali in uno dei passi di computazione che esegue.
Si può osservare come in ognuna delle cinque categorie di impronte digitali il numero dei loop coincida con il numero dei delta. Questo è conseguenza di un caso particolare del teorema di Hopf-Poincaré e del fatto che vicino al bordo dell'immagine si può assumere che la direzione delle ridge line sia costante.
Tale supposizione è ragionevole, infatti, osservando una qualunque impronta digitale possiamo notare che ai margini dell'immagine, le ridge line sono sostanzialmente orizzontali, salvo un inarcamento più o meno accentuato verso l'alto soprattutto nella parte superiore. Questa prima classificazione delle impronte è utile per rendere più veloce il processo di identificazione di un'impronta da parte di un sistema automatico. Comparando solo impronte della stessa classe infatti, i sistemi automatici riescono a lavorare in tempi accettabili anche su una grande mole di dati.

Capitolo 2

Triangolazione di Delaunay e Diagramma di Voronoi

2.1 Introduzione

In "*Le monde de M. Descartes et le traitè de la lumière*", pubblicato nel 1644, Cartesio usava diagrammi, come quello di Voronoi, per mostrare la decomposizione dello spazio in regioni convesse, ognuna delle quali era composta di materia che si avvolgeva intorno ad una stella; affermava infatti che il sistema solare è fatto di vortici.
Tuttavia, la prima presentazione del concetto del diagramma di Voronoi apparve nel lavoro di Dirichlet (1850) e prese il nome dal matematico russo Georgy Fedoseevich Voronoi (o Voronoy), che nel 1908 definì e studiò il caso generale n-dimensionale. Questo concetto continuò a svilupparsi successivamente e si dimostrò subito utile in vari campi della scienza.
Una prima estensione del diagramma di Voronoi fu in cristallografia, dove una serie di punti veniva posizionata regolarmente nello spazio e le celle di Voronoi erano etichettate come Wirkungsberich (*area di influenza*) (Niggli, 1927). Thiessen (1911) utilizzò le celle di Voronoi, dette *poligoni di Thiessen*, per calcolare stime accurate riguardo le medie regionali delle precipitazioni.
Un'altra estensione del diagramma di Voronoi è stata vista nelle scienze naturali e sociali per lo studio delle aree di mercato (Bogue 1949). Il diagramma di Voronoi in geografia viene utilizzato per analizzare i modelli di punti 2D (Boots, 1974). Fu solo nei primi anni '70 che furono sviluppati un certo numero di algoritmi per la costruzione efficiente del diagramma di Voronoi. Ciò ha motivato ulteriori sviluppi in diverse aree usando l'informatica e la geometria computazionale.

Varie applicazioni dei diagrammi di Voronoi si stanno espandendo dalle scienze naturali all'ingegneria e dalle scienze mediche alle scienze sociali.
Esistono differenti nomi associati a questo stesso concetto a seconda del campo in cui sono utilizzati: *trasformazioni dell'asse mediale* in biologia e fisiologia, *zone di Weigner- Seitz* in chimica e fisica, *domini d'azione* in cristallografia e *poligoni di Thiessen* in meteorologia e geografia.
Dirichelet e Voronoi introdussero formalmente il concetto, che prese il nome di *Diagramma di Voronoi*.
Quest'ultimo fu il primo a considerare il duale di questa struttura, ma fu Delaunay a stabilirne le proprietà. Il duale del diagramma di Voronoi prese così il nome di *triangolazione di Delaunay*. Con il tempo, gli algoritmi per la costruzione di quest'ultima struttura si dimostrarono più veloci e più utili nelle applicazioni. Il metodo ad elementi finiti, richiede la divisione dello spazio in elementi semplici, come per esempio i triangoli in due dimensioni e i tetraedri in tre dimensioni.
In particolare, la triangolazione di Delaunay ha delle proprietà specifiche che la rendono preferibile ad altri tipi di triangolazioni [5].

2.2 Alcune nozioni di base

Innanzitutto vediamo alcune nozioni e definizioni generali che riguardano triangoli e tetraedri. Successivamente vedremo alcune nozioni di base riguardanti le triangolazioni, triangolazioni conformi e la triangolazioni di Delaunay [6].

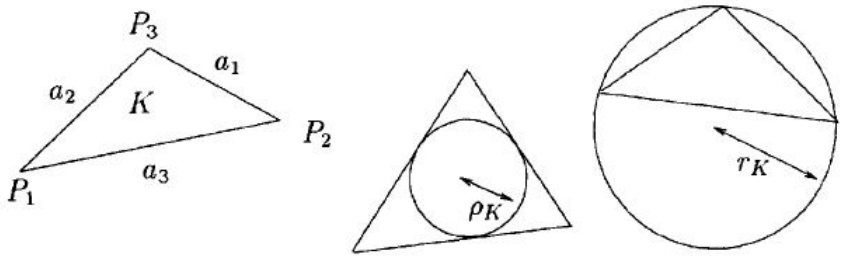


Figura 2.1: *Triangolo con vertici e lati numerati, incerchio, circumcerchio*

2.2.1 Il triangolo

Definizione elementare. Un triangolo è un poligono con tre lati. Esso può quindi essere rappresentato da una terna ordinata di vertici, detti P_i, dati in senso orario o antiorario:

$$K = (P_1, P_2, P_3)$$

Ci sono sei modi (o permutazioni) per esprimere la terna che definisce un triangolo. Nel caso in cui sia definito un orientamento, sono rilevanti solo tre permutazioni. Quindi, per un triangolo in un piano, l'orientamento è implicitamente definito (usando la normale del piano) e le tre possibili definizioni implicano che la sua superficie, indicata con S_k è un'area "con segno" ed è data da:

$$S_k = \frac{1}{2} \cdot \begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix}$$

o

$$S_k = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Dove x_i y_i sono le coordinate dei vertici P_i con $i = 1,2,3$ e $|.|$ indica il determinante della matrice.
Inoltre, questa definizione ci consente di definire esplicitamente i lati (o bordi) di un dato triangolo. Infatti l'i-esimo lato di un triangolo è indicato da a_i, con $i = 1,2,3$ ed è il lato che congiunge il vertice P_{i+1} con il vertice P_{i+2} , dove $P_j = P_{j-3}$, se $j > 3$.

2.2.2 Il Circumcerchio

Questo cerchio è completamente descritto dal centro, *circumcentro*, e dal raggio, *circumraggio*.
Esistono due modi per calcolarlo: risolvendo un sistema lineare o usando l'equazione del cerchio.
L'equazione del circumcerchio è data da:

$$\Delta_k(x, y) = 0$$

Con

$$\Delta_k(x,y) = \begin{vmatrix} l_1^2 - l^2 & l_2^2 - l_1^2 & l_3^2 - l_1^2 \\ x_1 - x & x_2 - x_1 & x_3 - x_1 \\ y_1 - y & y_2 - y_1 & y_3 - y_1 \end{vmatrix}$$

Dove $l^2 = x^2 + y^2$ e $l_i^2 = x_i^2 + y_i^2$ con $i = 1,2,3$.
Inoltre, espandendo questo determinante, le coordinate x_C e y_C del circumcentro sono ottenute esplicitamente, così come r_K, il circumraggio.

2.2.3 L'incerchio

L'incerchio è un cerchio inscritto ad un triangolo. Il suo raggio, l'*inraggio* è definito da:

$$\rho_K = \frac{S_K}{p_K}$$

dove p_K è il semiperimetro del triangolo K.

2.2.4 Il tetraedro

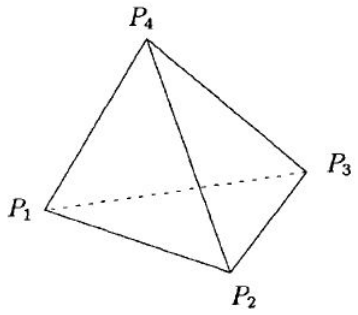


Figura 2.2: Tetraedro

Il tetraedro è un poliedro con quattro facce triangolari. Esso è definito da una quadrupla, lista ordinata di 4 vertici P_i:

$$K = (P_1, P_2, P_3, P_4)$$

Se assumiamo che le facce del tetraedro siano orientate, allora anche le normali a queste facce saranno orientate.

Il Volume, V_k,sarà dotato di "segno" e sarà dato da:

$$V_k = \frac{1}{6} \cdot \begin{vmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{vmatrix}$$

o

$$V_k = \frac{1}{6} \cdot \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix}$$

dove x_i, y_i, z_i sono le coordinate dei vertici P_i dell'elemento K.

2.2.5 La Circumsfera

Ad ogni tetraedro è associata la sua *circumsfera*, cioè la sfera circoscritta ai suoi vertici. Il raggio, o *circumraggio*, e il centro, o *circumcentro*, di questa sfera possono essere ottenuti analogamente al caso bidimensionale usando la seguente equazione:

$$\Delta_k(x, y, z) = 0$$

Con

$$\Delta_k(x, y, z) = \begin{vmatrix} l_1^2 - l^2 & l_2^2 - l_1^2 & l_3^2 - l_1^2 & l_4^2 - l_1^2 \\ x_1 - x & x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_1 - y & y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_1 - z & z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{vmatrix}$$

Dove $l^2 = x^2 + y^2 + z^2$ e $l_i^2 = x_i^2 + y_i^2 + z_i^2$

Oppure possono essere ottenuti risolvendo un sistema di equazioni lineari per trovare il *circumcentro*.
Per calcolare il *circumraggio* si usa la seguente formula:

$$r_k = \frac{\sqrt{(a+b+c)(a+b-c)(b+c-a)(a-b+c)}}{24 \cdot V_k}$$

dove a, b e c sono i prodotti delle lunghezze di due spigoli opposti.

2.2.6 L'insfera

La sfera inscritta al tetraedro è detta anche insfera. Il raggio dell'insfera, o inraggio, è dato da:

$$\rho_k = \frac{3_K}{S_1 + S_2 + S_3}$$

dove S_i è la superficie della faccia i dell'elemento.

Osservazione: In due dimensioni, un triangolo con un cerchio circoscritto di raggio limitato, la cui superficie è quasi zero e i cui bordi hanno una lunghezza non degenere, non esiste, vedi ***Figura 2.3***. Tuttavia, in tre dimensioni, un tetraedro il cui volume è piccolo quanto vogliamo, ma il cui circumraggio è limitato e i cui bordi godono di una lunghezza ragionevole esiste. Un tale elemento, il cosiddetto "*frammento*", è una tipica entità di meshes tetraedriche tridimensionali (tuttavia, come si vedrà, un tale frammento può essere un elemento di Delaunay).

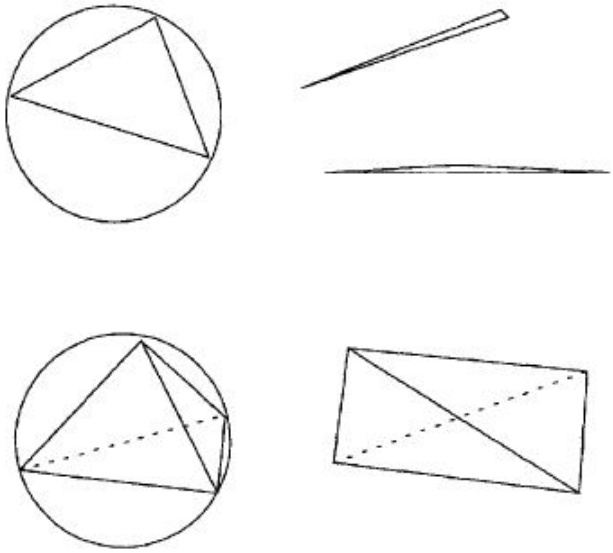


Figura 2.3: A destra triangoli e tetraedri degeneri. A sinistra elementi ben-modellati

2.3 Il simplesso

Definizione 2.3.1. *Sia $\mathbb{R}$ uno spazio di dimensione d. Sia S un insieme di punti in $\mathbb{R}^d$. Se $S = A_1, \ldots, A_{n,\ldots}$, allora:*

$$\sum_{i=0}^{n} \lambda_i A_i$$

rappresenta una combinazione lineare dei punti in S.

Definizione 2.3.2. *Questa combinazione lineare degli n elementi di S, per $\sum_{i=0}^{n} \lambda_i = 1$ definisce un sottospazio di $\mathbb{R}^d$, detta combinazione affine degli A_i.*
Se, per ogni i, $\lambda_i \geq 0$, allora quella combinazione è detta combinazione convessa.

Definizione 2.3.3. *La chiusura convessa di S, detta Conv(S), è il sottoinsieme di $\mathbb{R}^d$ generato da tutte le combinazioni lineari convesse dei membri di S.*
Questo insieme è il più piccolo insieme convesso che contiene l'insieme S.

Definizione 2.3.4. *La chiusura convessa, di un numero finito di punti in $\mathbb{R}^d$ è detta politopo ed è un insieme chiuso e limitato di $\mathbb{R}^d$. Un politopo in dimensione k si dice k-politopo, la cui dimensione affine è ancora in dimensione k.*

Definizione 2.3.5. *La chiusura convessa di $k+1$ punti, con kd, non in uno spazio affine di dimensione $k-1$, è un particolare k-politopo detto simplesso o k-simplesso.*
Perciò in due dimensioni, un 2-simplesso è un triangolo, mentre in tre dimensioni, un 3-simplesso è un tetraedro.

2.4 La triangolazione

Definizione 2.4.1. *Un insieme di punti S in $\mathbb{R}^d$ ($d = 2$ o $d = 3$) è in posizione generale, in due dimensioni, se non esistono tre punti collineari o quattro punti cociclici, cioè che appartengono alla medesima circonferenza.*
In tre dimensioni significa che non devono esistere quattro punti coplanari o cinque punti che appartengono alla stessa sfera.

Definizione 2.4.2. *Sia S un insieme di punti in posizione generale e sia Ω il dominio in $\mathbb{R}^d$ definito da Conv(S). Se K è un simplesso (triangolo o tetraedro secondo la posizione di d) allora τ_r è un ricoprimento simpliciale di $\Omega = Conv(S)$ se valgono le seguenti condizioni:*

- *(H_0) L'insieme dei vertici degli elementi di τ_r è esattamente S*
- *(H_1) $\Omega = \cup_{K \in \tau_r} K$*
- *(H_2) Ogni elemento di K in τ_r è non vuoto*
- *(H_3) L'intersezione degli elementi, presi due a due, è un insieme vuoto.*

La condizione (H_2) non è strettamente necessaria per definire un ricoprimento, ma in questo contesto è utile e quindi sarà assunta.

Definizione 2.4.3. *τ_r è una triangolazione conforme o semplicemente triangolazione di Ω se τ_r è un ricoprimento simpliciale e se inoltre vale la seguente condizione:*

- *(H_4) l'intersezione degli elementi di τ_r è:*

- *un insieme vuoto*

- *un vertice*

- *uno spigolo*

- *una faccia (se d=3)*

Generalmente, in dimensione d, l'intersezione può essere al più una k-faccia, per $k = 1, ..., d-1$.

Le triangolazioni e più specificamente, la *triangolazione di Delaunay*, sono state studiate per lungo tempo, in particolare nella *geometria computazionale*.
La geometria computazionale è emersa dal campo della progettazione e dell'analisi degli algoritmi alla fine degli anni '70.
Il successo di tale disciplina può da un lato essere spiegato grazie alla bellezza dei problemi studiati ed alle relative soluzioni ottenute, dall'altro, dai molti domini applicativi: grafica computerizzata, sistemi di informazione geografica (GIS), robotica ed altri, in cui gli algoritmi geometrici giocano un ruolo fondamentale [8] .

Gli studi sulle triangolazioni risalgono già al tempo di Euclide (330-270 a.C.), ma i padri dello sviluppo moderno delle triangolazioni sono Dirichlet (1805-1859), Voronoi (1868-1908) e Delaunay (1890-1980).

2.4.1 Diagramma di Voronoi

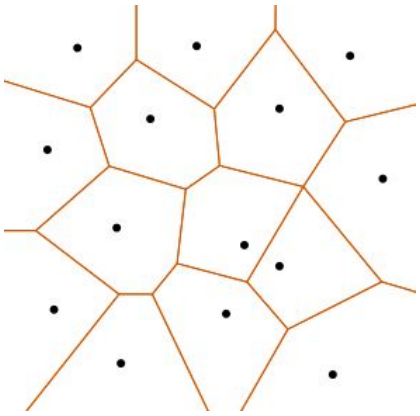


Figura 2.4: Diagramma di Voronoi in dimensione 2

Definizione 2.4.4. *La distanza euclidea tra due punti p e q è denotata come dist(p,q):*

$$dist(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Definizione 2.4.5. *Sia $P = p_1, \ldots, p_n$ un insieme di n punti distinti nel piano. Il diagramma di Voronoi di P (Vor(P)) è definito come la suddivisione del piano in n celle, una per ogni punto in P, con la proprietà che un punto q giace nella cella corrispondente a un punto p_i se e solo se $dist(q,p_i) < dist(q,p_j)$, per ogni $p_j \in P$, con $j \neq i$.*

Diamo uno sguardo più da vicino al diagramma di Voronoi. Innanzitutto studiamo la struttura di una singola cella di Voronoi [7].

Definizione 2.4.6. *Siano p,q due punti nel piano, definisco la bisettrice di p e q come la bisettrice perpendicolare del segmento $\overline{pq}$. Tale bisettrice suddivide il piano in due semipiani. Sia h(p,q) il semipiano aperto contenente p e h(q,p) il semipiano aperto contenete q. Si noti che $r \in h(p,q)$ se e solo se $dist(r,p) < dist(r,q)$.*
Da ciò otteniamo che la cella di Voronoi corrispondente ad un punto o sito p_i, denotata con $V(p_i)$ è :

$$V(p_i) = \bigcap_{1 \leq j \leq n} h(p_i, p_j), \quad con\ j \neq i$$

Allora $V(p_i)$ è l'intersezione di n-1 semipiani, e quindi una regione poligonale convessa aperta è limitata da al più n-1 vertici e n-1 spigoli.

Ogni cella di Voronoi è l'intersezione di un numero di semipiani, perciò il diagramma di Voronoi è una suddivisione del piano in cui i bordi sono rettilinei.

Le celle di Voronoi sono poligoni (poliedri in tre dimensioni) chiusi e convessi. Le celle aperte e disgiunte ricoprono tutto lo spazio e costituiscono, nel piano (se d=2), la *tassellazione di Dirichlet* o più in generale il *diagramma di Voronoi* in $\mathbb{R}^d$.
Ogni cella di Voronoi è associata ad un punto e che in due dimensioni ogni cella ha il bordo costituito alternativamente da lati di Voronoi (celle 1-dimensionali) e vertici di Voronoi (celle 0-dimensionali); in tre dimensioni avremo anche facce di Voronoi (celle 2-dimensionali).

Teorema 2.4.7. *Sia S un insieme di n punti nel piano. Se tutti i punti sono collineari allora Vor(P) è costituito da n-1 rette parallele. Altrimenti Vor(P) è connesso e i suoi bordi sono segmenti o semirette.*

Dimostrazione. La prima parte del teorema è facile da provare, quindi supponiamo che non tutti i siti in P siano collineari.
Per prima cosa mostriamo che i bordi di Vor(P) sono segmenti o semirette.
Sappiamo che i bordi di Vor(P) sono parti di linee rette, cioè parti di bisettrici tra due punti.
Supponiamo per assurdo che ci sia un bordo e di Vor(P) che è una linea piena. Sia e sul bordo delle celle di Voronoi $V(p_i)$ e $V(p_j)$. Sia $p_k \in P$ un punto che non è collineare con p_i e p_j. La bisettrice di p_j e p_k non è parallela ad e , quindi interseca e. Ma allora la parte di e che si trova all'interno di $h(p_k, p_j)$ non può essere sul bordo di $V(p_j)$ perché è più vicino a p_k che a p_j, siamo giunti ad una contraddizione.
Resta da provare che Vor(P) è connesso.
Se così non fosse, allora ci sarebbe una cella $V(p_i)$ che divide il piano in due. Poiché le celle di Voronoi sono convesse, $V(p_i)$ sarebbe costituita da una striscia delimitata da due linee continue parallele. Ma abbiamo appena dimostrato che i bordi del diagramma di Voronoi non possono essere linee piene, si ha perciò una contraddizione. □

Analizziamo ora la complessità del diagramma di Voronoi, cioè il numero totale di vertici e spigoli. Dato che ci sono n punti e ogni cella ha al più n-1 vertici e bordi, la complessità di Vor(P) è al massimo quadratica. Nel dettaglio si ha il seguente teorema:

Teorema 2.4.8. *Per $n \geqslant 3$, il numero di vertici nel diagramma di Voronoi di un insieme in n punti siti nel piano è al più $2n-5$ e il numero di bordi è al massimo $3n-6$.*

Dimostrazione. Se i punti sono collineari, il teorema segue immediatamente dal teorema precedente.
Assumiamo quindi che non sia questo il caso. Dimostriamo il teorema utilizzando la formula di Eulero, la quale afferma che per qualsiasi grafo planare connesso, con m_v nodi, m_e archi, ed m_f facce, si ha la seguente relazione:

$$m_v + m_e + m_f = 2$$

Non possiamo applicare direttamente la formula di Eulero a Vor(P) perché Vor(P) ha bordi semi-infiniti e quindi non è un grafo corretto. Per rimediare alla situazione aggiungiamo un vertice extra v^∞ "all'infinito" all'insieme di vertici e colleghiamo tutti i bordi semi-infiniti di Vor(P) a questo vertice. Ora abbiamo un grafo planare connesso al quale possiamo applicare la formula di Eulero.
Otteniamo la seguente relazione tra n_v, numero di vertici di Vor(P), n_e, numero dei bordi di Vor(P) ed n, numero dei punti:

$$(n_v + 1) - n_e + n = 2$$

Inoltre, ogni bordo del grafo ha esattamente due vertici, quindi se sommiamo i gradi di tutti i vertici otteniamo il doppio del numero dei bordi. Perché ogni vertice, incluso v^∞, ha grado almeno tre. Si ha:

$$2n_e \geqslant 3 \cdot (n_v + 1$$

Unendo le ultime due equazioni si ha la tesi del teorema.

□

Possiamo ora osservare un'altra caratterizzazione dei vertici e bordi del diagramma di Voronoi.

Teorema 2.4.9. *Per il diagramma di Voronoi Vor(P) di un insieme di punti P vale quanto segue:*

i. *Un punto q è un vertice di Vor(P) se e solo se il suo più grande cerchio vuoto $C_p(q)$ contiene tre o più punti sulla frontiera.*

ii. *La bisettrice tra i punti p_i e p_j definisce un bordo di Vor(P) se e solo se vi è un punto q sulla bisettrice tale che $C_p(q)$ contenga sia p_i che p_j sul suo bordo, ma nessun altro punto.*

Dimostrazione. Dimostriamo il punto (i):
Supponiamo ci sia un punto q tale che $C_p(q)$ contenga tre o più punti sul bordo. Siano p_i,p_j e p_k tre di questi punti. Poiché l'interno di $C_p(q)$ è vuoto, q deve trovarsi sul bordo di ciascuna delle celle V(p_i), V(p_j) e V(p_k), e q deve essere un vertice di V(p).
D'altra parte, ogni vertice q di V(P) è incidente ad almeno tre bordi e, quindi, ad almeno tre celle Voronoi V(p_i),V(p_j) e V(p_k). Il vertice q deve essere equidistante a p_i, p_j e p_k e non può esserci un altro punto più vicino a q, poiché V(p_i), V(p_j) e V(p_k) non si incontrerebbero in q.
Quindi l'interno del cerchio con p_i, p_j e p_k sul suo bordo, non contiene alcun punto.

Dimostriamo il punto (ii):
Supponiamo ci sia un punto q con le proprietà dichiarate nel teorema.
Quindi $C_p(q)$ non contiene nessun punto al suo interno e p_i e $p_j$sono sul suo bordo, abbiamo: $dist(q,p_i) = dist(q,p_j) \leqslant dist(q,p_k)$, $\forall\ 1 \leqslant k \leqslant n$.
Ne segue che q giace sul bordo o vertice di Vor(P).
La prima parte del teorema implica che q non può essere un vertice di Vor(P).Quindi q giace sul bordo di Vor(P), che è definito dalla bisettrice di p_i e p_j. Al contrario la bisettrice di p_i e p_j definisce un bordo di Voronoi. Il cerchio vuoto più grande di qualsiasi punto q all'interno di questo bordo deve contenere p_i e p_j sul suo bordo e nessun altro punto. □

2.4.2 Triangolazione di Delaunay

Figura 2.5: B.N. Delaunay

Tra i possibili tipi di triangolazioni, focalizzeremo l'attenzione sulla triangolazione di Delaunay [6] .

Definizione 2.4.10. ***(Criterio di Delaunay)***
Sia S un insieme di punti.
τ_r *è una triangolazione di Delaunay di* $\Omega = Conv(S)$ *se il circumcerchio o la circumsfera (aperta) associata ai suoi elementi è vuota.*
Questo criterio, detto anche criterio della sfera vuota, ci dice che la boccia aperta associata agli elementi non deve contenere nessun vertice (mentre la boccia chiusa contiene solo i vertici dell'elemento in considerazione). Questo è ciò che caratterizza la triangolazione di Delaunay.

Questa proprietà porta a molte altre caratteristiche di cui gode una triangolazione di Delaunay.

Alcune proprietà in dimensione due
Tra le diverse possibili triangolazioni di S, la triangolazione di Delaunay:

- massimizza il minimo angolo formato dagli spigoli della triangolazione

- minimizza il massimo circumraggio per ogni elemento

Di conseguenza, una triangolazione i cui angoli non sono ottusi è una triangolazione di Delaunay.

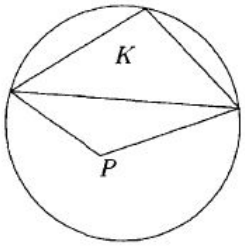


Figura 2.6: Se il "Criterio della sfera vuota" viene violato, il disco di K racchiude il punto p

Alcune proprietà in qualsiasi dimensione

- il massimo raggio della minima sfera associata agli elementi è minimo
- in una triangolazione di Delaunay, l'unione delle circumsfere associate agli elementi che condividono un punto interno è contenuta nella stessa unione associata allo stesso punto in ogni altro tipo di triangolazione
- la somma delle radici delle lunghezze degli spigoli pesati con la somma dei volumi degli elementi che condividono gli stessi spigoli è minima

Tuttavia, se tutti i simplessi in una triangolazione contengono il loro circumcentro, allora quella triangolazione è una *triangolazione di Delaunay* [6] .

Un problema molto importante è quello di assicurare l'esistenza, in una triangolazione, di un insieme di spigoli (in due dimensioni) o di facce (in tre dimensioni). Chiamiamo *Const* l'insieme di questi elementi.

Definizione 2.4.11. *τ_r è una triangolazione constrained di Ω, se gli elementi di Const sono anche elementi di τ_r. In particolare una triangolazione constrained può soddisfare il criterio di Delaunay localmente, eccetto che negli intorni degli spigoli e delle facce vincolate.*

Se si considera il duale del diagramma di Voronoi, si ottiene una triangolazione di Delaunay. Tale risultato è stato pubblicato da Delaunay nel "*Sur la sphère vide*" del 1984 ed è considerato uno dei più importanti risultati di Delaunay.

Sia P un insieme di n punti nel piano, ricordiamo che il diagramma di Voronoi di P, denotato con $Vor(P)$, è la suddivisione del piano in n regioni, una per ciascun

punto di P, tale che la regione con $p \in P$ contenga tutti i punti per i quali p è il sito più vicino.
La regione di un sito p è chiamata cella di Voronoi ed è indicata con V(P).
Ora vediamo più nel dettaglio il duale del grafo del diagramma di Voronoi.
Tale grafo G ha un nodo per ogni cella di Voronoi ed un arco tra due nodi se le celle di Voronoi corrispondenti condividono un bordo. Ciò significa che G ha un arco per ogni lato di Vor(P).
Considerando l'incorporamento in linea retta di G, dove il nodo corrispondente alla cella di Voronoi V(p) è il punto p e l'arco che collega i due nodi di V(p) e V(q) rappresenta il segmento $\overline{pq}$ si ottiene il *grafo di Delaunay* di P, denotato con DG(P).
In altre parole, unendo i vertici appartenenti a due celle adiacenti si ottiene la triangolazione. Si può osservare inoltre che le k celle di Voronoi sono ortogonali rispetto alle (d-k) facce della triangolazione di Delaunay.
Di seguito sono riportate alcune delle principali proprietà del grafo di Delaunay.

Teorema 2.4.12. *Il grafo di Delaunay di un insieme di punti planari è un grafo planare.*

Dimostrazione. Occorre innanzitutto definire una proprietà che riguarda i bordi del diagramma di Voronoi formulata in termini di grafi di Delaunay.

Osservazione. Il bordo $\overline{p_i p_j}$ si trova nel grafo di Delaunay DG(P) se e solo se esiste un disco chiuso C_{ij} con p_i e p_j sul suo bordo e nessun altro punto di P è contenuto in esso.

Sia t_{ij} un triangolo i cui vertici sono p_i,p_j ed il centro di C_{ij}. Notiamo che il lato di t_{ij} che connette p_i con il centro di C_{ij} è contenuto in $V(p_i)$; vale lo stesso per p_j.
Ora sia $\overline{p_k p_l}$ altri bordi di DG(P), e come prima definiamo il cerchio C_{kl} ed il triangolo t_{kl}.
Supponiamo per assurdo che $\overline{p_i p_j}$ e $\overline{p_k p_l}$ si intersechino. Sia p_k che p_l dovrebbero giacere fuori C_{ij} e quindi anche fuori di t_{ij} . Ciò implica che $\overline{p_k p_l}$ dovrebbe intersecare uno dei bordi di t_{ij} incidente al centro di C_{ij} dove interseca uno dei bordi di t_{kl} incidente al centro diC_{kl}. Ma questo contraddice il fatto che questi bordi siano contenuti in celle disgiunte di Voronoi. □

In particolare si può osservare che se i punti di P sono distribuiti casualmente, la possibilità che 4 punti si trovino su un cerchio è molto piccola [6] . Un insieme di punti è in posizione generale se non contiene quattro punti su un cerchio. Se P è in posizione generale, allora tutti i vertici del diagramma di Voronoi hanno grado tre, di conseguenza tutte le facce limitate di DG(P) sono triangoli. Perciò DG(P) è spesso definita come triangolazione di Delaunay di P.
Una triangolazione di Delaunay di P può essere definita come una qualsiasi triangolazione ottenuta aggiungendo bordi al grafo di Delaunay. Poichè tutte le facce di DG(P) sono convesse, ottenere una tale triangolazione è facile.
Si può osservare anche che la triangolazione di Delaunay è unica se e solo se DG(P) è una triangolazione, nel caso in cui P è in posizione generale.

Teorema 2.4.13. *Sia P un insieme di punti nel piano. Valgono le seguenti condizioni:*

(i) Tre punti $p_i, p_j, p_k \in P$ sono vertici della stessa faccia del grafo di Delaunay di P se e solo se il cerchio passante per p_i, p_j, p_k non contiene alcun punto di P al suo interno

(ii) Due punti $p_i, p_j \in P$ formano un bordo sul grafo di Delaunay di P se e solo se esiste un disco chiuso C che contiene p_i e p_j sul suo contorno e non contiene nessun altro punto di P.

Tale teorema implica la seguente caratterizzazione delle triangolazioni di Delaunay.

Teorema 2.4.14. *Sia P un insieme di punti nel piano e sia T una triangolazione di P. Allora T è una triangolazione di Delaunay di P se e solo se il circumcerchio di qualsiasi triangolo di T non contiene nessun punto di P al suo interno.*

Vediamo altre definizioni che riguardano la Triangolazione di Delaunay, prima di descrivere un importante risultato che è alla base dei più importanti algoritmi per la costruzione di triangolazioni di Delaunay (**Lemma 2.**)

Definizione 2.4.15. *Sia S un insieme di n punti e sia $P \subset S$ e $V(P) \neq \emptyset$ una cella di Voronoi. Si definisce una cella di Delaunay, T(P), la chiusura convessa di P.*

Definizione 2.4.16. *Una triangolazione di Delaunay T(P) è la collezione di tutte le celle di Delaunay T(P),* $\forall P \subset S$ *con* $V(P) \neq \emptyset$

Osservazione. Se i punti da triangolare sono in posizione generale la triangolazione di Delaunay è unica.
Se invece i punti dati includono d+2 punti cociclici (cosferici), il duale del diagramma di Voronoi è un unico ricoprimento dell'insieme di punti dato, che però include poligoni convessi diversi da simplessi.

Poichè nelle applicazioni si usano soprattutto triangoli e tetraedri, a partire dal ricoprimento, si può ottenere una triangolazione dividendo i poligoni in triangoli e i poliedri in tetraedri. E' importante sottolineare che sono possibili moltissime divisioni e quindi moltissime triangolazioni. Ad esempio nella ***Figura 2.7***, esistono due possibili modi per dividere il quadrilatero in due triangoli, ma una sola di queste rispetterà il criterio della sfera vuota.

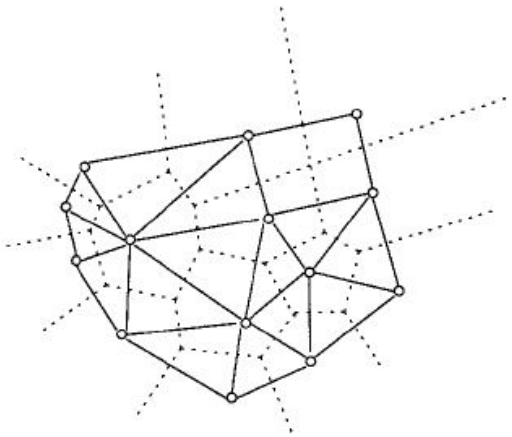


Figura 2.7: *Ricoprimento di Delaunay di un insieme con quattro punti cociclici*

Definizione 2.4.17. *Siano* $P_i e P_j$ *due siti in S e sia D la triangolazione di Delaunay di S. Il lato che congiunge* $P_i e P_j$ *è un lato di Delaunay se appartiene alla triangolazione T, cioè se esiste un cerchio vuoto (sfera vuota)che passa per* $P_i e P_j$*. Dove con cerchio vuoto (sfera vuota) si intende che non contiene nessun vertice di S al suo interno.*

Definizione 2.4.18. *Un elemento K di una triangolazione di Delaunay D è detto elemento di Delaunay.*

Lemma 1. Sia T una triangolazione. tutti gli elementi di T sono Delaunay ⇔ tutti i lati di T sono Delaunay.

Dimostrazione. Considero il caso bidimensionale. (⇒) Se tutti i triangoli di T sono Delaunay, allora il circumcerchio di ogni triangolo è vuoto. Poichè ogni lato di T appartiene ad un triangolo di T, ogni lato è contenuto in un cerchio vuoto e così è Delaunay [6] . (⇐) Supponiamo per assurdo che tutti i lati di T diano Delaunay e che esista un triangolo K che non sia Delaunay. Essendo T una triangolazione, K contiene solamente i suoi tre vertici e nessun altro sito, perciò almeno un sito P_i è contenuto nel circumcerchio di K, ma non nel triangolo stesso. Sia l_K il lato di K che divide P_i dall'interno di K e sia p_j il vertice di K opposto a l_K. Si può notare che è possibile tracciare un cerchio che contiene l_K, ma non contiene né P_i né P_j, il che significa che il lato l_K è Delaunay. Contraddizione. □

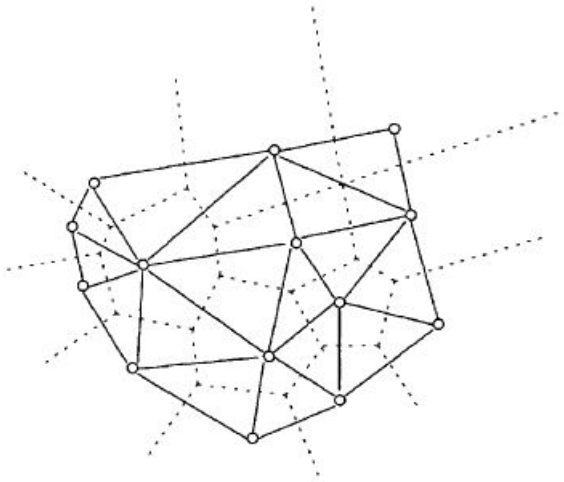


Figura 2.8: *Triangolazione di Delaunay*

Lemma 2. Sia T una triangolazione arbitraria della chiusura convessa di un insieme di punti S. Se per ogni coppia di simplessi adiacenti in T vale il criterio della sfera vuota, allora questo criterio vale globalmente e T è una triangolazione di Delaunay.

Dimostrazione. Innanzitutto si osserva che il criterio di Delaunay è simmetrico. Consideriamo due simplessi adiacenti che dividono una comune (d-1)-faccia e sia P(risp. Q) il vertice in questi simplessi opposti a questa faccia. Si ha:

$$Q \notin B_P \Leftrightarrow P \notin B_Q$$

dove B_P (risp. B_Q) è la boccia associata al simplesso che ha P (risp.Q) come vertice, vedi ***Figura 2.9***.

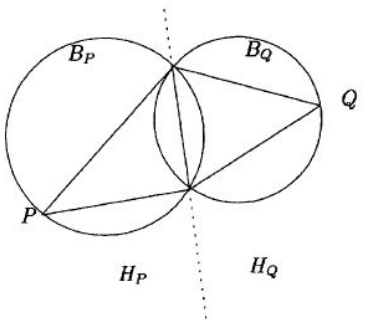


Figura 2.9: Sfere ed Iperpiani

Allora se $Q \notin B_P$, si può notare che:

$$B_P \cap H_Q \Leftrightarrow B_Q \cap H_Q$$

o viceversa,

$$H_Q \cap B_P \Leftrightarrow H_Q \cap B_Q$$

dove, se si considera la stessa configurazione per i due simplessi, H_P (risp.H_Q) è il semispazio contenente P (risp. Q) limitato dall'iperpiano che supporta la (d-1)-faccia comune. Questo risultato ci porta alla conclusione.
Si assumi che il criterio della sfera vuota valga per ogni coppia di elementi adiacenti e che esista nella triangolazione un punto P_n contenuto nella boccia associata al simplesso che non è adiacente ai simplessi che hanno P_n come vertice.

Sia K_0 questo simplesso, vedi ***Figura 2.10***.

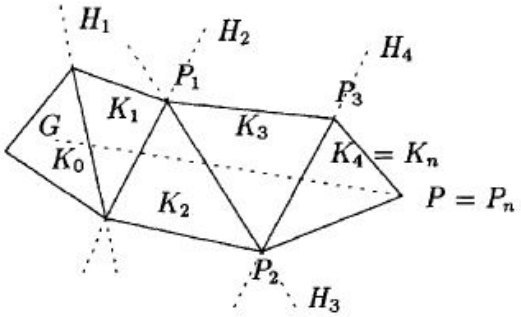


Figura 2.10: Il simplesso K_0 (Lawson's proof)

Consideriamo il punto G, che è il baricentro di K_0, e consideriamo il segmento $\overline{GP_n}$. Questa linea congiunge il simplesso K_0 al simplesso che indichiamo con K_n (uno dei simplessi della boccia di P_n) ed interseca un insieme di simplessi, indichiamoli con $K_0, K_1, ..., K_n$. Se i punti sono in posizione generale questi n simplessi hanno intersezione contenuta in una (d-1)- faccia. Se i punti non sono in posizione generale cambiando la scelta del punto G possiamo ricondurci a questo caso. Per ipotesi si ha:

$$P_n \in B_0$$

dove B_i è la boccia associata al simplesso K_i e K_{n-1} è adiacente a K_n, dopo il criterio della sfera vuota per i simplessi adiacenti:

$$P_n \notin B_{n-1}$$

I simplessi K_i sono scelti in base alla loro intersezione con il segmento $\overline{GP_n}$.
Sia P_i il vertice di K_i che non appartiene a K_{i-1} e sia H_i il semispazio contenente K_i ma non K_{i-1}. Secondo la definizione, si ha che $P_n \in H_i$. Abbiamo dimostrato perciò che esiste un indice i tale che:

$$P_n \in B_i \Rightarrow P_n \in B_{i+1}$$

Infatti, poiché $P_n \in H_{i+1}$ e $B_i \cap H_{i+1} \subset B_{i+1} \cap H_{i+1}$, allora $P_n \in B_{i+1}$. Dato che $P_n \in B_0 \Rightarrow P_n \in B_{n-1}$. Contraddizione. □

Capitolo 3

Impronte digitali e Triangolazione di Delaunay

Lo scopo principale della tesi è quello di confrontare grafi attraverso l'Edit Distance, ottenuti dalla triangolazione di Delaunay costruita sui punti minutiae delle impronte digitali. Innanzitutto è necessario capire come avviene l'estrazione delle minutiae in un'impronta digitale; perciò di seguito ho riportato alcuni dei passi principali da applicare per questo scopo.

3.1 Estrazione delle minutiae

3.1.1 Segmentazione

Il primo passo è quello dalla segmentazione, ovvero la separazione tra l'area di *foreground* (dove è contenuta l'impronta) e quella di *background* (lo sfondo)[1].

Figura 3.1: Immagine di un'impronta digitale prima e dopo la segmentazione

Tale separazione è molto utile nel momento in cui si vogliano estrarre delle caratteristiche locali da un'immagine molto rumorosa, poiché generalmente il rumore è presente in maggiore quantità sullo sfondo. Il foreground si caratterizza per la presenza di un pattern striato ed orientato, mentre il background presenta un pattern isotropico, caratterizzato quindi dall'assenza di un'orientazione dominante.
Per questi motivi non è possibile utilizzare semplicemente delle soglie locali o globali per eseguire una separazione efficace.
Per affrontare la segmentazione sono stati proposti vari approcci [1].
Come già osservato le regioni dello sfondo sono caratterizzate da isotropia, mentre l'impronta presenta delle direzioni ben precise.
Come si misura l'anisotropia (proprietà per cui il valore di una grandezza fisica ,in una sostanza o nello spazio, non è uguale in tutte le direzioni)?

- **Presenza di un picco in un istogramma locale di orientazioni**:
  L'orientazione delle ridge line è stimata in ogni pixel, si potrebbe calcolare l'immagine orientazione e definire l'istogramma per ogni suo blocco di dimensione fissata; la presenza di un picco significativo denota un pattern orientato, mentre un istogramma "piatto" è caratteristico di un segnale isotropico. Un problema si potrebbe riscontrare quando si hanno aree perfettamente uniformi, nelle quali non è possibile calcolare informazioni sull'orientazione e quindi si avranno istogrammi uniformi, ad esempio nelle zone con intensità costante.

- **Varianza dei livelli di grigio nella direzioni ortogonale al gradiente**:
  Nelle regioni rumorose il pattern non dipende dalla direzione, mentre l'area dell'impronta è caratterizzata da una varianza molto elevata nella direzione ortogonale all'orientazione delle ridge e molto bassa lungo le ridge.
  Questa tecnica è stata ideata da Ratha, Chen e Jain [15], in cui un algoritmo può segmentare l'immagine assegnando ogni blocco allo sfondo oppure all'impronta a seconda della varianza dei livelli di grigio nella direzione ortogonale all'orientazione delle creste. In basa alla varianza è inoltre possibile dare delle indicazioni sulla qualità dell'immagine, classificando ogni blocco come "buono", "medio", "scarso" e "sfondo", vedi ***Figura 3.2.***.

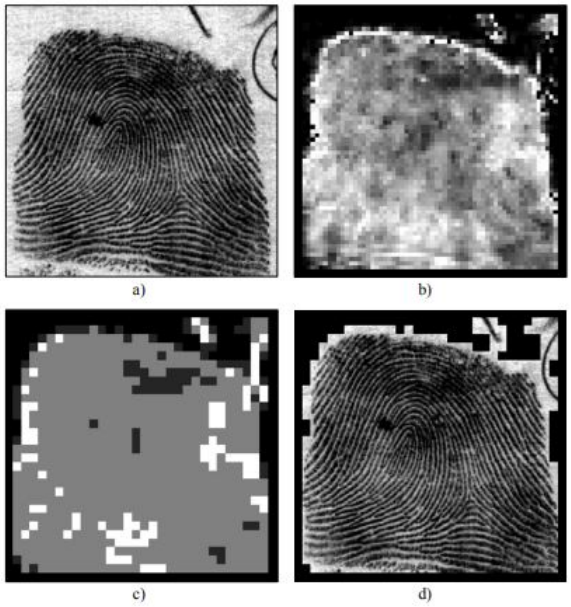


Figura 3.2: Segmentazione di un'impronta digitale proposta da Ratha, Chen,Jain (1995) [15]. a) immagine originale b) Immagine della varianza c) qualità immagine derivata dalla varianza d) immagine segmentata

- **Magnitudo del gradiente**:
  Maio e Maltoni [9] hanno ideato un altro metodo , che consiste nell'utilizzo della media della norma del gradiente in ogni blocco dell'immagine. Poiché l'area dell'impronta è ricca di edge dovuti all'alternanza di ridge e valli, il gradiente è elevato nel foreground e basso altrove.
  Il valore medio della magnitudo del gradiente può essere calcolato in questo modo:

$$\overline{G} = \frac{1}{N} \cdot \sum_{(i,j)\in I} (\nabla_x^2 + \nabla_y^2) \tag{3.1.1}$$

  dove N è il numero di pixel dell'immagine I.

- **Combinazione di più caratteristiche**:
  Per ogni pixel si calcolano alcune caratteristiche (coerenza del gradiente, media e varianza dell'intensità) e l'assegnazione a foreground/background è operata da un classificatore. Infine si effettua un post-processing.

- **Spettro della trasformata di Fourier**:
  Un'altra idea interessante è quella di andare a controllare lo spettro della trasformata di Fourier; infatti le zone che contengono creste e valli esibiscono un profilo sinusoidale, con definite frequenza ed orientazione, in corrispondenza delle quali lo spettro della trasformata di Fourier mostrerà dei picchi.

Sono state sviluppate molte altre tecniche di segmentazione attraverso algoritmi di *machine learning* [1], generalmente più accurate, ma anche più complesse, degli approcci basati sul *thresholding* rispetto a qualche caratteristica locale.
Per avere un'idea si pensi al metodo proposto da Chen et al. [15]; un classificatore lineare viene "addestrato" a selezionare blocchi dell'impronta sulla base di:

- un indicatore dell'ammassamento del blocco;
- la differenza tra la media dell'intensità del blocco e la media delle intensità di tutta l'immagine;
- la varianza del blocco.

3.1.2 Orientazione locale delle ridge line

Il secondo passo consiste nella creazione di un'immagine direzionale, cioè un'immagine in cui vengono definite le rette tangenti alle ridge line in ogni punto. L'orientazione locale delle ridge line in posizione $[i, j]$ è definita come l'angolo θ_{ij} che le ridge line, per un intorno piccolo a piacere del punto (i, j), formano con l'asse delle x.
L'immagine direzionale D è una matrice in cui ogni elemento θ_{ij}, corrispondente al nodo $[i, j]$ di una griglia quadrata posizionata sul pixel $[x_i, y_j]$, denota l'orientazione media delle ridge in un intorno di $[x_i, y_j]$.

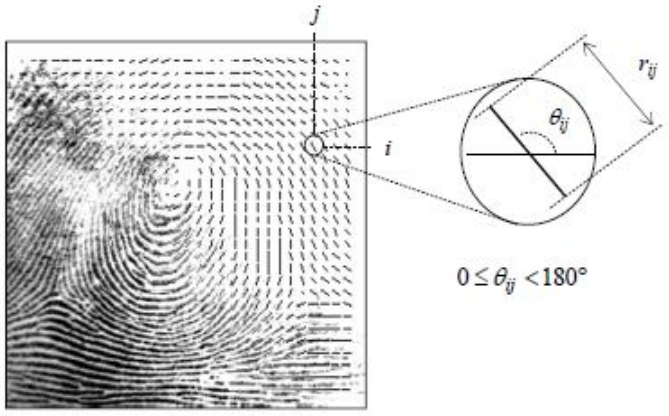


Figura 3.3: L'immagine sbiadita di un'impronta digitale nella corrispondente immagine direzionale calcolata su una griglia quadrata 16×16. *Ogni elemento denota l'orientazione locale dei ridge dell'impronta: la lunghezza dell'elemento è proporzionale alla sua affidabilità*

La funzione continua che associa ad ogni punto delle ridge line la sua direzione, è chiamata LRO, cioè *Local Ridge Orientation*. L'immagine direzionale è ,in altre parole,la rappresentazione grafica della funzione LRO, infatti una retta passante per un punto è univocamente determinata dall'angolo θ modulo πZ.
Invece di calcolare l'orientazione locale in ogni pixel, la maggior parte degli approcci effettua una stima in posizioni discrete.
L'approccio più semplice e naturale per l'estrazione dell'orientazione locale è basato sul calcolo del gradiente dell'immagine.
Il gradiente $\nabla(x_i, y_j)$ nel punto $[x_i, y_j]$ dell'immagine I è un vettore bidimensionale $[\nabla_x(x_i, y_j), \nabla_y(x_i, y_j)]$, dove ∇_x e ∇_y sono rispettivamente le derivate di I in $[x_i, y_j]$ rispetto alle direzioni x e y.
L'angolo di fase del gradiente indica la direzione del massimo cambiamento d'intensità dei pixel. La direzione θ_{ij} di un ipotetico "edge" che attraversa la regione centrata in $[x_i, y_j]$ è ortogonale all'angolo di fase del gradiente in $[x_i, y_j]$.
Alcuni problemi che si potrebbero riscontrare sono i seguenti:

- Non linearità e discontinuità attorno a $\frac{\pi}{2}$.
- La stima di una singola orientazione rappresenta un'analisi di basso livello eccessivamente sensibile al rumore. D'altra parte non è possibile fare una semplice media di più gradienti a causa della circolarità degli angoli.
- Il concetto di orientazione media non sempre è ben definito.

Una semplice soluzione, per superare il problema della circolarità, consiste nel raddoppiare gli angoli; ogni elemento dell'immagine direzionale D è codificato allora dal vettore:

$$d_{ij} = \left[r_{ij} \cdot \cos(2\theta_{ij}), r_{ij} \cdot \sin(2\theta_{ij}) \right] \tag{3.1.2}$$

Dove r_{ij} denota la coerenza dell'orientazione θ_{ij}.
La media degli angoli in una finestra locale di dimensione $n \times n$ può essere calcolata effettuando separatamente la media per le due componenti x e y.

$$\overline{d} = \left[\frac{1}{n^2} \cdot \sum_{i,j} (r_{ij} \cdot \cos(2\theta_{ij})), \frac{1}{n^2} \cdot \sum_{i,j} (r_{ij} \cdot \sin(2\theta_{ij})) \right] \tag{3.1.3}$$

Calcolo dell'orientazione locale

L'orientazione locale dominante può essere calcolata in modo robusto tramite la seguente formula [10], basata sulla media locale delle stime del gradiente in una finestra W di dimensione 17×17 centrata nel punto $[x_i, y_j]$.

$$\begin{aligned}
\theta_{ij} &= \frac{\pi}{2} + \frac{1}{2} \cdot a\tan 2(2G_{xy}, G_{xx} - G_{yy}) \\
G_{xy} &= \sum_{h=-8}^{8} \sum_{k=-8}^{8} \nabla_x(x_i + h, y_j + k) \cdot \nabla_y(x_i + h, y_j + k) \\
G_{xx} &= \sum_{h=-8}^{8} \sum_{k=-8}^{8} \nabla_x(x_i + h, y_j + k)^2 \\
G_{yy} &= \sum_{h=-8}^{8} \sum_{k=-8}^{8} \nabla_y(x_i + h, y_j + k)^2.
\end{aligned} \tag{3.1.4}$$

dove $\nabla_x \, e \, \nabla_y$ sono le componenti x e y del gradiente calcolato su una maschera di Sobel 3×3 mentre $a\tan 2(y, x)$ è la funzione che calcola l'arcotangente tenendo conto dei segni dei suoi argomenti per determinare il quadrante di appartenenza.
Osserviamo inoltre che la coerenza r della stima θ può essere derivata dalla concordanza dei vettori orientazione nella finestra W ; infatti, a causa della continuità delle increspature dell'impronta, cambi troppo rapidi di orientazione spesso denotano un

stima errata. Ad esempio, Kass e Witkin [11] definiscono la coerenza come:

$$r = \frac{\left|\sum_W d\right|}{\sum_W |d|} \tag{3.1.5}$$

La (3.1.5) è uno scalare con valori in [0,1]; assume il valore massimo se le orientazioni sono tutte concordi, mentre assume il minimo se esse sono tutte opposte. Se misuriamo la coerenza usando la (3.1.5) per il metodo esposto nella (3.1.4), allora otteniamo:

$$r_{ij} = \frac{\sqrt{(G_{xx} - G_{yy})^2 + 4G_{xy}^2}}{G_{xx} + G_{yy}} \tag{3.1.6}$$

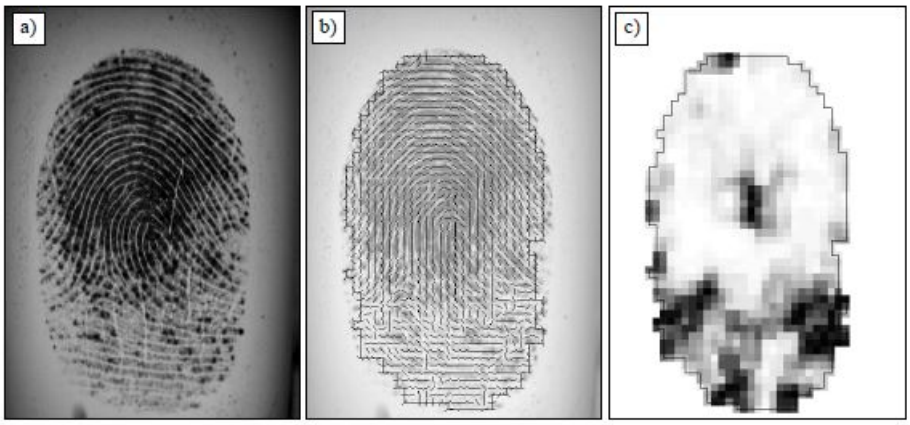


Figura 3.4: a)Impronta digitale b)La sua orientazione locale c)La sua coerenza locale su blocchi 3x3

Frequenza locale delle ridge line

La frequenza locale delle ridge line f_{xy} nel punto [x,y] è definita come il numero di ridge per lunghezza unitaria lungo un ipotetico segmento centrato nel punto [x,y] e ortogonale all'orientazione locale delle ridge.
E' possibile calcolare un'immagine delle frequenze F, analoga all'immagine direzionale D, stimando la frequenza in posizioni discrete organizzate in una matrice.

Figura 3.5: Due impronte digitali e la loro corrispondente immagine della frequenza

Hong, Wan, and Jain (1998) [16] hanno proposto un possibile approccio che consiste nel contare il numero medio di pixel tra picchi consecutivi dei livelli di grigio lungo la direzione ortogonale all'orientazione locale delle ridge line. La superficie S corrispondente all'impronta è sezionata con un piano parallelo all'asse z e ortogonale all'orientazione locale.
La frequenza f_{ij} si calcola come segue:

- Si definisce una finestra di dimensione 32×16 nel sistema di coordinate delle ridge (ruotate per allineare l'asse y con l'orientazione della ridge line);
- La x-signature dei livelli di grigio si ottiene sommando, per ogni colonna x, i livelli di grigio dei pixel corrispondenti nella finestra orientata;
- f_{ij} è calcolata come l'inverso della distanza media tra due picchi consecutivi nella x-signature.

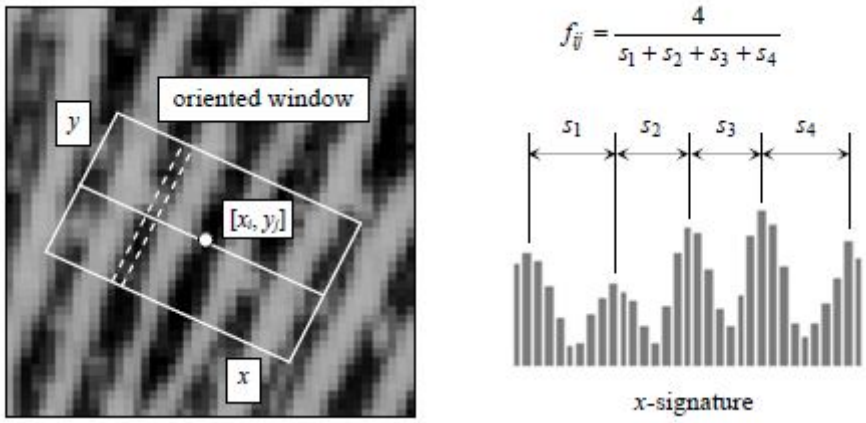


Figura 3.6: Finestra orientata centrata in $[x_i, y_j]$. Le linee tratteggiate mostrano i pixel i cui livelli di grigio sono accumulati per una data colonna della x-signature. La x-signature sulla destra mostra 5 picchi; le 4 distanze tra picchi consecutivi vengono calcolate per determinare la frequenza della cresta locale

Un approccio alternativo consiste nel modellare localmente il ridge pattern come una superficie sinusoidale e sfruttare il teorema della variazione per stimare la frequenza.
La variazione V di una funzione h nell'intervallo $[x_1, x_2]$ corrisponde al "cambiamento verticale" in h. Se la funzione h è periodica in $[x_1, x_2]$ (o le variazioni di ampiezza in questo intervallo sono piccole), la variazione può essere espressa in funzione della frequenza (f) e dell'ampiezza media (α_m).
Inoltre la variazione e l'ampiezza media di un ridge pattern bidimensionale possono essere stimati tramite le derivate parziali del primo e del secondo ordine.
La varazione può essere calcolata come segue:

$$V(h) = \int_{x_1}^{x_2} \left| \frac{dh(x)}{dx} \right| dx = \int_{x_1}^{x_2} \left| \frac{dh}{dx} \cdot dx \right| = \alpha_1 + \alpha_2 + ... + \alpha_8 \tag{3.1.7}$$

$$V(h) = (x_2 - x_1)2 \cdot f \cdot \alpha_m \Rightarrow f = \frac{V(h)}{2(x_2 - x_1) \cdot \alpha_m} \tag{3.1.8}$$

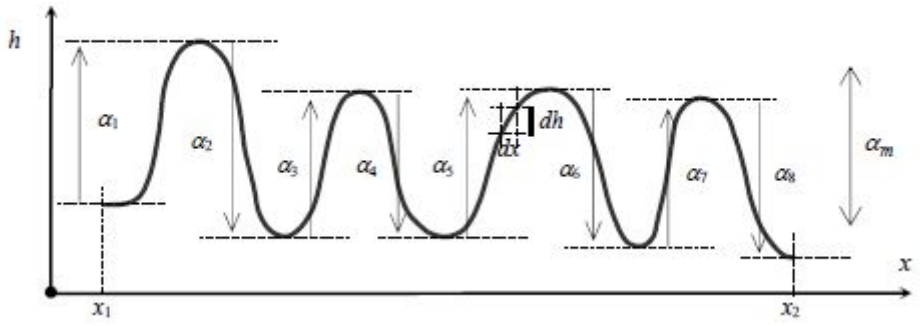


Figura 3.7: La variazione di una funzione h nell'intervallo $[x_1, x_2]$ è la somma dell'ampiezza $\alpha_1 ... \alpha_8$ (α_m è l'ampiezza media). Se la funzione è periodica o l'ampiezza è costante nell'intervallo di interesse α_m può essere usata per approssimare i singoli valori di α. Quindi la variazione può essere espressa come $2\alpha_m$ moltiplicata per il numero dei periodi della funzione nell'intervallo.

Localizzazione di singolarità

La maggior parte degli approcci si basa sull'immagine direzionale dell'impronta. Un metodo pratico ed elegante si basa sul calcolo dell'*indice di Poincarè*.
Sia G un campo vettoriale e sia C una curva immersa in G, l'indice di Poincarè $P_{G,C}$ è definito come la rotazione totale dei vettori di G lungo C.

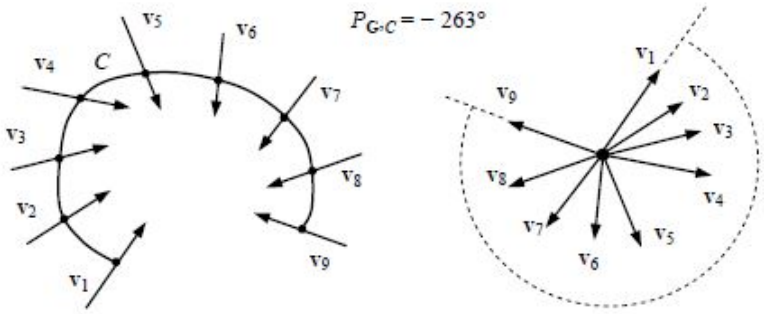


Figura 3.8: Indice di Poincarè calcolato su una curva C imersa in un campo vettoriale G

Sia G il campo associato all'immagine delle orientazioni dell'impronta D e sia $[i, j]$ la posizione dell'elemento θ_{ij} nell'immagine.

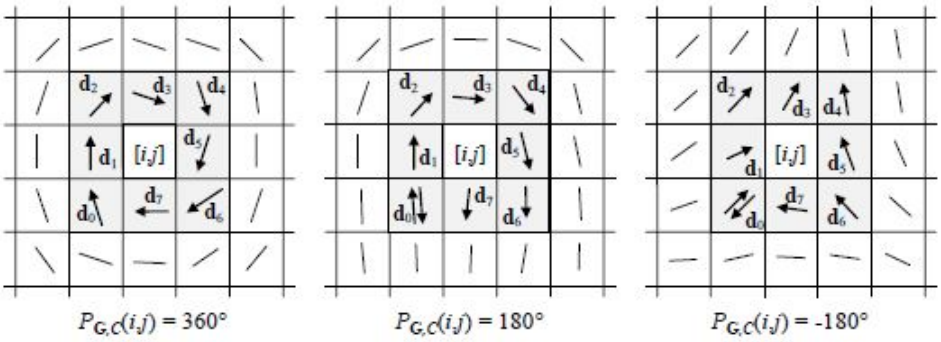


Figura 3.9: Esempi di indici di Poincarè in 8 intorni di punti che appartengono (da sinistra a destra) ad una singolarità whorl, loop e delta rispettivamente

Dove:

$$P_{G,C}(i,j) = \sum_{k=0}^{7} angle(d_k, d_{(k-1) mod 8} \tag{3.1.9}$$

L'indice di Poincarè $P_{G,C}(i,j)$ in [i,j] può essere calcolato nel modo seguente:

- La curva C è un percorso chiuso definito come sequenza ordinata di alcuni elementi di D, tali che [i,j] siano punti interni;

- L'indice $P_{G,C}(i,j)$ si calcola sommando algebricamente le differenze di orientazione tra elementi adiacenti di C;

- La somma delle differenze di orientazioni richiede che sia associata una direzione a ciascuna orientazione. Si può selezionare casualmente la direzione del primo elemento e assegnare la direzione più vicina a quella dell'elemento precedente a tutti gli elementi successivi.

- Si dimostra che, su curve chiuse, l'indice di Poincarè assume solo uno dei valori discreti 0°, ±180°, ±360°.

In particolare, per le singolarità di impronte digitali:

- $P_{G,C} = 0° \Rightarrow$ nessuna singolarità

- $P_{G,C} = 360° \Rightarrow whorl$

- $P_{G,C} = 180° \Rightarrow loop$

- $P_{G,C} = -180° \Rightarrow delta$

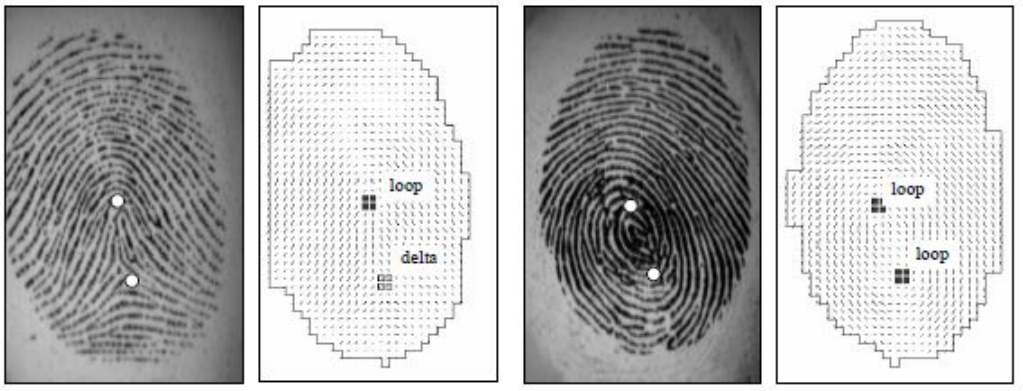


Figura 3.10: Rilevamento di singolarità con indice di Poincarè

Basandosi sull'osservazione che solo un numero limitato di singolarità può essere presente in un'impronta digitale, Karu e Jain (1996) [20] hanno proposto di lisciare (tecnica dello smoothing) iterativamente l'immagine di orientamento (attraverso la media) finché un indice di Poincaré non rileva un numero valido di singolarità. Infatti, una semplice analisi delle diverse classi di impronte digitali mostra che:

- Le impronte digitali dell'arco non contengono singolarità.

- Le impronte digitali di loop sinistro, loop destro e tented arch contengono un loop e un delta.

- Le impronte digitali Whorl contengono due loop (o un whorl) e due delta.

I suddetti vincoli sono ben dimostrati da Zhou, Gu e Zhang (2007) [17] i quali concludono che per ogni impronta digitale completamente catturata ci sono lo stesso numero di loop e delta.
Lo smoothing di orientazioni locali è necessario per controllare il rumore. Per capire quanto bisogna regolarizzare è necessario iterare finché non si localizza un numero valido di singolarità o si raggiunge il numero massimo di passi consentiti.

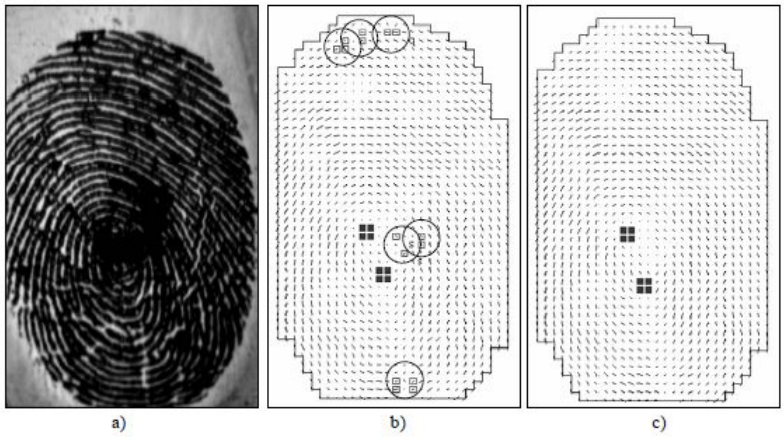


Figura 3.11: a) Impronta digitale di scarsa qualità b) Le singolarità dell'impronta digitale in a) sono estratte tramite il metodo di Poincarè c) Regolarizzazione dell'immagine

3.1.3 Enhancement

Un ulteriore passo è costituito dall'*enhancement*, un processo che, basandosi sull'immagine direzionale precedentemente ottenuta, ha come scopo quello di migliorare la qualità dell'immagine dell'impronta. Infatti le prestazioni degli algoritmi di estrazione feature e riconoscimento di impronte dipendono fortemente dalla qualità dell'immagine.
Una porzione significativa delle immagini acquisite (circa 10%) e di scarsa qualità a causa delle condizioni della pelle (secca o umida, con tagli o abrasioni), rumore, pressione del dito non corretta, bassa qualità intrinseca delle impronte di particolari categorie di persone (lavoratori manuali, anziani).
Esistono diversi tipi di degradazioni associate alle impronte digitali:

- le ridge line non sono continue;
- le ridge line parallele non sono ben separate a causa della presenza di rumore;
- tagli e abrasioni.

Figura 3.12: a) Immagine impronta digitale di buona qualità b) Media qualità c) Scarsa qualità

Le aree dell'impronta individuate dopo la segmentazione possono essere divise in tre categorie [1]:

- Regione ben definita: caratterizzata da ridge line ben distinte;
- Regione recuperabile: ridge line parzialmente rovinate ma ancora chiaramente visibili; le zone circostanti forniscono informazioni sufficienti circa la loro struttura;
- Regione non recuperabile: ridge line completamente rovinate per la presenza di rumore e le zone circostanti non forniscono informazioni utili sulla loro struttura.

Ogni regione può essere classificata come appartenente a una delle tre categorie sulla base di diversi criteri di qualità dell'immagine: contrasto, consistenza delle orientazioni, frequenza delle ridge line o altre caratteristiche locali.

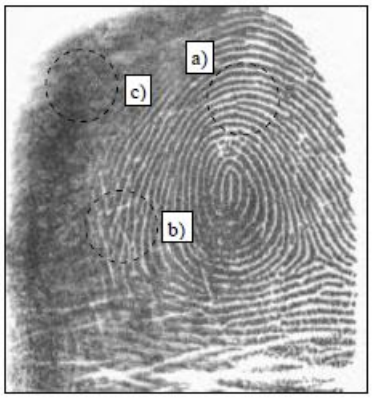


Figura 3.13: a) Ragione ben-definita b)Regione recuperabile c)Regione non recuperabile

L'input per la procedura di enhancement è solitamente un'immagine a livelli di grigio; l'output può essere un'immagine a livelli di grigio o un'immagine binaria.
Le tecniche di miglioramento della qualità delle immagini generiche non risultano molto efficaci con le impronte digitali; tuttavia alcune operazioni (es. aumento del contrasto, normalizzazione, modifica dell'istogramma) possono rappresentare un utile passo di pre-processing. Una possibile tecnica di normalizzazione usata da Hong, Wan, and Jain (1998) [19] agisce sul singolo pixel determinandone il nuovo valore di intensità in base alla formula seguente:

$$I'|x,y| = \begin{cases} m_0 + \sqrt{(I[x,y]-m)^2 \cdot \frac{v_0}{v}}, & \text{se I[x,y] > m,} \\ m_0 - \sqrt{(I[x,y]-m)^2 \cdot \frac{v_0}{v}}, & \text{altrimenti.} \end{cases}$$

dove m e v sono la media e la varianza nell'immagine e m_0 e v_0 sono la media e la varianza desiderate dopo la normalizzazione.

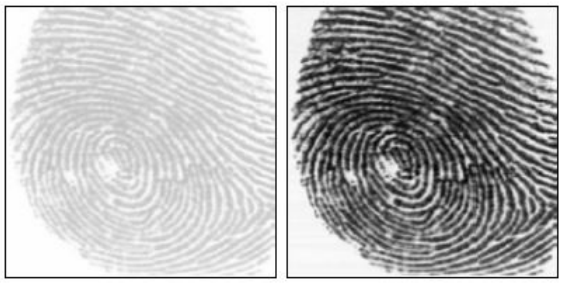


Figura 3.14: Esempio di normalizzazione con i metodo descritto di Hong,Wan,Jain usando $m_0 = 100$ *e* $v_0 = 100$

Enhancement - Filtraggio contestuale

Nel filtraggio contestuale le caratteristiche del filtro utilizzato cambiano a seconda del contesto locale; viene precalcolato un insieme di filtri e in ogni regione dell'immagine si seleziona quello più appropriato. Nell'ambito delle impronte digitali il contesto locale viene descritto dalla frequenza e dall'orientazione delle ridge line che determinano le caratteristiche del ridge pattern. Sono stati proposti molti tipi diversi di filtro per l'enhancement, ma tutti perseguono gli stessi obiettivi:

- mediare i valori dei livelli di grigio nella direzione delle ridge per "riempire" piccoli buchi (*filtro low-pass*);
- aumentare la separazione tra ridge e valli e separare ridge parallele agendo in direzione ortogonale rispetto a quella delle ridge line (*filtro band-pass*).

3.1.4 Miglioramento delle immagini ed estrazione delle minutiae

Approccio tradizionale [1]:

- *Binarizzazione:* conversione di un'immagine a livelli di grigio in un'immagine binaria.
- *Thinning:* l'immagine binaria è sottoposta a un passo di assottigliamento che riduce lo spessore delle ridge line ad 1 pixel;

- *Localizzazione:* si fa ricorso a una scansione dell'immagine per localizzare i pixel corrispondenti alle minuzie

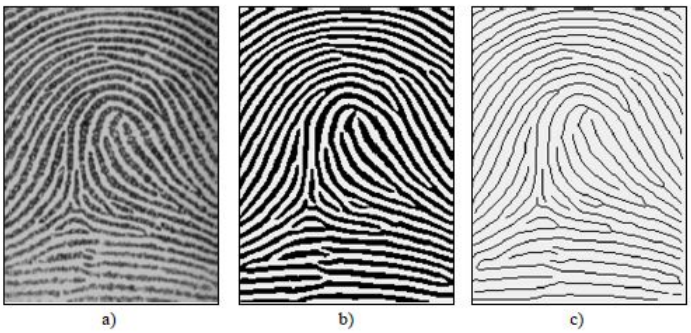


Figura 3.15: a) Impronta digitale b) Immagine binarizzata c)Scheletro dell'immagine dopo il thinning

Binarizzazione

Ci sono diversi approcci utilizzati per la binarizzazione. Ad esempio usando un *threshold globale*, spesso inaffidabile a causa delle differenze delle condizioni di acquisizione, ma al contempo semplice e computazionalmente rapido; oppure preferendo un *threshold locale*, in modo da non perdere informazioni a causa delle differenze di intensità e contrasto nelle varie regioni dell'immagine [1].
Moayer e Fu hanno proposto una tecnica dove vengono utilizzati operatori differenziali, in cui viene calcolato il Laplaciano dell'immagine ad ogni iterazione, assegnati valori 0 ed 1 ai pixel che escono dall'intervallo delimitato da due threshold dinamici, quindi avvicinati i valori di threshold in modo da assicurare la convergenza.
Ratha, Chen, and Jain (1995)[18] hanno introdotto un approccio basato sulla localizzazione di picchi nei profili dei livelli di grigio in una sezione ortogonale all'orientazione delle ridge line. Attorno a ogni pixel $[x, y]$ si posiziona una finestra orientata di dimensione 16×16. Il profilo dei livelli di grigio si ottiene proiettando l'intensità dei pixel nella sezione centrale. Il profilo è successivamente smussato tramite un'operazione di media locale; i picchi e i due pixel a essi adiacenti da entrambi i lati sono associati al foreground nell'immagine binarizzata.

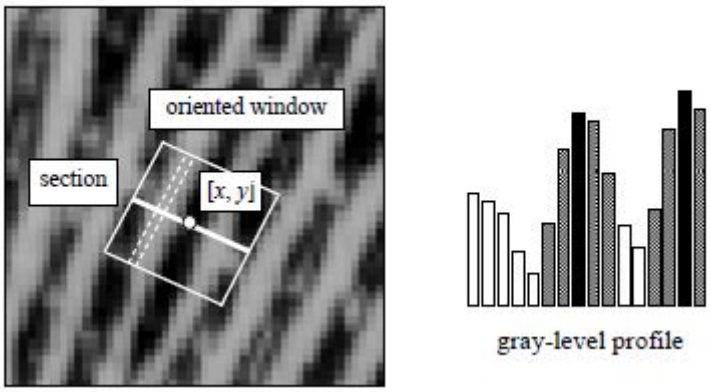


Figura 3.16: Esempio di un profilo dei livelli di grigio ottenuta proiettando l'intensità dei pixel nella sezione centrata a $[x, y]$ *e normale alla local ridge orientation* θ_{xy}

Thinning e Localizzazione

Dopo aver provveduto alla binarizzazione con qualche algoritmo, in genere viene eseguito il *thinning*, assottigliamento, che riduce ogni linea ad un pixel di larghezza [1]; il risultato sarà lo scheletro dell'impronta. Prima di questo passo potrebbero essere necessari anche degli algoritmi per colmare le interruzioni delle creste, riempire buchi, eliminare collegamenti anomali tra creste.
Una volta eseguito il thinning basta semplicemente contare quanti punti ci sono nell'intorno quadrato 3×3 di ogni pixel e perciò la localizzazione delle minuzie si basa sull'analisi del *crossing number*.

$$cn(p) = \frac{1}{2} \cdot \sum_{i=1\ldots8} (|val(p_{imod8} - val(p_{i-1}|) \tag{3.1.11}$$

dove p_0, p_1, ..., p_7 sono i pixel appartenenti a una sequenza ordinata di pixel che costituiscono l'intorno del pixel p e $val(p) \in 0, 1$ è il valore del pixel p. Si nota che un pixel p con $val(p) = 1$:

- È un punto interno a una ridge line se $cn(p) = 2$;
- Corrisponde a una terminazione se $cn(p) = 1$;
- Corrisponde a una biforcazione se $cn(p) = 3$;
- Appartiene a una minuzia più complessa se $cn(p) > 3$.

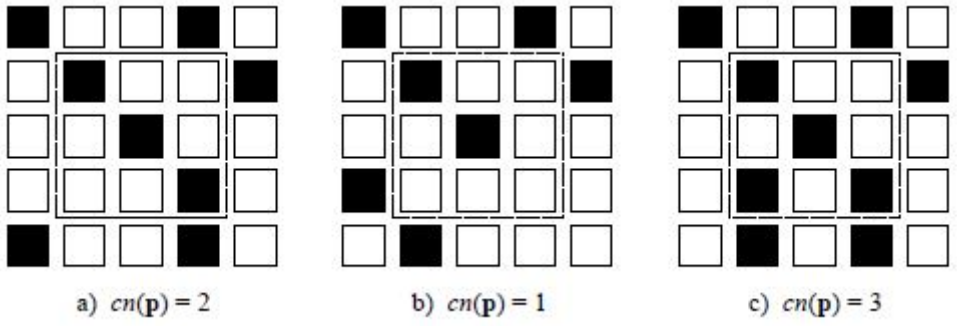


Figura 3.17: a) punto interno a una ridge line b) terminazione c) biforcazione

In letteratura sono presenti molti metodi che, invece, di binarizzare l'immagine, lavorano direttamente in scala di grigi.
Questo perché si riscontrano vari problemi durante l'estrazione di minuzie da immagini binarie:

- Durante il processo di binarizzazione si potrebbero perdere informazioni significative;
- La binarizzazione e l'assottigliamento sono costose dal punto di vista dei tempi di elaborazione;
- L'assottigliamento potrebbe introdurre un numero elevato di false minuzie;
- Molte tecniche di binarizzazione non forniscono buoni risultati quando applicate a immagini di scarsa qualità.

Uno dei principali e più noti algoritmi per la diretta estrazione di minutiae dall'immagine è quello proposto da Maio e Maltoni [9]. L'idea di base è quella di inseguire le ridge line in un'immagine a livelli di grigio, guidati dall'orientazione locale. Seguire il profilo delle creste significa, ad ogni passo, sezionare la cresta trasversalmente, trovare il massimo locale, spostarsi lungo l'orientazione della cresta per calcolare la nuova sezione. Il procedimento si itera fino a che non si riesca più a trovare un massimo locale, per cui siamo in presenza di una minutia.
Altri approcci sono stati utilizzati, alcuni sfruttando reti neurali, altri sfruttando le proprietà di simmetria delle minutiae.

3.2 Applicazione della triangolazione di Delaunay

Una volta estratte le minutiae dall'impronta digitale, come accennato all'inizio di questo capitolo, viene costruita la triangolazione di Delaunay.
Cerchiamo ora di comprendere il motivo per il quale tra le varie triangolazioni viene scelta proprio la triangolazione di Delaunay.

3.2.1 Perchè si usa la triangolazione di Delaunay?

Il principale scopo della tesi è quello di confrontare grafi attraverso l'Edit Distance, ottenuti dalla triangolazione di Delaunay costruita sui punti minutiae delle impronte digitali.
Analizziamo più nel dettaglio il motivo per il quale tra le varie triangolazioni viene scelta la Triangolazione di Delaunay, già argomentata nel capitolo precedente.

L'idea è quella di associare un'unica struttura topologica con le minutiae della fingerprint [12] [13]. I triangoli minutiae di Delaunay hanno un buon potere discriminatorio, tra tutti i possibili triangoli minutiae, sono gli unici a soddisfare le proprietà della triangolazione di Delaunay.
Alcune caratteristiche chiave della triangolazione di Delaunay sono:

- è unica (supponendo non vi siano degenerazioni);
- può essere calcolata in modo efficiente in tempo $O(N)$;
- il rumore ha un'influenza solamente locale.

Inoltre ciò consentirebbe di ridurre i requisiti di memoria senza sacrificare l'accuratezza del riconoscimento, preservare selettività dell'indice e migliorare i tempi di riconoscimento.

La triangolazione è un processo che prende una regione dello spazio e la divide in sottoregioni. Lo spazio può essere di qualsiasi dimensione, ma ci limiteremo a considerare uno spazio 2D poiché si tratta di punti minutiae(2D).
In questo caso, le sottoregioni sono semplicemente dei triangoli. Il principale obbiettivo è quello di associare una struttura topologica 2D alle minutiae [13].
Dato un set S di punti $P_1, ..., P_n$, possiamo calcolare la triangolazione di Delaunay di S calcolando prima il suo diagramma di Voronoi.
Il diagramma di Voronoi suddivide lo spazio 2D in regioni attorno a ciascun punto, in modo che tutti i punti nella regione attorno a P_i siano più vicini a P_i rispetto a qualsiasi altro punto di S.
Dato il diagramma di Voronoi, la triangolazione può essere costruita collegando il centro di ogni coppia di regioni di Voronoi adiacenti.
Come già osservato la triangolazione di Delaunay ha diverse proprietà, tra cui:

(i) La triangolazione di Delaunay di un insieme di punti non degenere è unica;

(ii) Un circumcerchio costruito sui tre punti di un triangolo di Delaunay non contiene altri punti;

(iii) L'angolo minimo tra tutti gli angoli di tutti i triangoli in una triangolazione di Delaunay è maggiore del minimo angolo in qualsiasi altra triangolazione ottenuta con gli stessi punti.

La proprietà (i) evidenzia l'importanza dell'uso di questi triangoli nell'indicizzazione e quindi nella riduzione dei requisiti di memoria. Dalla proprietà (ii) si può dedurre che l'inserimento di un nuovo punto in una triangolazione di Delaunay influisce solo sui triangoli i cui circumcerchi contengono quel punto. Di conseguenza, il rumore influenza la triangolazione di Delaunay solo localmente.
L'ultima proprietà (iii) implica che i triangoli ottenuti non siano dei triangoli "magri"; quest'ultimi non sono triangoli desiderabili, poiché il calcolo della trasformazione geometrica potrebbe diventare instabile(piccoli errori nelle posizioni delle minutiae potrebbero produrre grandi errori nel calcolo dei parametri nella trasformazione).
La triangolazione di Delaunay ed il digramma di Voronoi sono algoritmi molto efficienti, poichè il numero di spigoli in entrambi è proporzionale ad una piccola costante per il numero dei punti (O(N)). La complessità dell'algoritmo è $O(N)$.

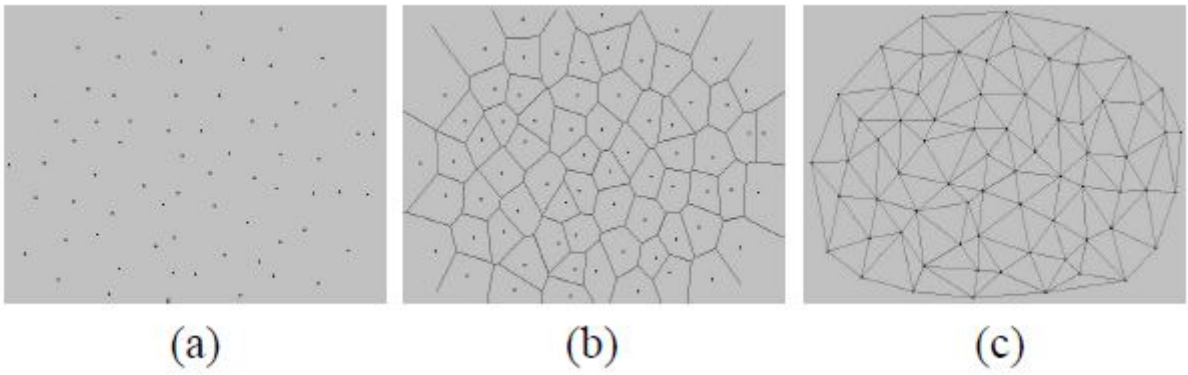


Figura 3.18: (a)insieme di punti (b) Diagramma di Voronoi (c) Triangolazione di Delaunay

3.2.2 Un esempio teorico sull'utilizzo della Triangolazione di Delaunay per il matching

Un'applicazione della triangolazione di Delaunay per il matching di impronte digitali è riportata nell'articolo [14]. Una volta creati i triangoli minutiae, vengono abbinati utilizzando degli invarianti.
Si osservi ad esempio la ***Figura 3.19***. In generale, una coppia di triangoli minutiae forniscono informazioni sufficienti per calcolare una trasformazione che allinea set di minutiae.
Per stimare i buoni allineamenti si utilizza la "tecnica del voto", applicata nello spazio di trasformazione. Ogni ipotetica trasformazione viene esplicitamente verificata, utilizzandola per allineare set di minutiae, per poi contare il numero di minutiae sovrapposte.

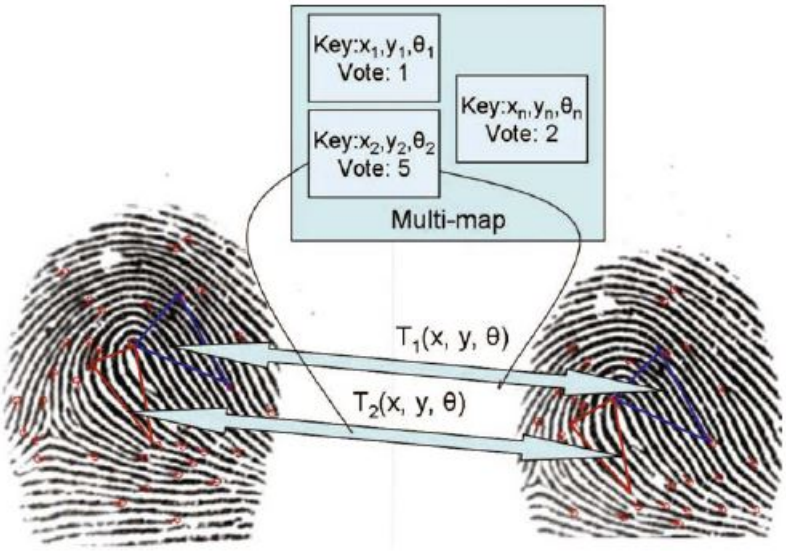


Figura 3.19: Matching triangoli minutiae. I buoni allineamenti sono calcolati usando la tecnica di voto nello spazio di trasformazione.

La complessità computazionale per formare triangoli minutiae è $O(n^3)$. Per mantenere bassa la complessità, si potrebbe associare una struttura topologica unica con le minutiae usando la triangolazione di Delaunay per il matching; infatti ciò ridurrebbe il numero di triangoli minutiae a $O(n)$, aumentando la velocità di corrispondenza senza diminuire la precisione.
Una volta calcolata la triangolazione di Delaunay di un set di minutiae, vengono considerati due gruppi di caratteristiche di ciascun triangolo, relative ai lati e agli angoli del triangolo.
Più precisamente, dato un triangolo minutiae, vedi ***Figura 3.20***, il primo gruppo di caratteristiche, indicato come V_t, include attributi che sono invarianti per trasformazioni rigide:

$$V_t = \left[\frac{l_1}{l_3}, \frac{l2}{l3}, \cos(A) \right]$$

Dove $l_1 \leq l_2 \leq l_3$,con:

$$0 \leq \frac{l_1}{l_3} \leq 1$$

$$0 \leq \frac{l2}{l3} \leq 1$$

$$-1 \leq \cos(A) \leq 1$$

e A è l'angolo più grande del triangolo (l'angolo tra i due lati più piccoli).

La ragione per cui si utilizza il coseno dell'angolo e non l'angolo stesso è perchè il valore dell'angolo è sensibile al rumore introdotto dall'algoritmo di estrazione, mentre il coseno riesce a filtrare meglio tale rumore.
Si può osservare inoltre che sebbene la triangolazione di Delaunay tenda ad evitare triangoli cosiddetti "magri", che porterebbero ad un trasformazione geometrica instabile, ciò non può essere garantito. Pertanto vengono rifiutati triangoli il cui angolo è maggiore di una determinata soglia.

Il secondo gruppo di caratteristiche è indicato con V_m e coinvolge gli angoli delle minutiae.

$$V_m = [\widehat{m_1}, \widehat{m_2}, \widehat{m_3}]$$

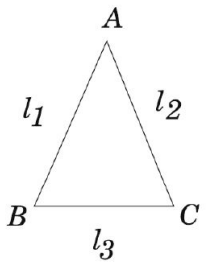


Figura 3.20: Triangolo definito da tre minutiae

Considerando due set di minutiae T e Q, i triangoli in T sono confrontati con i triangoli in Q usando i seguenti tre vincoli:

- **Consistenza della similarità**: Verifica la somiglianza tra due triangoli minutiae. Ciascun triangolo è rappresentato da sei invarianti, se le differenze tra coppie di invarianti corrispondenti sono tutte al di sotto di una soglia, allora si considera che è triangoli corrispondano.

- **Consistenza della planarità**: Verifica se i triangoli delle minutiae corrispondenti possono essere allineati mediante trasformazioni piane. Ad esempio si osservi la ***Figura 3.21***. Se si ordinano le minutiae partendo dalla prima di ogni triangolo e procedendo in senso orario, l'ordine della triangolo sinistro sarebbe $m_{11}\, m_{12}\, m_{13}$ mentre l'ordine nel triangolo destro sarebbe $m_{21}\, m_{23}\, m_{22}$. Tali incoerenze possono essere corrette cambiando l'ordine delle minutiae nel triangolo sinistro (ad esempio a partire da m_{12}). La coerenza della somiglianza può essere ora testata usando il nuovo ordinamento.

- **Consistenza dell'orientamento delle minutiae**: Eseguita stimando le trasformazioni rigide che allineano i triangoli corrispondenti e calcolando le differenze di orientamento delle minutiae.

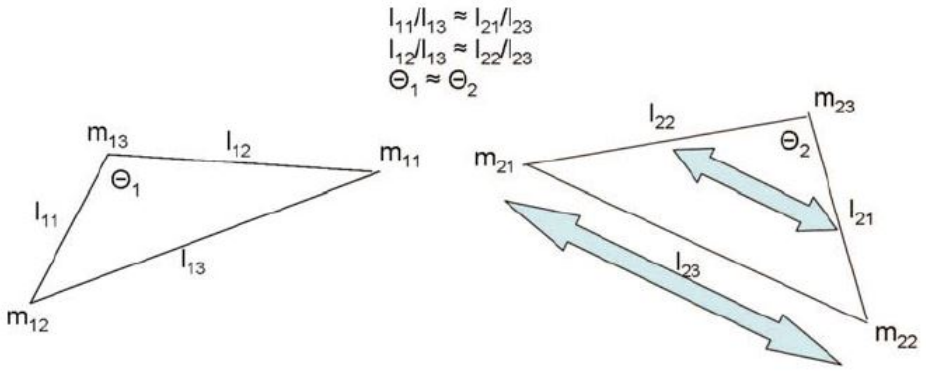


Figura 3.21: Un esempio di inconsistenza nell'ordinamento di minutiae tra triangoli

Se tutti i vincoli sono soddisfatti allora i triangoli minutiae corrispondono.
Per trovare trasformazioni "globalmente coerenti" si utilizza lo schema di voto in cui ogni coppia di triangoli assegna un voto ponderato nello spazio di trasformazione. Per diminuire gli errori vengono assegnati voti anche agli intorni vicini usando però pesi più bassi. Le trasformazioni che ricevono molti voti vengono poi considerate per altre verifiche.
Nella ***Figura 3.22*** si possono osservare gli input usati nello spazio di trasformazione. Ogni input contiene i parametri di trasformazione, il numero di voti ed un elenco di minutiae corrispondenti.
Inoltre,poiché si allineano triangoli minutiae usando trasformazioni rigide, lo spazio di trasformazione è tridimensionale (x,y,h).

X	Y	Cos(θ)	Sen(θ)	Vote	p_{11}	p_{12}	...	p_{1n}
					p_{12}	p_{22}		p_{2n}

Figura 3.22: Input nello spazio di trasformazione

La fase di verifica determina se due impronte digitali corrispondono allo stesso dito di un individuo o meno.

Ciò viene eseguito allineando le due impronte digitali e calcolando le sovrapposizioni. La sovrapposizione tra set di minutiae è determinata considerando le differenze tra le posizioni corrispondenti e gli orientamenti degli angoli.
In altre parole, data un'impronta digitale, viene generata una lista di candidati e per ogni corrispondenza viene calcolata una trasformazione, che verrà applicata sull'impronta candidata, in modo da allinearla con quella originale della query per poi confrontarle.
Il numero delle minutiae sovrapposte viene normalizzato per calcolare un punteggio di somiglianza tra i set di minutiae. Specificatamente, sia t e q il numero di minutiae nei due gruppi, rispettivamente, se m è il numero di minutiae corrispondenti, allora il punteggio di somiglianza s è calcolato nel modo seguente:

$$s = \frac{2m}{t+q} \times 100$$

Spesso però, la pressione del dito sul sensore o l'elasticità della pelle, possono produrre deformazioni e quindi degli errori. Perciò l'articolo [13] ha proposto di migliorare l'allineamento aggiungendo una seconda fase di verifica, nella quale viene utilizzata una trasformazione affine, in modo da allineare impronte digitali in maniera più accurata.

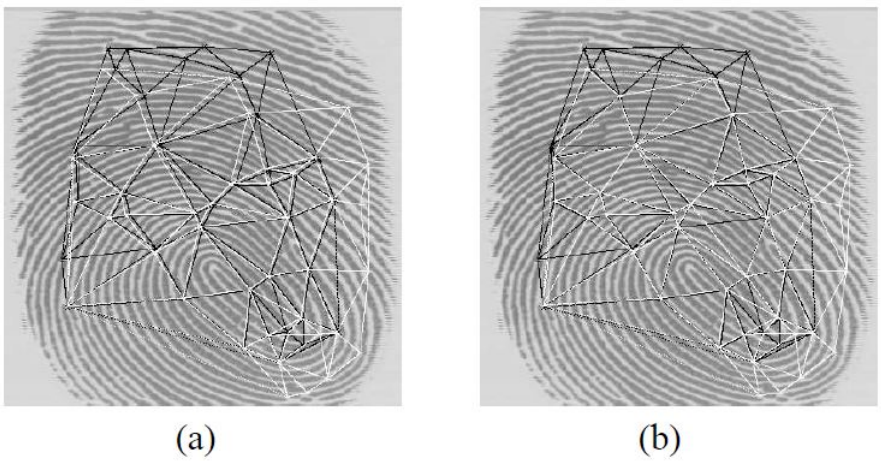


Figura 3.23: Allineamento di due impronte digitali. (a) usando una trasformazione di similarità (b) usando una trasformazione affine

3.2.3 Il nostro metodo

Il mio lavoro, che consiste nel confrontare due grafi costruiti attraverso la triangolazione di Delaunay sui punti minutiae estratti da impronte digitali, è stato implementato usando l'ambiente software MATLAB. Il primo passo consiste nell'estrazione delle minutiae.
E' possibile utilizzare dapprima la funzione "$find\_minutiae$" per trovare automaticamente le minutiae dell'impronta. Questa funzione utilizza:

- la funzione "segmentation", per ottenere una rapida, ma non ottimale, segmentazione dell'impronta, separando le creste dallo sfondo;
- la funzione "enhancement", per migliorare grossolanamente l'impronta, al fine di rendere più precisa l'estrazione delle minutiae.

La funzione "$find\_minutiae$" ha due parametri, che sono nell'ordine:

1. percorso al file di origine (che contiene l'immagine dell'impronta da analizzare);
2. percorso al file di destinazione (che conterrà le posizioni delle minutiae).

Il codice è il seguente:

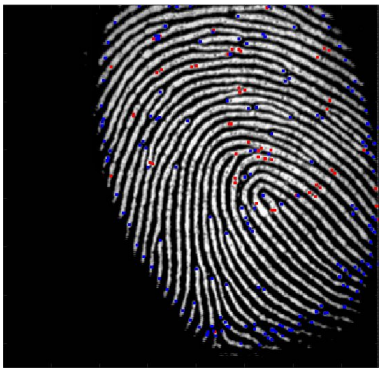
```
function minutiae = find_minutiae(filename, outname)
%% Read image from file and convert to gray scale (normalized to [0,1])
gray_image = imread(filename);
if size(gray_image, 3) == 3
    gray_image = rgb2gray(gray_image);
end
gray_image = adapthisteq(gray_image);
gray_image = normalize(double(gray_image));

%% Put ridges on local maxima
gray_image = 1-gray_image;

%% Perform segmentation
[masked_img, mask] = segmentation(gray_image);

%% Perform global thresholding (Otsu's method)
bin_img = masked_img > graythresh(masked_img(logical(mask)));

%% Perform thinning
thin_image = double(bwmorph(bin_img, 'thin', Inf));

%% Compute minutiae
cross_numbers = (conv2(thin_image, ones(3,3), 'same')-1) .* thin_image;
bifurcation = bwmorph(cross_numbers == 3, 'shrink', Inf);
endings = bwmorph(cross_numbers == 1, 'shrink', Inf);
[i, j] = find(bifurcation | endings);
minutiae = [i, j];
%% Visualize minutiae
im3 = repmat(gray_image, 1, 1, 3);
[i,j] = find(imdilate(bifurcation, strel('square',3)));
im3(sub2ind(size(im3), i, j, ones(size(i))*1)) = 1;
im3(sub2ind(size(im3), i, j, ones(size(i))*2)) = 0;
im3(sub2ind(size(im3), i, j, ones(size(i))*3)) = 0;
[i,j] = find(imdilate(endings, strel('square',3)));
im3(sub2ind(size(im3), i, j, ones(size(i))*1)) = 0;
im3(sub2ind(size(im3), i, j, ones(size(i))*2)) = 0;
im3(sub2ind(size(im3), i, j, ones(size(i))*3)) = 1;
figure('Name','Minutiae');
image(im3);
axis image;
```

```
%Show binary image and thinning
binary_image=im2bw(imread(filename));
thin_image=~bwmorph(binary_image,'thin',Inf);
figure;imshow(binary_image);title(filename);
figure;imshow(thin_image);title('Thinned Image');


%% Save output CSV
csvwrite(outname, minutiae);

end
```

(a) *Immagine originale.*

(b) *Impronta digitale binarizzata.*

(c) *Thinning.*



(d) *Estrazione punti minutiae.*

Figura 3.24: *Immagini relative all'esecuzione della funzione "find_minutiae", applicata ad un'impronta digitale, presa dal database FVC2006*

Per una maggiore precisione e per evitare false minutiae è stato successivamente creato un altro algoritmo, che utilizza la funzione "Ginput"; tale funzione, cioè la "funzione *select_minutiae*" visualizza l'immagine, il cui nome è dato come input e consente manualmente di identificare le minutiae, inoltre genera un file in formato CSV che elenca la posizione di tutte le minutiae selezionate. Il CSV "*Read comma-separated value*" è un formato di file basato su file di testo utilizzato per l'importazione ed esportazione (ad esempio da fogli elettronici o database) di una tabella di dati.
Il codice è il seguente:

```
function minutiae = select_minutiae(A, nome)
IMA = imread(A); dimA = size(IMA);
fh = figure;
imshow(A);
hold on;
n = 0;
[xx, yy, button] = ginput(1);
while button == 1
    if (1<=xx && xx<= dimA(1) && 1<=yy && yy<=dimA(2))
        n = n+1;
        x(n) = xx;
        y(n) = yy;
        plot(x(n), y(n), 'r*');
    end
    [xx, yy, button] = ginput(1);
end
% Salvataggio in CSV
if false
    fileID = fopen(nome, 'w');
    for i=1:n-1
        fprintf(fileID, '%d %d\n', x(i), y(i));
    end
    fclose(fileID);
else
    minutiae = [y', x'];
    csvwrite(nome, minutiae);
end
close(fh);
end
```

Dopo che le minutiae sono state trovate, bisogna eseguire la triangolazione, cioè applicare la triangolazione di Delaunay sui punti minutiae, usando le loro coordinate. Viene utilizzata la funzione "$triangulate\_writeGXL$",che, come dice il nome, esegue sia la triangolazione, sia la scrittura in formato GXL del grafo risultante. Per convertire il file nel formato .GXL viene utilizzata la funzione "$create\_GXL$".
La sintassi per eseguire la funzione "$triangulate\_writeGXL$" prevede l'utilizzo di due argomenti:

1. percorso al file che contiene l'impronta;
2. percorso al file che contiene le posizioni delle minutiae.

Inoltre la funzione salva automaticamente:

- il file GXL;
- l'immagine che mostra la triangolazione sopra l'impronta iniziale, in formato $.fig$.

```
function triangulation = triangulate_writeGXL(img_name, minutiae_filename)
if false
    fileID = fopen(minutiae_filename,'r');
    minutiae = fscanf(fileID,'%d %d\r\n');
    fclose(fileID);
else
    minutiae = csvread('minutiae/101_2_minutiae.csv');
end
% Triangulation
triangulation = delaunay(minutiae(:,1), minutiae(:,2));
% Write in GXL
pathstr = 'GXL';
[~, name, ~] = fileparts(minutiae_filename);
output_filename = fullfile(pathstr, [name, '.gxl']);
create_GXL(triangulation, minutiae, output_filename);
% Plot
h = figure;
imshow(img_name);
title('Minutiae and triangulation');
hold on;
triplot(triangulation, minutiae(:,2), minutiae(:,1), 'color', 'blue');
fig_filename = fullfile(pathstr, [name, '.fig']);
savefig(h, fig_filename);

end
```

Una volta creati questi grafi in formato $.GXL$ verranno confrontati mediante l'algoritmo GED "*Graph Edit Distance*".
Questa nuova applicazione verrà spiegata dettagliatamente nel capitolo successivo.

Le immagini seguenti mostrano un esempio dell'applicazione della funzione Ginput e relativa triangolazione di Delaunay, su un'impronta digitale presa dal database FVC2006 del laboratorio di Sistemi Biometrici dell'Università di Bologna.

(a) *Immagine originale di un'impronta digitale.*

(b) *Nella stessa immagine vengono contrassegnate in rosso le minutiae, tramite il Ginput.*

(c) *Triangolazione sui punti minutiae.*

Capitolo 4

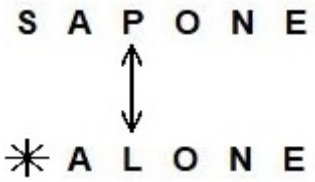
Edit Distance

4.1 Introduzione

L'Edit Distance è stata proposta per risolvere i problemi di confronto fra stringhe (cioè parole).
Quest'applicazione si occupa principalmente di confrontare sequenze differenti l'una dall'altra, usando delle "*edit operations*" avente ciascuna dei costi.
L'Edit Distance viene utilizzata in vari campi ed è infatti abbastanza potente per una vasta gamma di applicazioni, ad esempio è utile nel controllo ortografico automatico, selezionando parole "corrette" da un dizionario interno al software; o nella bioinformatica, in cui quantifica la somiglianza di sequenze di DNA, che possono essere viste come stringhe delle lettere A,G,C e T.
Essa è anche conosciuta con il termine di *distanza di Levenshtein*, il quale nel 1966 definì il più semplice insieme delle edit operations:

- Inserimento di un carattere.
- Cancellazione di un carattere.
- Sostituzione di un carattere.

Se le diverse operazioni hanno costi diversi si parla della "*general Edit Distance*". Altrimenti, se tutte le operazioni costano 1, si parla di "*simple Edit Distance*" o semplicemente *Edit Distance* (ed). In quest'ultimo caso viene cercato semplicemente il numero minimo di inserimenti, cancellazioni e sostituzioni per rendere entrambe le stringhe uguali. Alla luce di queste definizioni, allora, l'Edit Distance tra le due stringhe rappresentanti, ad esempio, le parole "alone" e "sapone" sarà di 2 unità: una sostituzione (da "p" a "l") e una cancellazione ("s").



Si può inoltre osservare che consentendo solo inserimenti e cancellazioni al costo 1, è possibile calcolare la più lunga sottosuccessione comune (LCS) tra due stringhe. Un'altra semplificazione, che ha ricevuto molta attenzione, è la variante che consente solo sostituzioni (distanza di Hamming).
Un'estensione dell'Edit Distance la arricchisce con trasposizioni (cioè una sostituzione della forma $ab \rightarrow ba$ al costo 1). Le trasposizioni sono molto importanti nelle applicazioni di scrittura, perché sono tipici errori di battitura, ma esistono pochi algoritmi per gestirli.
L'Edit Distance è usata per confrontare non solo due stringhe, ma anche due alberi o due grafi. Vedremo nelle sezioni successive, più nel dettaglio, come si comporta quest'applicazione nei vari casi, soffermandomi principalmente sulla "Graph Edit Distance"(GED).
L'applicazione GED può essere calcolata tra grafi diretti o indiretti e viene utilizzata in svariati campi, come ad esempio nella biologia, chimica, informatica o progettazione. Nel mio lavoro utilizzeremo quest'algoritmo per il confronto di impronte digitali.
Tutti i risultati che otterremo attraverso l'utilizzo del GED saranno riportati ed analizzati nel capitolo successivo.

4.2 Edit Distance tra stringhe

Nella seguente sezione analizziamo più nel detteglio come si comporta l'Edit distance tra due stringhe [21] [22].

4.2.1 Confronto tra stringhe

Consideriamo s, x, y, z, v, w stringhe arbitrarie e $a, b, c, ...$ lettere dell'alfabeto Σ^*.
Sia $s \in \Sigma^*$, denotiamo la sua lunghezza come $|s|$ e sia s_i l'i-esimo carattere di s, per un intero $i \in 1...|s|$.
Sia $s_{i...j} = s_i s_{i+1} ... s_j$ con $i > j$. Inoltre indichiamo con ε una stringa vuota.
Più formalmente:
Sia Σ^* un alfabeto finito di dimensione $|\Sigma^*| = \sigma$.
Sia $T \in \Sigma^*$ un testo di lunghezza $n = |T|$. Sia $P \in \Sigma^*$ un *pattern* di lunghezza $m = |P|$.
Sia $k \in \mathbb{R}$ l'errore massimo.
Sia $d : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R}$ la funzione distanza. Allora , dati T, P, k e $d(.)$ esiste i tale che $d(P, T_{i..j}) < k$.
La distanza $d(x, y)$ tra due stringhe x e y, è il costo minimo di una sequenza di operazioni che trasformano x in y. Il costo di una sequenza di operazioni è la somma dei costi delle singole operazioni. Le operazioni sono un insieme finito di regole della forma $\delta(z, w) = t$ dove z e w sono stringhe diverse e t è un numero reale non negativo. Una volta che l'operazione ha convertito una sottostringa z in w , non può essere eseguita nessun'altra operazione su w. Se per ogni operazione della forma $\delta(z, w)$ esiste la rispettiva operazione $\delta(w, z)$ allo stesso costo, allora la distanza è simmetrica, cioè $d(x, y) = d(y, x)$. Si noti anche che:

- $d(x, y) \geq 0$ per tutte le stringhe x e y, queste richiederebbe almeno un'operazione a costo non-zero.
- $d(x, x) = 0$,quindi ogni stringa può essere trasformata banalmente in se stessa usando zero operazioni.
- $d(x, z) \leq d(x, y) + d(y, z)$, dalla disuguaglianza triangolare.

Quindi se la distanza è simmetrica, lo spazio delle stringhe forma una spazio metrico.
Nelle varie applicazioni, l'insieme delle possibili operazioni è il seguente:

- *Inserimento*: $\delta(\epsilon, a)$, cioè inserire la lettera a.
- *Cancellazione*: $\delta(a, \epsilon)$, cioè cancellare la lettera a.

- *Sostituzione* o *Rimpiazzamento*: $\delta(a,b)$ per $a \neq b$, cioè sostituire a con b.

- *Trasposizione*: $\delta(ab,ba)$ per $a \neq b$, cioè scambio le lettere adiacenti a e b.

Le più comuni *funzioni distanza* usate sono le seguenti:

- *Levenshtein o Edit Distance*: Consente inserimenti, cancellazioni, sostituzioni. Nella versione semplificata, tutte le operazioni costano 1. Può essere definita come "il minimo numero di inserimenti, cancellazioni e sostituzioni per rendere due stringhe uguali". Inoltre la distanza è simmetrica e si ha : $0 \leq d(x,y) \leq max(|x|,|y|)$.

- *Hamming distance*: Consente solo sostituzioni, con costo 1 nella versione semplificata. La distanza è simmetrica e si ha: $0 \leq d(x,y) \leq |x|$.

- *Longest common susequence distance*: Consente solo inserimenti e cancellazioni, tutte al costo 1. La distanza corrisponde al numero dei caratteri spaiati, è simmetrica e sia ha $0 \leq d(x,y) \leq |x| + |y|$.

Notiamo che se la distanza di Hamming o l'Edit Distance vengono utilizzate, allora il problema ha senso per $0 < k < m$. Se $k = 0$ corrisponde alla corrispondenza esatta tra le stringhe. Perciò si può definire $\alpha = k/m$ livello d'errore, con $0 < \alpha < 1$. Questo valore dà un'idea di " *error ratio* " durante il matching.

Osserviamo inoltre che, date due sequenze X e Y, un *allineamento*, è una corrispondenza tra sottostringhe delle due sequenze.
Occorre trovare un allineamento ottimo delle due che corrisponda alla minima distanza tra X e Y, ove la distanza è definita secondo le regole seguenti:

- si ammette che nelle due sequenze possano essere inseriti spazi indicati con $*$ e che,per ogni posizione nelle due sequenze, il carattere o lo spazio che appare nella X sia posto in corrispondenza (allineato) con il carattere o lo spazio nella stessa posizione della Y;

- la distanza è la somma delle distanze tra coppie di caratteri o spazi, uno in X e l'altro in Y, in posizioni corrispondenti nell'allineamento: caratteri uguali hanno distanza zero, indicata con $\emptyset$ (*match*); caratteri diversi hanno distanza 1 (*mismatch*); un carattere allineato a uno spazio ha distanza 1 (space);

- la presenza di uno spazio si interpreta come l'avvenuta cancellazione di un carattere dalla sequenza, o come l'inserzione di un carattere nell'altra in posizio-

ne corrispondente: per questo motivo non si considerano spazi in posizioni corrispondenti di X,Y;

- l'edit distance tra X e Y è la distanza relativa all'allineamento (o agli allineamenti) che minimizza tale distanza.

Ad esempio, si considerino le sequenze alfabetiche:

X = INTENTION e Y = EXECUTION

Per trasformare una stringa nell'altra occorrono 5 operazioni: cancellazione della "i", sostituire la "e" con la "n", sostituire la "x" con la "t", inserire la "c", sostituire la "u" con la "n" , vedi ***Figura 4.1***.

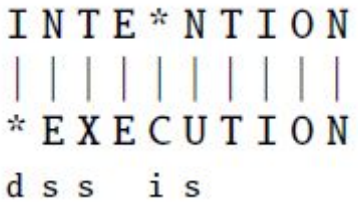


Figura 4.1: Rappresentazione della " minimum edit distance" come un allineamento tra stringhe. Dove la lettera d indica la cancellazione, la s indica la sostituzione e la lettera i indica l'inserimento

Come già osservato più volte, ognuna di queste operazioni ha un costo; assumendo che ognuna di esse abbia costo 1 ed assumendo che la sostituzione di una lettera con sè stessa, per esempio "t" con "t", abbia costo zero, l'Edit Distance tra le due stringhe "intention" ed "execution" ha costo 5.
Inoltre la distanza di Levenshtein ha proposto una versione alternativa in cui ogni inserimento o cancellazione ha un costo di 1 e le sostituzioni non sono consentite; in questo caso l'Edit Distance o distanza di Levenshtein tra le due stringhe precedenti ha un costo di 8.
Più formalmente, per qualsiasi linguaggio L e stringa x su un alfabeto Σ^*, l'Edit Distance $d(L,x)$ è data da:

$$d(L,x) = \min_{y \in L} d(x,y)$$

dove $d(x,y)$ è la stringa Edit Distance.

Alcune osservazioni sugli algoritmi

Esistono in letteratura molti algoritmi che calcolano l'Edit Distance tra stringhe [22], quindi il path più breve per trasformare una stringa in un'altra, vedi ***Figura 4.2***.

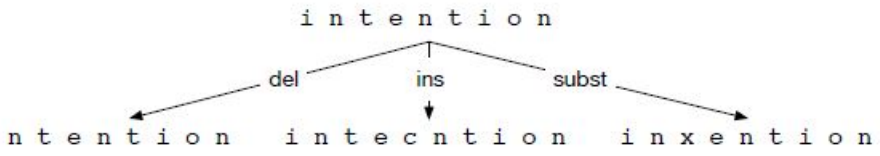


Figura 4.2: Un esempio che mostra come ricercare l'Edit Distance

Si potrebbe utilizzare la cosiddetta "*programmazione dinamica*", introdotta da Bellman nel 1957.
L'intuizione di un problema di programmazione dinamica è che un grosso problema può essere risolto combinando correttamente le soluzioni a vari sotto-problemi.

Nella sua autobiografia, Bellman (1984) spiega come ha originariamente ideato il termine programmazione dinamica:

"... Gli anni '50 non erano anni buoni per la ricerca matematica. Il Segretario alla Difesa ... aveva una paura patologica e l'odio per la parola, la ricerca ... ho deciso quindi di usare la parola "programmazione". Volevo convincere che questo era dinamico, che era un palcoscenico multiplo ... ho pensato, prendiamoci una parola che ha un significato assolutamente preciso, cioè dinamico ... è impossibile usare la parola, dinamica, in senso dispregiativo. Prova a pensare ad una combinazione che potrebbe dargli un significato dispregiativo. È impossibile. Quindi, pensavo che la programmazione dinamica fosse un buon nome. Era qualcosa che nemmeno un deputato poteva obiettare ".

Si consideri il percorso più breve delle parole trasformate che rappresenta la minima Edit Distance tra le stringhe "intention" e "execution" [22], vedi ***Figura 4.3***.

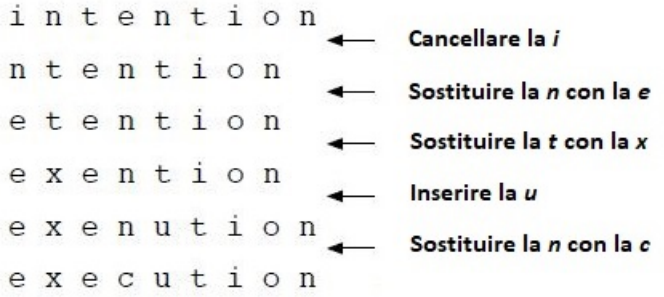


Figura 4.3: Path tra intention ed execution

Prendendo una stringa, ad esempio "exention" che si trova nel percorso ottimale, allora l'intuizione della programmazione dinamica è che se "exection" si trova nell'elenco delle operazioni ottimali,la sequenza ottimale deve includere anche il percorso ottimale da "intention" a "exention".

Il primo algoritmo di programmazione dinamica è stato ideato da Wagner e Fisher nel 1974 [22].
Date due stringhe, la stringa sorgente ("source") X di lunghezza n e la stringa di destinazione ("target")Y di lunghezza m, definiremo D(i,j) come l'Edit Distance tra $X[1..i]$ e $Y[1..j]$. L'Edit Distance tra X e Y è perciò D(n,m).
D(n,m) viene definita per ricorrenza, usando la programmazione dinamica, nel modo seguente [22]

$$D[i,j] = min \begin{cases} D[i-1,j] + del-cost(source[i]) \\ D[i,j-1] + ins-cost(target[j]) \\ D[i-1,j-1] + sub-cost(source[i], target[j]) \end{cases}$$

Se si considera la versione della distanza di Leveneshtein, nella quale l'inserimento e la cancellazione ha costo 1 (ins-cost(.)= del-cost(.) = 1) e la sostituzione ha costo 2

(eccetto la sostituzione di un carattere con se stesso che ha costo 0) il calcolo di D[i,j] diventa [22]

$$D[i,j] = min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1] + \begin{cases} 2, & \text{se } source[i] \neq target[j] \\ 0, & \text{se } source[i] = target[j] \end{cases} \end{cases}$$

Tale algoritmo ha una complessità temporale di $O(mn)$. Una volta costruita la programmazione dinamica completa, la sua complessità spaziale è ancora $O(mn)$; questo può essere migliorato a $O(min(m,n))$. Tuttavia, questa ottimizzazione rende difficile leggere la serie minima di edit operations. Una soluzione di spazio lineare a questo problema è offerta dall'*algoritmo di Hirschberg*, dal nome del suo inventore, Dan Hirschberg; esso è un algoritmo di programmazione dinamica che trova l'allineamento ottimale della sequenza tra due stringhe. L'ottimizzazione viene misurata con la distanza di Levenshtein, definita come la somma dei costi delle inserzioni, delle sostituzioni, delle eliminazioni e delle azioni nulle necessarie per cambiare una stringa nell'altra.
L'algoritmo di Hirschberg è comunemente usato nella biologia computazionale per trovare i massimi allineamenti globali di sequenze di DNA e proteine.
Conoscere la minima Edit Distance è utile non solo nella biologia computazionale, ma anche nella correzione di errori ortografici o nell'elaborazione del linguaggio e del parlato nel riconoscimento vocale. L'allineamento gioca un ruolo importante nella traduzione automatica, in cui le frasi devono essere abbinate tra loro. Invece di considerare l'Edit Distance tra una stringa e l'altra, l'Edit Distance del lessico è il minimo valore dell'Edit Distance tra una stringa fissa e una qualsiasi stringa presa da un insieme di stringhe.

4.3 Edit Distance tra alberi

Gli alberi sono tra le strutture più studiate nell'informatica. In particolare, il problema del confronto degli alberi si verifica in diverse aree, come la biologia computazionale, l'analisi delle immagini, la dimostrazione automatica dei teoremi e l'ottimizzazione del compilatore. Ad esempio, nella biologia computazionale, il calcolo della somiglianza tra gli alberi viene utilizzato nel confronto della struttura dell'RNA.

4.3.1 Confronto tra alberi

Sia T un albero radicato [23]. Allora T è:

- *Labeled tree*: se ad ogni nodo è assegnato un simbolo dell'alfabeto finito Σ
- *Ordered tree*: se viene dato un ordine da sinistra a destra tra i fratelli in T.

Se T è un "*ordered tree*" vengono definite le seguenti operazioni:

- *Rimpiazzamento*: Cambio l'indice di un nodo v in T.
- *Cancellazione*: Cancello il nodo v in T con genitore v' facendo diventare i figli di v figli di v'.
- *Inserimento*: Il complementare della cancellazione. Inserisco un nodo v come un figlio di v' in T, diventando v genitore di una sottosequenza consecutiva dei figli di v'.

Per gli alberi non ordinati è possibile definire le operazioni in modo simile. In questo caso, inserire ed eliminare operazioni su un sottoinsieme anziché su una sottosequenza.

Siano T_1 e T_2 due alberi indicizzati (*labeled tree*) ordinati o non ordinati. Assumiamo che su ogni operazione venga assegnato un costo. Si definisce un "*edit script*" S tra T_1 e T_2 una sequenza di edit operations che trasformano T_1 in T_2.
Il costo di S è la somma dei costi delle operazioni in S. Un "*optimal edit script*" rappresenta il costo minimo di edit script tra T_1 e T_2 e questo costo è il "tree edit distance", denotato con $\delta(T_1, T_2)$.
Il "*tree edit distance problem*" consiste nel calcolare l'edit distance ed il corrispondente edit script.
Più formalmente, siano T_1 e T_2 due alberi indicizzati. Rappresentiamo ciascuna edit operarion con $(l_1 \longrightarrow l_2)$, dove $(l_1, l_2) \in (\Sigma_\lambda \times \Sigma_\lambda) \setminus (\lambda, \lambda)$.

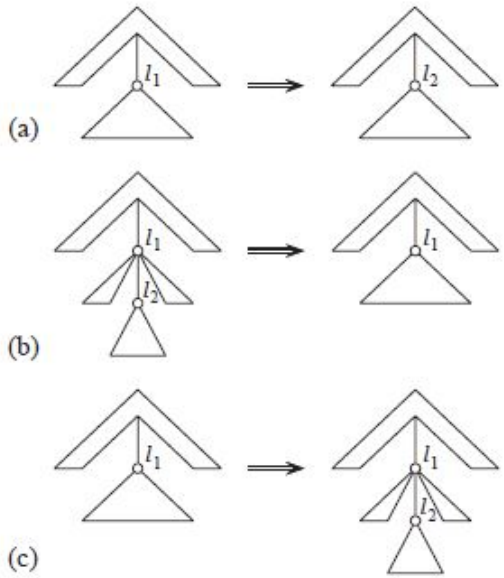


Figura 4.4: *Tree edit operations: (a) Sostituzione di un nodo l_1 con un nodo l_2 (b) Cancellazione del nodo l_2 (c) Inserimento del nodo l_2 come figlio del nodo l_1*

L'operazione è una sostituzione se $l_1 \neq \lambda$ e $l_2 \neq \lambda$, una cancellazione se $l_2 = \lambda$ ed un inserimento se $l_1 = \lambda$.
Estendiamo la notazione tale che $(v \longrightarrow w)$ per i nodi v e w con $(label(v) \longrightarrow label(w))$.
In questo caso v o w potrebbero essere λ. Sia ora γ una funzione costo, si definisce un'edit operation $\gamma(l_1 \longrightarrow l_2) = \gamma(l_1, l_2)$. Il costo di una sequenza $S = s_1, ..., s_k$ di operazioni è dato da $\gamma(S) = \sum_{i=1}^{k} \gamma(s_i)$.
L'edit distance $\gamma(T_1, T_2)$ tra T_1 e T_2 è definita come:

$$\gamma(T_1, T_2) = min\ \{\gamma(S) | S \text{ è una sequenza di operazioni che trasformano } T_1 \text{ in } T_2\}$$

Poichè γ è una distanza metrica, anche δ diventa una distanza metrica.

Una "*edit distance mapping*" tra T_1 e T_2 è una rappresentazione delle operazioni di modifica, che viene utilizzata in molti algoritmi per il problema della *tree edit distance*, cioè l'edit distance tra alberi.

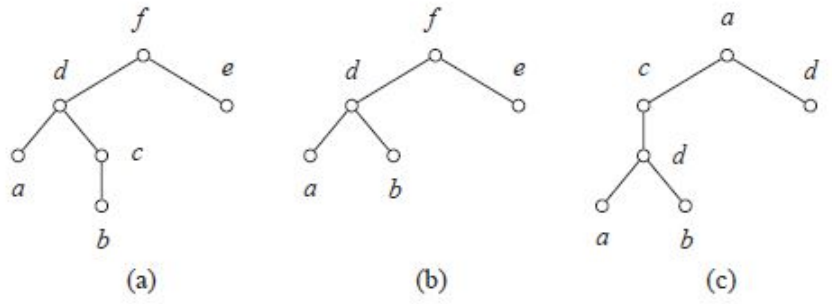


Figura 4.5: *Trasformazione dell'albero (a) nell'albero (c).*
(a) Un albero (b) L'albero dopo la cancellazione del nodo "c" (c) L'albero dopo aver inserito il nodo "c" e sostituito "f" con "a" e "e" con "d"

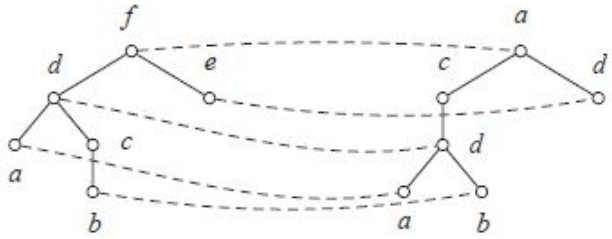


Figura 4.6: *Mappatura corrispondente all'edit script della figura precedente*

Formalmente, si consideri la tripla (M, T_1, T_2) una *mapping edit distance ordinata* da T_1 a T_2, se $M \subseteq V(T_1) \times V(T_2)$ e per ogni coppia $(v_1, w_1), (v_2, w_2) \in M$:

1. $v_1 = v_2$ se e solo se $w_1 = w_2$
2. v_1 è l'antenato di v_2 se e solo se w_1 è l'antenato di w_2
3. v_1 è alla sinistra di v_2 se e solo se w_1 è alla sinistra di w_2

Siano N_1 e N_2 l'insieme dei nodi in T_1 e T_2 rispettivamente, il costo di M è dato da:

$$\gamma(M) = \sum_{(v,w)\in M} \gamma(v \longrightarrow w) + \sum_{v\in N_1} \gamma(v \longrightarrow \lambda) + \sum_{w\in N_2} \gamma(\lambda \longrightarrow w)$$

La mappatura può essere composta. Siano $T_1, T2, T_3$ alberi indicizzati. Siano M_1 e M_2 mappature da T_1 a T_2 e da T_2 a T_3 rispettivamente. Si definisce:

$$M_1 \circ M_2 = \{(v,w) \mid \exists u \in V(T_2 \text{ tale che } (v,u) \in M_1\, e\, (u,w) \in M_2\}.$$

Da ciò segue facilmente che $M_1 \circ M_2$ stesso diviene una mappatura da T_1 a T_3. Quindi il minimo costo della mappatura è equivalente all'edit distance:

$$\gamma(T_1,T_2) = min\{\gamma(M) \mid (M,T_1,T_2) \text{è una edit distance mapping }\}.$$

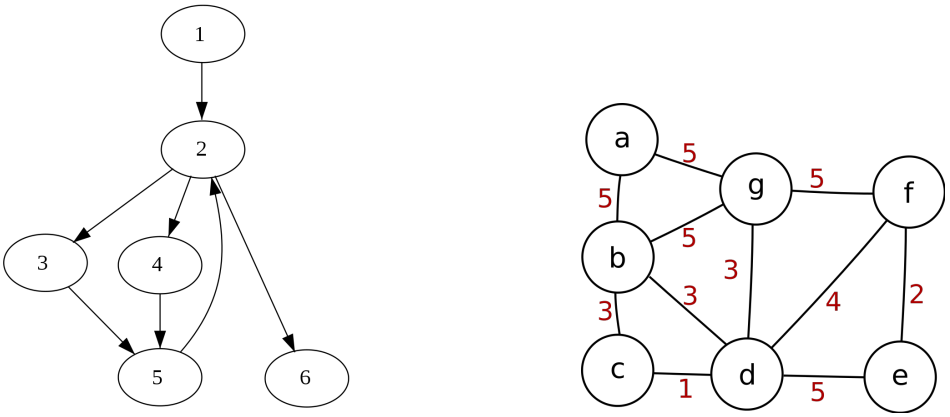
In altre parole, per calcolare l'edit distance, è possibile calcolare il "*minimum cost mapping*".

4.4 Edit Distance tra grafi: GED

Un grafo è una struttura dati costituita da oggetti semplici, ovvero i vertici (nodi) e da collegamenti tra i vertici.
Tali collegamenti possono essere:

- orientati: dotati di una direzione e di un verso; in questo caso sono chiamati archi ed il grafo è detto "*orientato*".
- non orientati: dotati di una direzioni, ma non di un verso; in questo caso sono chiamati spigoli ed il grafo è detto "*non orientato*".
- dati associati a nodi e/o collegamenti: in questo caso si ha un grafo "*pesato*", in cui si può associare ad ogni arco un valore o peso, ad esempio un'etichetta simbolica o un attributo numerico (costo, capacità, lunghezza, etc.).

Solitamente un grafo viene raffigurato sul piano da punti o cerchi, che rappresentano i nodi; i collegamenti tra i vertici sono rappresentati da segmenti o curve che collegano due nodi; nel caso di un grafo orientato, il verso degli archi è indicato da una freccia.



(a) *Grafo orientato.* (b) *Grafo pesato.*

Figura 4.7: *Esempi di grafi.*

Per i grafi pesati, sia i vertici che i bordi possono essere caratterizzati da attributi che possono variare da etichette numeriche, a descrizioni più complesse come stringhe o vettori [24]. Questa disposizione porta a delle rappresentazioni molto complesse che vengono utilizzate in molti domini applicativi.
Confrontare tali grafi è un problema cruciale per il riconoscimento di pattern basato su grafi. Perciò, per far fronte a questo problema, sono stati proposti in letteratura un enorme numero di algoritmi.

Esistono due metodi principali: "*embedding-based method*" e "*matching-based method*".
Nel primo metodo, l'idea chiave è quella di proiettare i grafi di input da confrontare in uno spazio vettoriale. Mentre nel secondo metodo il confronto tra due grafi richiede il calcolo e la valutazione della corrispondenza "migliore" tra di essi; in questo caso il matching deve essere tollerante agli errori, cioè deve tollerare differenze nella topologia.
Un "*error-tolerant matching-based method*" usato per confrontare due grafi è il GED ovvero il *Graph Edit Distance*.
In questo metodo vengono introdotte un insieme di "*edit operation*" ed ognuna è caratterizzata da un costo; il GED è il costo totale della sequenza di operazioni meno costosa che trasforma un grafo nell'altro.
Nel GED non si applica nessuna restrizione e può essere applicato a qualsiasi tipo di grafo, inclusi gli ipergrafi. Come già osservato esso è stato utilizzato in molte applicazioni, ad esempio nel rilevamento di malware , nella chemioinformatica o nell'analisi di documenti.
Un possibile problema del GED è la sua complessità computazionale dal momento che è noto per essere NP-completo. Il calcolo dell'esatto GED è esponenziale al numero di nodi ed è fattibile solo per grafi di dimensioni piuttosto piccole (tipicamente 10 nodi). Per superare questo limite, sono stati proposti svariati metodi nell'ultimo decennio, alcuni basati su metodi più veloci ed ottimali che approssimano l'esatto GED.
Ad esempio viene utilizzata la " *Binary Linear Programming*" (BLP), ovvero vengono utilizzati dei problemi lineari binari, che sono restrizioni di problemi lineari interi (ILP).

4.4.1 Definizioni e proprietà

Definizione 4.4.1. *Un grafo G è una 4-tupla* $G = (V, E, \mu, \xi)$ *dove:*

- V *è un insieme di vertici;*
- E *è un insieme di archi, tale che* $\forall e = (i, j) \in E,\ i \in V$ *e* $j \in V$
- $\mu : V \to L_V$ *è una funzione che contrassegna i vertici e quindi associa un indice* $\mu(v)$ *a* $v \in V$*, dove* L_V *è l'insieme degli indici per i vertici;*
- $\xi : E \to L_E$ *è una funzione che contrassegna gli archi e quindi associa un indice* $\xi(e)$ *a* $e \in E$*, dove* L_E *è l'insieme degli indici per gli archi.*

Un grafo G è detto *semplice* se non presenta loop (un arco che collega un vertice con se stesso) e non presenta multiarchi (più archi tra gli stessi vertici) [25]. In questo caso $E \subseteq \{(i, j) \in V \times V | i \neq j\}$. Altrimenti G è un *multigrafo* e E è un *multiinsieme*.
Un grafo G è detto *indiretto* se la relazione E è simmetrica, cioè se i suoi archi non sono orientati. In questo caso, $\forall (i, j) \in E$, $(j, i) \in E$ e $(i, j) = (j, i)$. Altrimenti G è un grafo *diretto*.

Il GED è comunemente usato per misurare la differenza tra due grafi ed è tollerante agli errori [25]. Esso definisce la diversità tra due grafi attraverso il minimo costo di trasformazione richiesto per modificare un grafo in un altro.

Definizione 4.4.2. *L'Edit Distance per grafi, d(.,.) è una funzione:*

$$d\colon G \times G \to \mathbb{R}_+$$
$$(G_1, G_2) \mapsto d(G_1, G_2) = \min_{(\sigma_1,\ldots,\sigma_k)\in\Gamma(G_1,G_2)} \sum_{i=1}^{k} c(o_i)$$

dove $G_1 = (V_1, E_1, \mu_1, \xi_1)$ e $G_2 = (V_2, E_2, \mu_2, \xi_2)$ sono due grafi e $\Gamma(G_1, G_2)$ è l'insieme di tutte le edit path $o = (o_1, \ldots, o_k)$ che servono per trasformare G_1 in G_2.

Le edit operations o_i sono:

- sostituizione di un vertice con un altro: $(v_1 \rightarrow v_2)$;
- sostituzione di un arco con un altro: $(e_1 \rightarrow e_2)$;
- cancellazione di un vertice: $(v_1 \rightarrow \epsilon)$;
- cancellazione di un arco: $(e_1 \rightarrow \epsilon)$;
- inserimento di un vertice: $(\epsilon \rightarrow v_2)$;
- inserimento di un arco: $(\epsilon \rightarrow e_2)$;

con $v_1 \in V_1$, $v_2 \in V_2$, $e_1 \in E_1$, e $e_2 \in E_2$.

Qui, ϵ è un falso vertice o arco che è usato solo per simulare una cancellazione o un inserimento. Inoltre $c(.)$ è una funzione che associa un costo ad ogni edit operation o_i.

Figura 4.8: Una possibile edit path tra i grafi g_1 e g_2

Come abbiamo già osservato, esistono in letteratura molti algoritmi per la risoluzione di problemi GED, alcuni esatti ed altri approssimativi. Vediamo in generale alcuni degli approcci esistenti.
La prima famiglia di calcolo esatto del GED è basata sull'algoritmo A^* ampiamente noto. L'algoritmo è stato descritto da Peter Hart, Nils Nilsson e Bertram Raphael nel 1968 [26], è un algoritmo di ricerca su grafi che individua un percorso da un dato nodo iniziale ad uno finale. Esso utilizza una stima euristica che classifica ogni nodo attraverso una stima della strada migliore che passa attraverso tale nodo.
L'algoritmo A^* ha una stima ottimale del costo del percorso attraverso ogni nodo considerato; tutto è basato su quanto A^* "conosce".
A^* non trascurerà mai la possibilità di trovare un percorso dal costo minore e quindi sarà ammissibile. Inoltre A^* considera il numero di nodi più basso di qualunque altro algoritmo di ricerca ammissibile, che utilizza una funzione euristica non molto più accurata di quella adottata da A^*.
In altre famiglie di algoritmi, il GED viene calcolato risolvendo un BLP, ovvere un problema di programmazione lineare binaria. Ad esempio, Justice ed Hero [27], hanno proposto recentemente una nuova formulazione del BLP per risolvere tali problemi, in cui si ricerca la matrice di permutazione che minimizza il costo per trasformare il grafo G_1 nel grafo G_2, con G_1 e G_2 due grafi non pesati e non orientati. Si ha quindi:

$$d(G_1, G_2) = \min_P \sum_{i=1}^{n} \sum_{j=1}^{n} c\left(l(A_1^i), l(A_2^j)\right) P_{i,j} + \frac{1}{2} c(0,1) \mid A_1 - PA_2P^T \mid^{ij}$$

Dove A_k è la matrice delle adiacenze di G_k e P è una matrice ortogonale di permutazione tale che $PP^T = P^TP = I$. Inoltre $l(A_k^i)$ è l'indice dell'i-esimo vertice in G_k e c è la funzione costo.
Per trasformare il problema di ottimizzazione non lineare in un problema lineare è stata usata una trasformazione lineare.
In questa formulazione il numero di vincoli e variabili cresce in maniera quadratica con il numero dei vertici dei grafi, perciò un problema potrebbe essere legato alla memoria del programma. Inoltre la trasformazione di grafi mediante una matrice di adiacenza limita la formulazione all'elaborazioni di grafi semplici.
Si può osservare inoltre che il GED può essere eseguito in un tempo ragionevole solo per piccoli grafi. Una soluzione potrebbe essere quella di ridurre il problema GED ad un problema di Assegnazione lineare ("*Linear Sum Assignment Problem*"), ovvero problemi di ricerca operativa in cui bisogna assegnare diverse attività in maniera ottimale.

Questo genere di problema può essere risolto utilizzando il cosiddetto "Hungarian method" (Munkres assignment algortihm, metodo di ottimizzazione combinatoria, sviluppato da Harold Kuhn nel 1955, detto "ungherese" in quanto basato su lavori di Dènes Konig e Jeno Eqervary) in un tempo polinomiale $O(n^3)$, dove n è la dimensione della matrice dei costi riguardanti i vertici.

4.4.2 Graph Edit Distance utilizzando BLP

Come già osservato, il GED può essere risolto utilizzando problemi di programmazione lineare binaria (BLP), che sono delle restrizioni dei problemi di programmazione lineare intera (ILP) in cui le variabili sono binarie [24]. Quindi in generale si ha la seguente formulazione:

$$\min_x c^T x \qquad (2a)$$

$$\text{subject to} Ax \leq b \qquad (2b)$$

$$x \in \{0,1\}^n \qquad (2c)$$

dove $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$ e $b \in \mathbb{R}^m$.

Una possibile soluzione è un vettore x di n variabili binarie $(2c)$ che rispetta i vincoli della disuguaglianza lineare $(2b)$.

Innanzitutto diamo delle definizioni più dettagliate riguardo le variabili e le funzioni di costo utilizzate per l'applicazione del GED tra due grafi $G_1 = (V_1, E_1, \mu_1, \xi_1)$ e $G_2 = (V_2, E_2, \mu_2, \xi_2)$.
Le operazioni di edit che servono per trasformare il grafo G_1 nel grafo G_2 sono:

(i) la sostituzione di un vertice (o un lato) di G_1 con un vertice (o un lato) di G_2;

(ii) la cancellazione di un vertice (o un lato) da G_1;

(iii) l'inserimento di un vertice (o un lato) in G_1.

Nella tabella sottostante viene definito un insieme di variabili binarie per ciascuna delle operazioni di edit.

Operazioni di edit	Variabili	Costi
Sostituzione del vertice i con il vertice k	$x_{i,k}$	$c_{i,k}$
Cancellazione del vertice i	u_i	$c_{i,\varepsilon}$
Inserimento del vertice k	v_k	$c_{\varepsilon,k}$
Sostituzione del lato ij con il lato kl	$y_{ij,kl}$	$c_{ij,kl}$
Cancellazione del lato ij	e_{ij}	$c_{ij,\varepsilon}$
Inserimento del lato kl	f_{kl}	$c_{\varepsilon,kl}$

Ad esempio, se si considerano V_1 e V_2 due insiemi di vertici, si potrebbe sostituire un vertice i di V_1 con k vertice di V_2 e ciò ha un "edit cost" di $c_{i,k}$. Inoltre se un vertice i è sostituito con un vertice k la variabile $x_{i,k}$ è uguale ad 1 ed è uguale a 0 altrimenti.

In base a tali notazioni, si può definire un "edit path" tra G_1 e G_2 come una 6-tupla (x, y, u, v, e, f) dove:
$\mathbf{x} = (x_{i,k})_{(i,k)\in V_1\times V_2}$, $\mathbf{y} = (y_{ij,kl})_{(ij,kl)\in E_1\times E_2}$, $\mathbf{u} = (u_i)_{i\in V_1}$, $\mathbf{e} = (e_{ij})_{ij\in E_1}$, $\mathbf{v} = (v_k)_{k\in V_2}$, $\mathbf{f} = (f_{kl})_{kl\in E_2}$.
Per calcolare il costo totale di un edit path, devono essere definiti inizialmente i costi elementari di ciascuna operazione di edit. Anche le notazioni dei costi, come per le variabili, sono riportate nella tabella precedente.

La funzione obiettivo da minimizzare è il costo globale indotto da un edit path (x, y, u, v, e, f) che trasforma il grafo G_1 nel grafo G_2:

$$C(x,y,u,v,e,f) = \sum_{i\in V_1}\sum_{k\in V_2} c_{i,k}\cdot x_{i,k} + \sum_{ij\in E_1}\sum_{kl\in E_2} c_{ij,kl}\cdot y_{ij,kl} + \sum_{i\in V_1} c_{i,\varepsilon}\cdot u_i + \sum_{k\in V_2} c_{\varepsilon,k}\cdot v_k + \sum_{ij\in E_1} c_{ij,\varepsilon}\cdot e_{ij} + \sum_{kl\in E_2} c_{\varepsilon,kl}\cdot f_{kl} \tag{4.4.1}$$

Vengono poi assegnati dei vincoli, i quali garantiscono che le soluzioni ammissibili dei problemi di programmazione lineare binaria (BLP) siano "edit path" che trasformano il grafo G_1 nel grafo G_2.

Un edit path è considerato ammissibile se e solo se sono soddisfatte le seguenti condizioni:

(i) Fornisce un mappatura 1 a 1 tra un sottoinsieme dei vertici di G_1 e un sottoinsieme dei vertici di G_2. Questa mappatura è equivalente alla sostituzione di un vertice. Più specificatamente si ha:

$$u_i + \sum_{k \in V_2} x_{i,k} = 1 \; \forall i \in V_1 \tag{4.4.2}$$

$$v_k + \sum_{i \in V_1} x_{i,k} = 1 \; \forall k \in V_2 \tag{4.4.3}$$

Il vincolo (4.4.2) garantisce che ogni vertice di G_1 venga mappato in un esatto vertice di G_2 o cancellato da G_1, mentre il vincolo (4.4.3) garantisce che ciascun vertice di G_2 venga mappato in un esatto vertice di G_1 o inserito in G_1.

(ii) Fornisce una mappatura 1 a 1 tra un sottoinsieme dei lati di G_1 e un sottoinsieme dei lati di G_2. Questa mappatura è equivalente alla sostituzione di un lato.
Più specificatamente si ha:

$$e_{ij} + \sum_{kl \in E_2} y_{ij,kl} = 1 \; \forall ij \in E_1 \tag{4.4.4}$$

$$f_{kl} + \sum_{ij \in E_1} y_{ij,kl} = 1 \; \forall kl \in E_2 \tag{4.4.5}$$

I vincoli (4.4.4) e (4.4.5) garantiscono una valida mappatura tra i lati.

(iii) Le mappature dei vertici e le mappature dei lati sono coerenti, vale a dire, la topologia del grafo è rispettata.

Il vincolo (iii) è dato dalla seguente proposizione:

Proposizione 4.4.3. *Un arco $ij \in E_1$ può essere mappato in un arco $kl \in E_2$ solo se i vertici (testa) $i \in V_1$ e $k \in V_2$ e i vertici (coda) $j \in V_1$ e $l \in V_2$ sono rispettivamente mappati.*

Tale vincolo viene espresso linearmente con le seguenti disuguaglianze:

$$y_{ij,kl} \leqslant x_{i,k} \; \forall (ij,kl) \in E_1 \times E_2 \tag{4.4.6}$$

$$y_{ij,kl} \leqslant x_{j,l} \; \forall (ij,kl) \in E_1 \times E_2 \tag{4.4.7}$$

Ad esempio, sia $i \in V_1$ tale che i è cancellata da G_1 ($u_i = 1$). Usando l'equazione (4.4.2) si ha che $\forall k \in V_2$, $x_{i,k} = 0$. Inoltre usando l'equazione (4.4.6) $\forall j \in V_1$ tale che $ij \in E_1$ e $\forall kl \in E_2$, $y_{ij,kl} \leqslant x_{i,k} = 0$ ed usando l'equazione (4.4.4), $e_{ij} = 1$, che cancella il lato ij. Di conseguenza, se $i \in V_1$ viene cancellato da G_1, allora tutti i lati $ij \in E_1$ sono cancellati da G_1. Allo stesso modo, se $k \in V_2$ viene inserito in G_1, allora tutti i lati $kl \in E_2$ sono inseriti in G_1.

Unendo le equazioni precedenti con i vincoli che assicurano una soluzione costituita da variabili binarie, si ottiene una versione semplificata della formulazione BLP, detta $F1$.
Tale formulazione ha $|V_1| + |V_2| + |E_1| + |E_2| + |V_1| \cdot |V_2| + |E_1| \cdot |E_2|$ variabili e $|V_1| + |V_2| + |E_1| + |E_2| + 2 \cdot |E_1| \cdot |E_2|$ vincoli.
Vediamo ora una seconda formulazione esatta del GED, detta $F2$, ottenuta utilizzando la formulazione del $F1$, perciò è teoricamente equivalente a $F1$, ma presenta un minor numero di variabili e vincoli.

Si considerino i seguenti nuovi vincoli:

$$\sum_{k \in V_2} x_{i,k} \leq 1 \qquad \forall i \in V_1 \tag{4.4.8}$$

$$\sum_{i \in V_1} x_{i,k} \leq 1 \qquad \forall k \in V_2 \tag{4.4.9}$$

$$\sum_{kl \in E_2} y_{ij,kl} \leq 1 \qquad \forall ij \in E_1 \tag{4.4.10}$$

$$\sum_{ij \in E_1} y_{ij,kl} \leq 1 \qquad \forall kl \in E_2 \tag{4.4.11}$$

La nuova funzione obiettivo diventa:

$$C'(x,y) = \sum_{i \in V_1} \sum_{k \in V_2} (c_{i,k} - c_{i,\epsilon} - c_{\epsilon,k}) \cdot x_{i,k} + \sum_{ij \in E_1} \sum_{kl \in E_2} (c_{ij,kl} - c_{ij,\epsilon} - c_{\epsilon,kl}) \cdot y_{ij,kl} + \gamma \tag{4.4.12}$$

dove:

$$\gamma = \sum_{i \in V_1} c_{i,\epsilon} + \sum_{k \in V_2} c_{\epsilon,k} + \sum_{ij \in E_1} c_{ij,\epsilon} + \sum_{kl \in E_2} c_{\epsilon,kl}.$$

Quest'ultima equazione mostra che il GED può essere ottenuto senza calcolare esplicitamente le variabili u, v, e, f.

Nella formulazione $F1$ il numero di vincoli topologici è $|E_1| \cdot |E|$.

Pertanto, il numero di vincoli aumenta in modo quadratico con la densità dei grafi. Si può dimostrare che è possibile formulare il problema GED con potenzialmente meno vincoli, lasciando invariato l'insieme delle soluzioni.
Innanzitutto occorre modificare la **proposizione 4.4.3**, considerando i nuovi vincoli:

- Dato un lato $ij \in E_1$ e un vertice $k \in V_2$, esiste al massimo un lato il cui vertice iniziale è k che può essere mappato con ij:

$$\sum_{kl \in E_2} y_{ij,kl} \leq x_{i,k} \qquad \forall k \in V_2, , \forall ij \in E_1 \tag{4.4.13}$$

- Dato un lato $ij \in E_1$ e un vertice $l \in V_2$, esiste al massimo un lato il cui vertice finale è l che può essere mappato con ij:

$$\sum_{kl \in E_2} y_{ij,kl} \leq x_{j,l} \qquad \forall l \in V_2, , \forall ij \in E_1 \tag{4.4.14}$$

Proposizione 4.4.4. *Sia Γ_1 l'insieme degli edit path (tra G_1 e G_2) sottintesi dall'insieme delle soluzioni ammissibili di $F1$ e sia Γ_2 l'insieme degli edit path ottenuti allo stesso modo sostituendo in $F1$ i vincoli (4.4.6) e (4.4.7) con i vincoli (4.4.13) e (4.4.14).*

Dimostrazione.

$\Gamma_2 \subseteq \Gamma_1$: Sia $ij \in E_1$ e sia $kl \in E_2$, supponiamo che il vincolo (4.4.13) sia soddisfatto.

$$x_{i,k} \geqslant \sum_{kl' \in E_2} y_{ij,kl'} \Rightarrow x_{i,k} \geqslant y_{ij,kl} + \sum_{kl' \in E_2, kl' \neq kl} y_{ij,kl'} \Rightarrow x_{i,k} \geqslant y_{ij,kl}$$

Quindi il punto (4.4.6) è soddisfatto $\forall ij \in E_1 \text{e} \forall kl \in E_2$. Allo stesso modo si può dedurre che (4.4.7) è soddisfatto usando il vincolo (4.4.14).

$\Gamma_1 \subseteq \Gamma_2$: Sia $ij \in E_1 \text{e} k \in V_2$. Se $\{l \in V_2 : kl \in E_2\} = \emptyset$, allora

$$\sum_{kl \in E_2} y_{ij,kl} = 0$$

e (4.4.13) è ancora soddisfatta. Quindi il vincolo (4.4.13) è soddisfatto per ogni $ij \in E_1$ e per ogni $k \in V_2$. Allo stesso modo si dimostra che (1.14) è soddisfatta usando (4.4.7) e (4.4.5). □

Il numero dei vincoli topologici (4.4.13) e (4.4.14) è ora $|E_1| \cdot |V_2|$.
Inoltre si può dimostrare che i vincoli (4.4.10) e (4.4.11) non sono necessari per la formulazione del GED, in quanto sono sottintesi dagli altri vincoli del BLP.

Proposizione 4.4.5. *Il vincolo (4.4.10) è dato dal (4.4.8) e (4.4.13).*

Dimostrazione.

Sia $ij \in E_1$. Dato (4.4.13). abbiamo:

$$\sum_{kl \in E_2} y_{ij,kl} \leq x_{i,k} \quad \forall k \in V_2 \Rightarrow \sum_{k \in V_2} \sum_{kl \in E_2} y_{ij,kl} \leqslant \sum_{k \in V_2} x_{i,k}$$

minimizzando a sinistra la disuguaglianza ed usando la (4.4.8) si ottiene:

$$\sum_{kl \in E_2} y_{ij,kl} \leqslant \sum_{k \in V_2} x_{i,k} \leqslant 1$$

Perciò la (4.4.10) è data da(4.4.8) e (4.4.13). Allo stesso modo, si può dimostrare che (4.4.11) è data da (4.4.9) e (4.4.14).

□

Da tutto ciò si deduce quindi che il problema del GED può essere risolto usando la (4.4.12) come funzione obiettivo e (4.4.8), (4.4.9), (4.4.13) e (4.4.14) come vincoli.

Si ha ora una nuova formulazione del $F1$ detta $F2$ del GED:

$$\min_{x,y}\Big(\sum_{i \in V_1} \sum_{k \in V_2} (c_{i,k} - c_{i,\epsilon} - c_{\epsilon,k}) \cdot x_{i,k} + \sum_{ij \in E_1} \sum_{kl \in E_2} (c_{ij,kl} - c_{ij,\epsilon} - c_{\epsilon,kl}) \cdot y_{ij,kl}\Big) + \gamma$$

$$\begin{aligned} \text{subject to} \quad & \sum_{k \in V_2} x_{i,k} \leqslant 1 \quad \forall i \in V_1 \\ & = \sum_{i \in V_1} x_{i,k} \leqslant 1 \quad \forall k \in V_2 \\ & = \sum_{kl \in E_2} y_{ij,kl} \leqslant x_{i,k} \quad \forall k \in V_2, \forall ij \in E_1 \\ & = \sum_{kl \in E_2} y_{ij,kl} \leqslant x_{j,l} \quad \forall l \in V_2, \forall ij \in E_1. \end{aligned} \tag{4.4.15}$$

Qui, γ non è una funzione di x e y, non ha nessun impatto nella minimizzazione del problema, tuttavia è importante utilizzarlo per ottenere il valore del GED.
La formulazione $F2$ ha $|V_1| \cdot |V_2| + |E_1| \cdot |E_2|$ variabili e $|V_1| + |V_2| + 2|V_2| \cdot |E_1|$ vincoli.

Osservazione 4.4.6. *Supponiamo che G_1 e G_2 siano grafi indiretti, cioè privi di orientazione. Le notazioni ij e ji si riferiscono allo stesso arco E_1 e kl e lk si riferiscono allo stesso arco E_2 . Si considera il nuovo vincolo:*

$$\sum_{kl \in E_2} y_{ij,kl} \leqslant (x_{i,k} + x_{j,k}) \quad \forall k \in V_2, \forall ij \in E_1 \tag{4.4.16}$$

Infatti, dato un arco $ij \in E_1$ e un vertice $k \in V_2$ esiste al massimo un arco che è incidente a k e che può essere mappato in ij. Tuttavia, $x_{i,k}$ e $x_{j,k}$ non possono essere allo stesso tempo uguale a 1, e quindi la somma $x_{i,k} + x_{j,k}$ è al più uguale ad 1.

Da tutto ciò è emerso che le funzioni obiettivo di $F1$ e $F2$ sono uguali, l'insieme dei vincoli di $F1$ e di $F2$ descrive il medesimo insieme di soluzioni ammissibili, perciò la loro soluzione ottimale è la stessa e quando viene raggiunta l'ottimalità, il valore delle loro funzioni obiettivo è il GED. Tuttavia è stato dimostrato anche che $F2$ utilizza meno variabili e meno vincoli di $F1$, ciò comporta una minore quantità di memoria da utilizzare, impiegando anche un minor tempo.

4.4.3 Il nostro metodo

L'implementazione è stata eseguita su MATLAB. La funzione GED ("*Graph Edit Distance*") permette di calcolare la distanza tra due grafi leggendoli da file in formato *.GXL*.
Gli argomenti utilizzati in input dalla funzione sono i seguenti:

- Percorso del file contenente il primo grafo;
- Percorso del file contenente il secondo grafo;
- Un vettore che contiene i pesi utilizzati dall'algoritmo, nell'ordine:
  - Costo per sostituire un nodo;
  - Costo per inserire/rimuovere un nodo;
  - Costo per sostituire un arco;
  - Costo per inserire/rimuovere un arco.

L'output della funzione è una struttura che ha i seguenti campi:

- ***Optimality***: ha valore 1 se il caso è ottimale (ovvero se il valore dell'Edit Distance è minore di una certa soglia, ad esempio < 100) e 0 altrimenti;
- ***Time***: il tempo impiegato dall'algoritmo espresso in secondi;
- ***Common-nodes***: contiene la lista dei nodi comuni ad entrambi i grafi;
- ***Distance***: ovvero l'Edit Distance vera e propria.

Il codice utilizzato per il GED è il seguente:

```
function out = ged(file1, file2, costs)
if isempty(strfind(getenv('PATH'), '/usr/local/bin'))
    setenv('PATH', [getenv('PATH'),':/usr/local/bin'])
end
exe_path = GetFullPath('GED_exe/F2_symbolic_distance.exe', 'lean');
exe_path = ['"', exe_path, '"'];
if isunix
    exe_path = ['wine ', exe_path];
end
full_path1 = ['"', GetFullPath(file1, 'lean'), '"'];
full_path2 = ['"', GetFullPath(file2, 'lean'), '"'];
cmd = strjoin({exe_path, num2str(costs), full_path1, full_path2}, ' ');
[~, cmdout] = system(cmd);
cmdout = strsplit(cmdout, ';');
cmdout{1} = regexp(cmdout{1}, '\d$', 'match');
out.optimality = str2double(cmdout{1});
out.time = str2double(strrep(cmdout{2}, ',', '.'));
out.common_nodes = strjoin(strsplit(cmdout{3}, ' '), ', ');
out.distance = str2double(cmdout{4});
end
```

Capitolo 5

Esperimenti e risultati

In questo capitolo verranno mostrati i risultati ottenuti dai numerosi esperimenti eseguiti utilizzando l'implementazione in MATLAB degli algoritmi precedentemente discussi.
Il campione utilizzato è costituito da 48 impronte digitali, prese dal database "*NIST Special Database 4 Gray Scale Images of Fingerprint Image Groups*(FIGS)".
Tali 48 impronte sono state acquisite da 24 individui e per ciascun individuo l'impronta è stata presa due volte per uno stesso dito, in momenti differenti.
Secondo il sistema di classificazione delle impronte digitali di Galton-Henry, è stato possibile suddividere ulteriormente il campione, perciò le 24 impronte digitali sono state suddivise nelle tre classi corrispondenti: Loop, Whorl ed Arch.

5.1 Primo Test

5.1.1 Confronto impronte su 10 minutiae

Nel primo confronto vengono mostrati i valori dell'Edit Distance, tra i grafi ottenuti in ciascuna impronta, per le coppie di impronte che appartengono allo stesso individuo.
Nelle seguenti tre tabelle, ciascuna per ogni classe (Arch,Whorl,Loop), sono stati presi 10 nodi, corrispondenti a 10 minutiae.
Innanzitutto, come abbiamo già visto nei capitoli precedenti, viene utilizzata la funzione "$select\_minutiae$", ovvero la funzione "Ginput", che permette di selezionare manualmente le minutiae.

Tale funzione presenta due parametri, che sono nell'ordine:

- percorso al file d'origine (che contiene l'immagine dell'impronta da analizzare)
- percorso al file di destinazione (che conterrà le posizioni delle minutiae)

La funzione salva automaticamente la lista delle minutiae selezionate nel file specificato come input (in formato CSV).
Successivamente viene utilizzata la funzione "$triangulate\_writeGXL$", la quale costruisce una triangolazione di Delaunay sui punti delle minutiae precedentemente selezionate. Tale funzione utilizza due argomenti:

- percorso al file d'origine (tale immagine viene utilizzata al fine di visualizzare la triangolazione su di essa)
- percorso al file di destinazione (file in cui ci sono le posizioni delle minutiae)

Questa funzione salva automaticamente un file con la scrittura in formato $.GXL$ del grafo risultante(GXL (*Graph eXchange Language*) è un formato file progettato per lo scambio di grafi). Per convertire nel formato GXL, viene utilizzata un'altra funzione, ovvero "$create\_GXL$" che crea header del file GXL; essa aggiunge tutte le minutiae come nodi di biforcazione, converte la matrice "triangulation" in una matrice (N _minutiae $\times$ 2), dove in ogni riga c'è un arco e tali archi non vengono ripetuti. Inoltre tale funzione salva ogni lato ottenuto dalla triangolazione come arco nel file GXL e infine scrive su file il contenuto del GXL. Tali file in formato $.GXL$ saranno confrontati utilizzando la funzione "GED" e quest'ultima funzione ci restituirà il valore effettivo dell'Edit Distance, ovviamente più questo valore sarà basso, maggiore sarà la probabilità che ciascuna coppia appartenga allo stesso individuo. Nella funzione GED, oltre ad utilizzare in input i due percorsi che contengono i due grafi, viene anche definito un vettore che contiene i pesi utilizzati dall'algoritmo, nell'ordine:

- Costo per sostituire un nodo;
- Costo per inserire/rimuovere un nodo;
- Costo per sostituire un arco;
- Costo per inserire/rimuovere un arco.

Nei seguenti esperimenti, abbiamo definito per ciascuna edit operation, il costo 1.

Tabella 1. Confronto impronte Arch: 10 minutiae

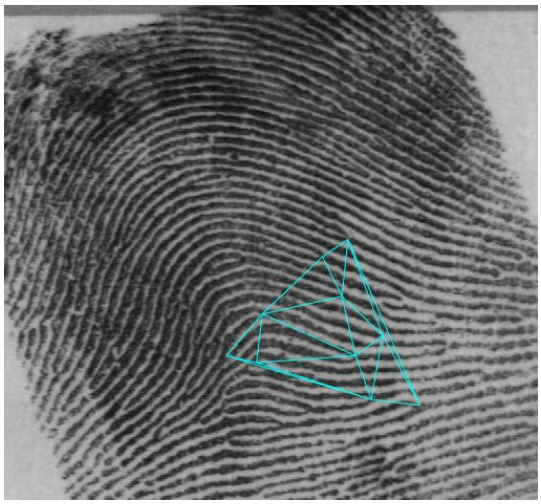
	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	3	7	4	1	7	3	4
f2	3	0	6	3	4	0	6	3
f3	7	6	0	7	6	6	0	7
f4	4	3	7	0	5	3	7	0
s1	1	4	6	5	0	4	6	5
s2	7	0	6	3	4	0	6	3
s3	3	6	0	7	6	6	0	7
s4	4	3	7	0	5	3	7	0

*Figura 5.1: **Tabella 1.** Confronto impronte Arch, 10 nodi*

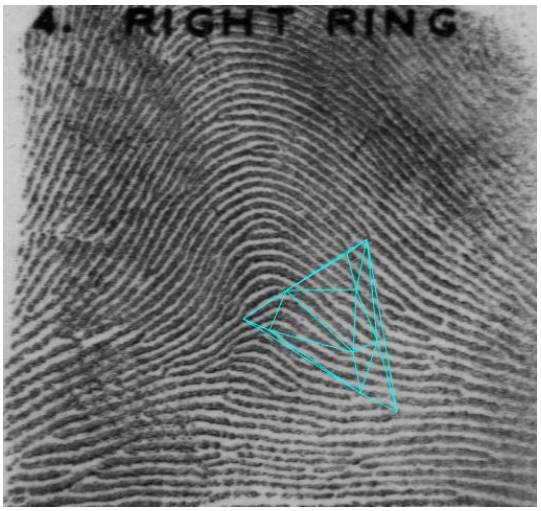
In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

- f1 = $f0383\_04A$ $s1 = s0383\_04A$
- f2 = $f0391\_01A$ $s2 = s0391\_01A$
- f3 = $f0418\_02A$ $s3 = s0418\_02A$
- f4 = $f0016\_08A$ $s4 = s0016\_08A$

Le coppie che corrispondono alla stessa persona sono appunto f1s1, f2s2, f3s3, f4s4. Come si può osservare, il valore dell'Edit Distance per ciascuna impronta con se stessa, ad esempio f1f1 viene zero, mentre i valori dell'Edit Distance di ciascuna coppia corrispondente allo stesso dito della stessa persona sono più bassi rispetto gli altri confronti. Infatti per trasformare un grafo in un altro, ad esempio f1s1 sono stati necessari poche sostituzioni di lati o nodi.
Nelle immagini seguenti viene mostrato il confronto tra coppie di impronte provenienti dalla stessa persona, ma catturate in momenti differenti.

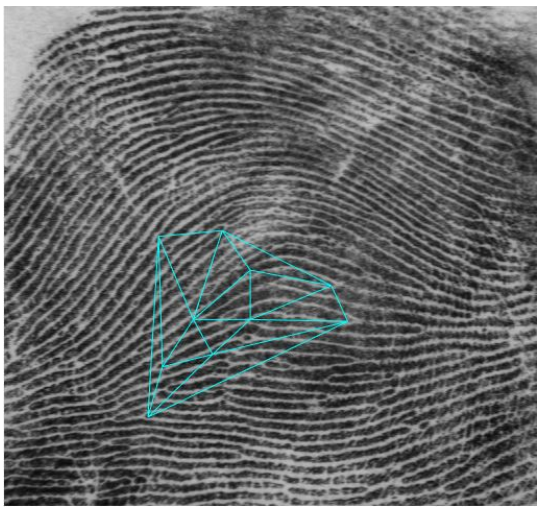


(a) $f0383\_04A$.



(b) $s0383\_04A$.

(c) $f0391\_01A$.



(d) $s0391\_01A$.

Tabella 2. Confronto impronte Whorl: 10 minutiae

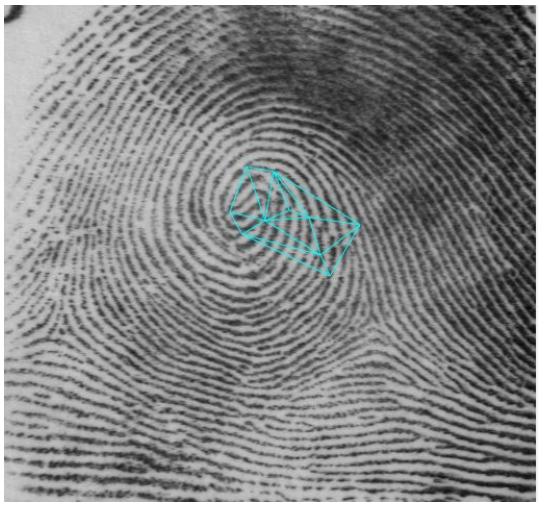
	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	4	7	4	0	4	5	4
f2	4	0	3	6	4	0	3	6
f3	7	3	0	7	7	3	1	7
f4	4	6	7	0	4	6	5	0
s1	0	4	7	4	0	4	5	4
s2	4	0	3	6	4	0	3	6
s3	5	3	1	5	5	3	0	5
s4	4	6	7	0	4	6	5	0

*Figura 5.2: **Tabella 2.** Confronto impronte Whorl, 10 nodi*

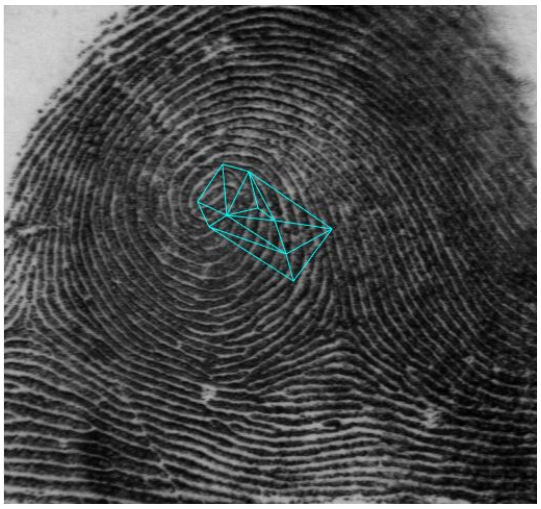
In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

- f1 = $f0014\_06W$ $s1 = s0014\_06W$
- f2 = $f0024\_06W$ $s2 = s0024\_06W$
- f3 = $f0252\_01W$ $s3 = s0252\_01W$
- f4 = $f0278\_03W$ $s4 = s0278\_03W$

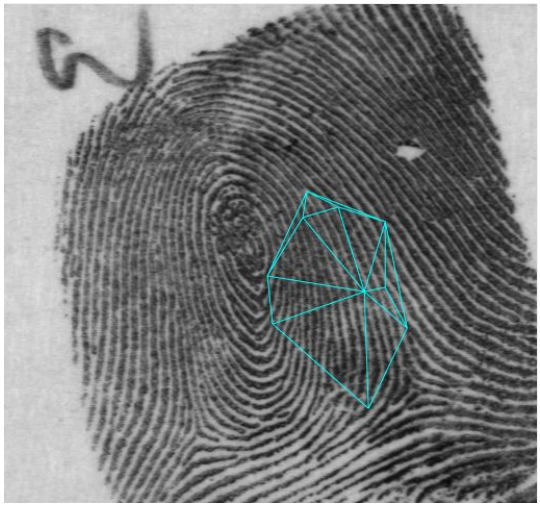
Le coppie corrispondenti alla stessa persona sono: f1s1, f2s2, f3s3, f4s4. Il valore dell'Edit Distance per ciascuna impronta con se stessa, ad esempio f1f1 viene zero, mentre i valori dell'Edit Distance di ciascuna coppia corrispondente allo stesso dito della stessa persona sono più bassi rispetto gli altri confronti.



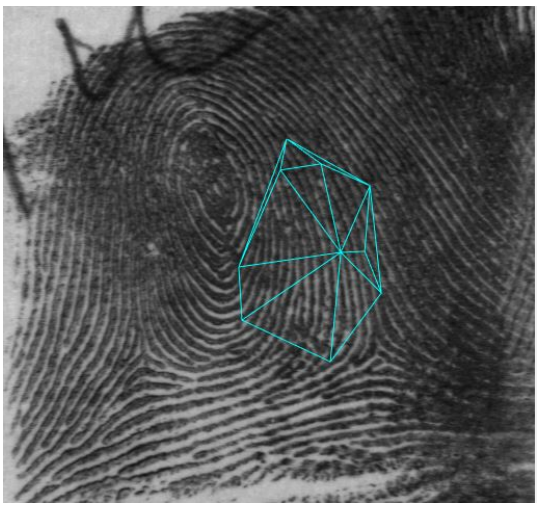
(a) $f0014\_06W$.



(b) $s0014\_06W$.



(c) $f0014\_06W$.



(d) $s0014\_06W$.

Tabella 3. Confronto impronte Loop: 10 minutiae

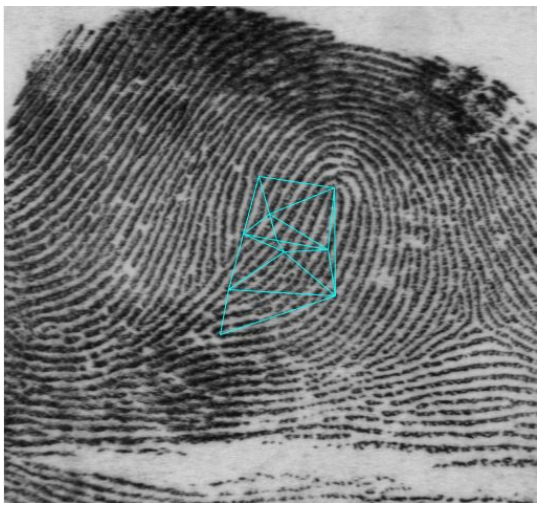
	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	3	3	3	1	5	3	3
f2	3	0	4	4	3	2	4	4
f3	3	4	0	3	3	4	0	3
f4	3	4	3	0	4	4	3	0
s1	1	3	3	4	0	5	6	3
s2	5	2	4	4	5	0	3	3
s3	3	4	0	3	6	3	0	3
s4	3	4	3	0	3	3	3	0

*Figura 5.3: **Tabella 3.** Confronto impronte Loop, 10 nodi*

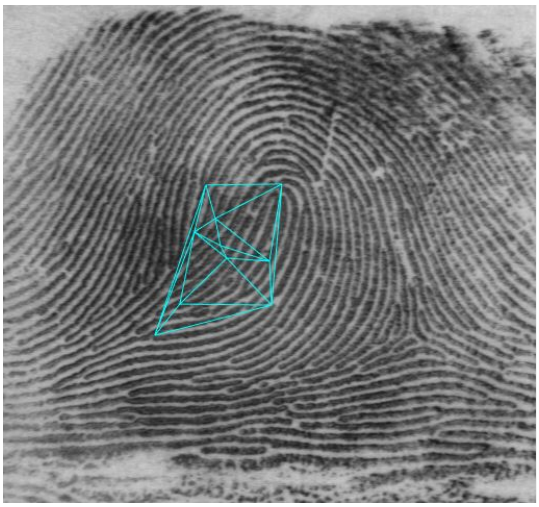
In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

- f1 = $f0506\_10L$ $s1 = s0506\_10L$
- f2 = $f0602\_07L$ $s2 = s0602\_07L$
- f3 = $f0750\_10L$ $s3 = s0750\_10L$
- f4 = $f0041\_09L$ $s4 = s0041\_09L$

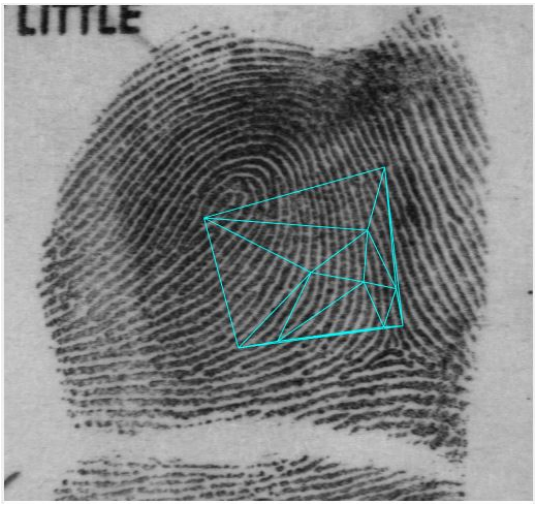
Le coppie corrispondenti alla stessa persona sono: f1s1, f2s2, f3s3, f4s4. Il valore dell'Edit Distance per ciascuna impronta con se stessa, ad esempio f1f1 viene zero, mentre i valori dell'Edit Distance di ciascuna coppia corrispondente allo stesso dito della stessa persona sono più bassi rispetto gli altri confronti.



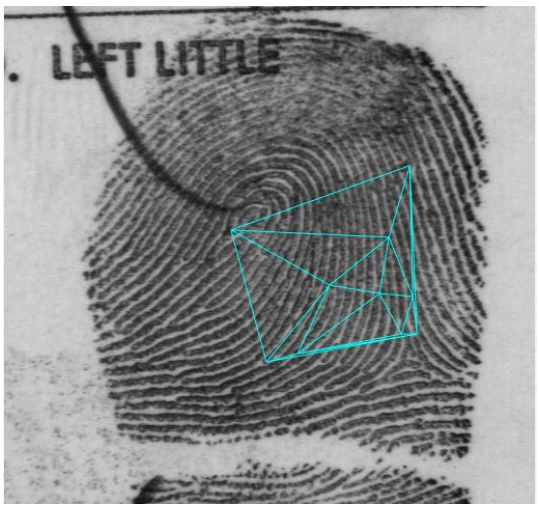
(a) $f0602\_07L$.



(b) $s0602\_07L$.



(c) $f0750\_10L$.



(d) $s0750\_10L$.

5.2 Secondo Test

5.2.1 Confronto impronte su 20 minutiae

Allo stesso modo, ho confrontato i grafi ottenuti dall'estrazione di 20 punti minutiae, utilizzando altre impronte. Nelle seguenti tre tabelle sono riportati i valori dell'Edit Distance, suddivise per classi (Arch, Whorl e Loop).

Tabella 1. Confronto impronte Arch: 20 minutiae

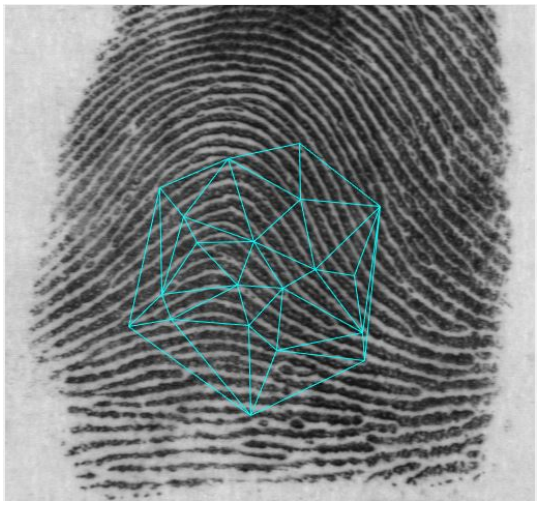
	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	32	46	32	16	30	28	34
f2	32	0	26	40	32	0	42	48
f3	46	26	0	26	34	46	0	30
f4	32	40	26	0	28	32	36	0
s1	16	32	34	28	0	40	26	46
s2	30	0	46	32	40	0	36	32
s3	28	42	0	36	26	36	0	34
s4	34	48	30	0	46	32	34	0

*Figura 5.4: **Tabella 1.** Confronto impronte Arch, 20 nodi*

In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

- f1 = $f0005\_03A$ $s1 = s0005\_03A$
- f2 = $f0031\_02A$ $s2 = s0031\_02A$
- f3 = $f0517\_10A$ $s3 = s0517\_10A$
- f4 = $f0315\_03A$ $s4 = s0315\_03A$

Le coppie corrispondenti alla stessa persona sono: f1s1, f2s2, f3s3, f4s4. Il valore dell'Edit Distance per ciascuna impronta con se stessa, ad esempio f1f1 viene zero, mentre i valori dell'Edit Distance di ciascuna coppia corrispondente allo stesso dito della stessa persona sono più bassi rispetto gli altri confronti.



(a) $f0031\_02A$.

(b) $s0031\_02A$.

(c) $f0315\_03A$.

(d) $s0315\_03A$.

Tabella 2. Confronto impronte Whorl: 20 minutiae

	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	29	38	31	0	32	42	34
f2	29	0	32	42	37	17	35	43
f3	32	32	0	31	36	34	14	30
f4	31	42	31	0	33	47	27	9
s1	0	37	36	33	0	32	42	28
s2	32	17	34	47	32	0	32	32
s3	42	35	14	27	42	32	0	28
s4	34	43	30	9	28	32	28	0

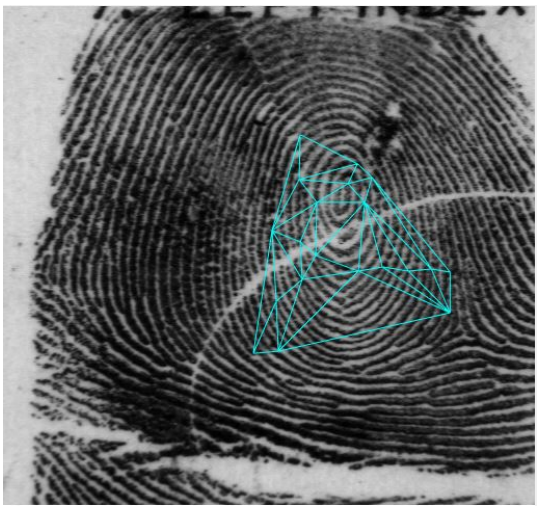
*Figura 5.5: **Tabella 2.** Confronto impronte Whorl, 20 nodi*

In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

- f1 = $f0257\_07W$ $s1 = s0257\_07W$
- f2 = $f0276\_09W$ $s2 = s0276\_09W$
- f3 = $f0312\_08W$ $s3 = s0312\_08W$
- f4 = $f0477\_04W$ $s4 = s0477\_04W$

Le coppie corrispondenti alla stessa persona sono: f1s1, f2s2, f3s3, f4s4. Il valore dell'Edit Distance per ciascuna impronta con se stessa, ad esempio f1f1 viene zero, mentre i valori dell'Edit Distance di ciascuna coppia corrispondente allo stesso dito della stessa persona sono più bassi rispetto gli altri confronti.

(a) $f0257\_07W$.



(b) $s0257\_07W$.

(c) $f0477\_04W$.

(d) $s0477\_04W$.

Tabella 3. Confronto impronte Loop: 20 minutiae

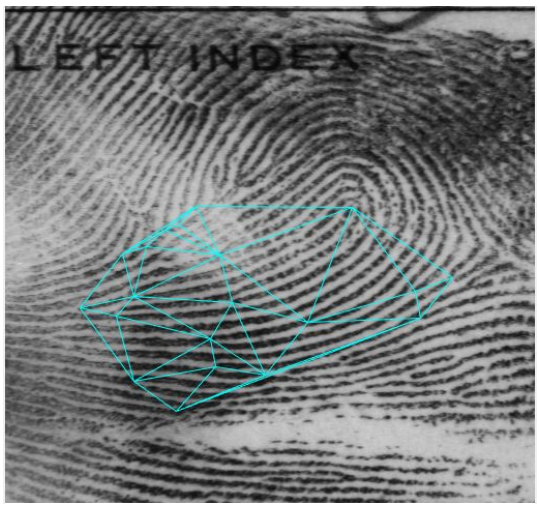
	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	35	54	47	24	4	54	45
f2	35	0	33	34	43	0	33	26
f3	54	33	0	35	34	55	0	61
f4	47	34	35	0	63	42	37	18
s1	24	43	34	63	0	57	36	47
s2	45	0	55	42	57	0	43	60
s3	54	33	0	37	36	43	0	39
s4	45	26	61	18	47	60	39	0

*Figura 5.6: **Tabella 3.** Confronto impronte Loop, 20 nodi*

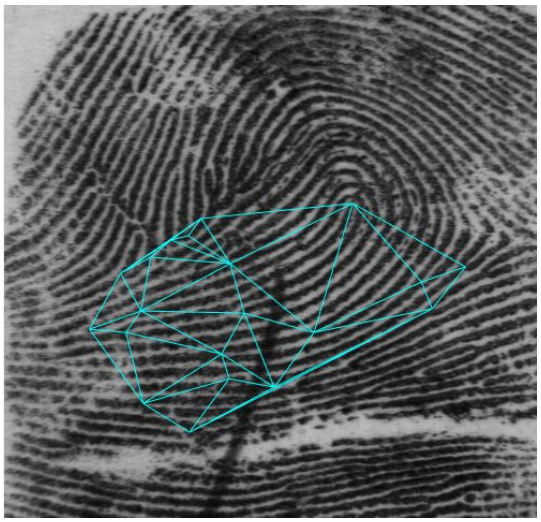
In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

- f1 = $f0157\_03L$ $s1 = s0157\_03L$
- f2 = $f0458\_07L$ $s2 = s0458\_07L$
- f3 = $f0361\_06L$ $s3 = s0361\_06L$
- f4 = $f0298\_06L$ $s4 = s0298\_06L$

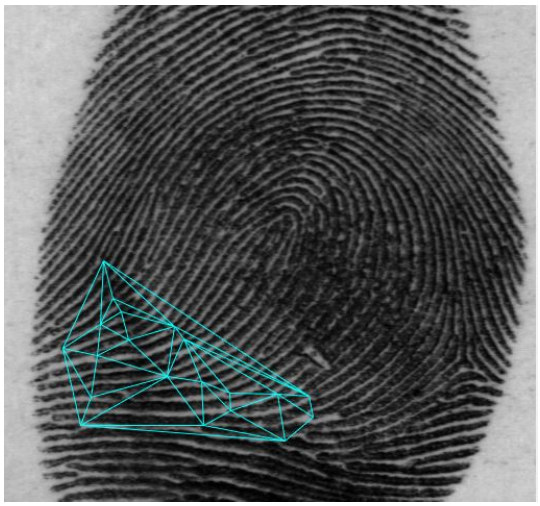
Le coppie corrispondenti alla stessa persona sono: f1s1, f2s2, f3s3, f4s4. Il valore dell'Edit Distance per ciascuna impronta con se stessa, ad esempio f1f1 viene zero, mentre i valori dell'Edit Distance di ciascuna coppia corrispondente allo stesso dito della stessa persona sono più bassi rispetto gli altri confronti.



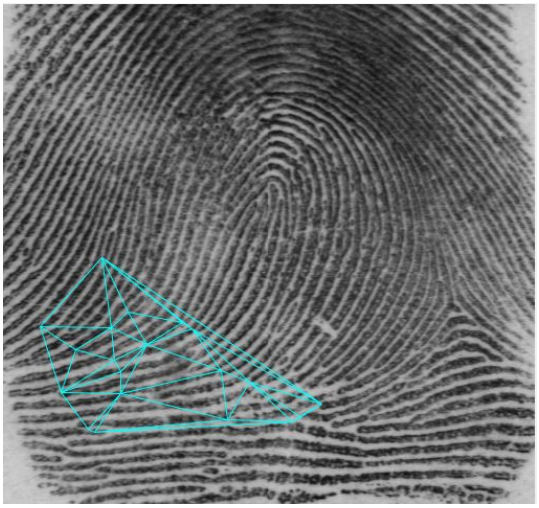
(a) $f0458\_07L$.



(b) $s0458\_07L$.



(c) $f0298\_06L$.



(d) $s0298\_06L$.

5.3 Terzo Test

5.3.1 Confronto impronte con perturbazioni

In questo Test viene effettuato un confronto tra il grafo di un'impronta digitale di un individuo con il grafo della stessa impronta digitale, il cui file è stato precedentemente perturbato secondo una distribuzione Gaussiana.
Viene utilizzata la funzione denominata "AddGaussianPerturbation", implementata su MATLAB. Tale funzione carica il file CSV, seleziona una percentuale di righe del file CSV (ovvero un numero compreso tra 0 e 100). Ognuna delle righe selezionate è costituita da due colonne, denominate x ed y, corrispondenti alle coordinate (espresse in pixel) delle minutiae.
La funzione agisce su ogni riga selezionata, aggiungendo:

- alla x una perturbazione gaussiana di centro "mu_x" e deviazione standard "$sigma\_x$";
- alla y una perturbazione gaussiana di centro "mu_y" e deviazione standard "$sigma\_y$"

Il valore della deviazione standard verrà fissato all'inizio di ogni perturbazione. Si osserva inoltre che la perturbazione non ammette ripetizioni, eliminando quindi i doppioni, perciò il numero di righe perturbate sarà casuale. Il codice utilizzato è il seguente:

```
function addGaussianPerturbation(inFile, amount, mu_x, sigma_x, mu_y, sigma_y, outFile)
M = csvread(inFile);
m = round(amount / 100 * size(M,1));
row_idx = round(rand(m,1) * (size(M,1)-1) + 1);
row_idx = unique(row_idx);
m = numel(row_idx);
real_amount = m / size(M,1) * 100;
if real_amount ~= amount
    disp(['Warning: using only ', num2str(real_amount), ...
        ' % of rows, due to repetitions, not ',...
        num2str(amount), ' % as requested']);
end
L = M(row_idx, :);
P = randn(m, 2);
P(:,1) = sigma_x * P(:,1) + mu_x;
P(:,2) = sigma_y * P(:,2) + mu_y;
M(row_idx, :) = L + P;
csvwrite(outFile, M);
```

Tabella 1. Confronto impronte perturbate Arch: 10 minutiae

	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	3	7	4	1	3	6	6
f2	3	0	6	3	3	2	5	5
f3	7	6	0	7	6	7	1	7
f4	4	3	7	0	5	6	7	2
s1	1	3	6	5	0	5	7	5
s2	3	2	7	6	5	0	6	4
s3	6	5	1	7	7	6	0	6
s4	6	5	7	2	5	4	6	0

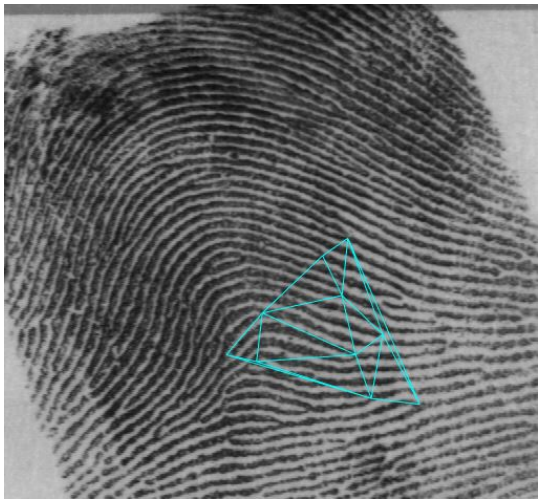
*Figura 5.7: **Tabella 1.** Confronto impronte perturbate Arch, 10 nodi*

In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

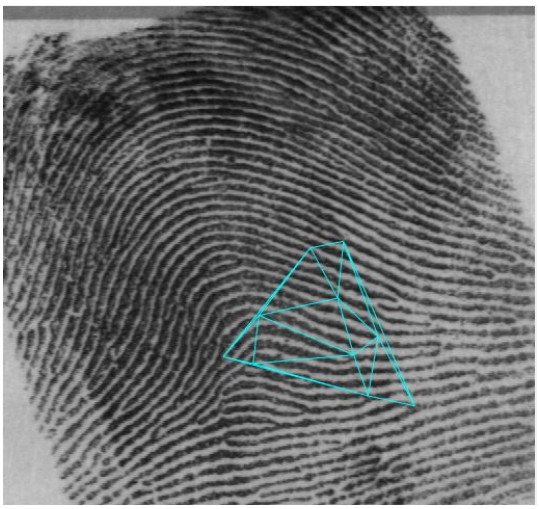
- f1 = $f0383\_04A$ $s1 = f0383\_04AP5$
- f2 = $f0391\_01A$ $s2 = f0391\_01AP5$
- f3 = $f0418\_02A$ $s3 = f0418\_02AP5$
- f4 = $f0016\_08A$ $s4 = f0016\_08AP5$

Le impronte s1,s2,s3,s4 sono le impronte perturbate. La perturbazione è stata eseguita sul 75% delle righe dei file CSV, con deviazione standard ($sigma\_x$ e $sigma\_y$) uguale a 5.

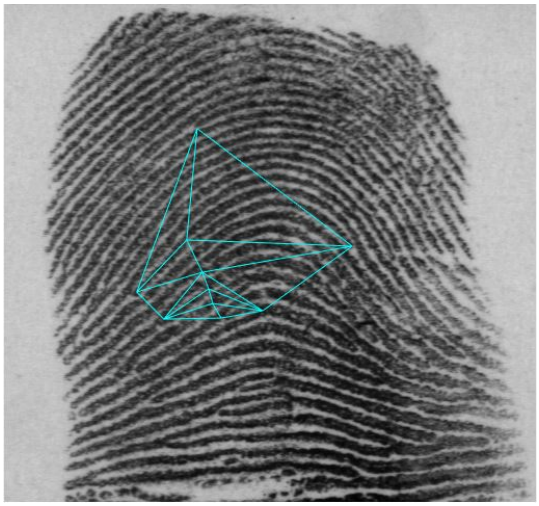
Si osserva dalla ***Tabella 1.*** che i valori dell'Edit Distance tra la stessa impronta perturbata e non perturbata, come ad esempio $f1s1$, sono molto più bassi (ottimali), rispetto tutti gli altri confronti.



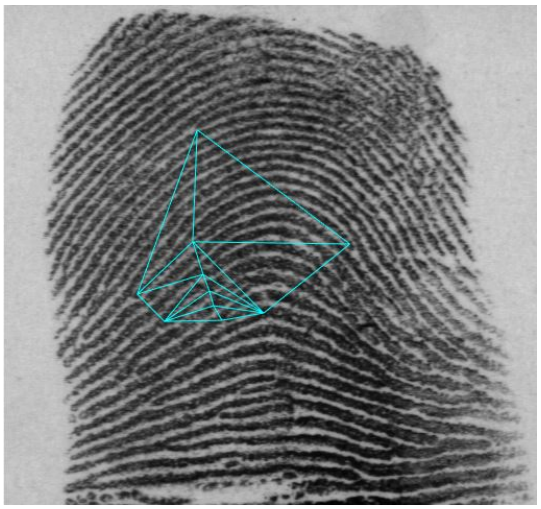
(a) $f0383\_04A$.



(b) $f0383\_04AP5$.



(c) $f0016\_08A$.



(d) $f0016\_08AP5$.

Tabella 2. Confronto impronte perturbate Whorl: 10 minutiae

	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	4	7	4	2	3	7	4
f2	4	0	3	6	4	2	3	6
f3	7	3	0	7	5	4	0	7
f4	4	6	7	0	6	3	7	0
s1	2	4	5	6	0	3	5	6
s2	3	2	4	3	3	0	4	3
s3	7	3	0	7	5	4	0	7
s4	4	6	7	0	6	3	7	0

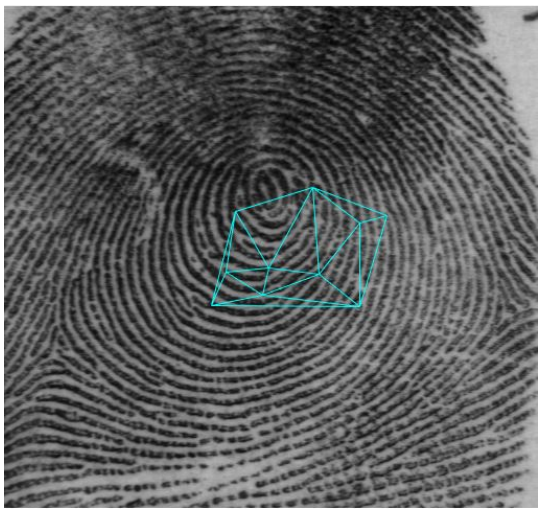
*Figura 5.8: **Tabella 2.** Confronto impronte perturbate Whorl, 10 nodi*

In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

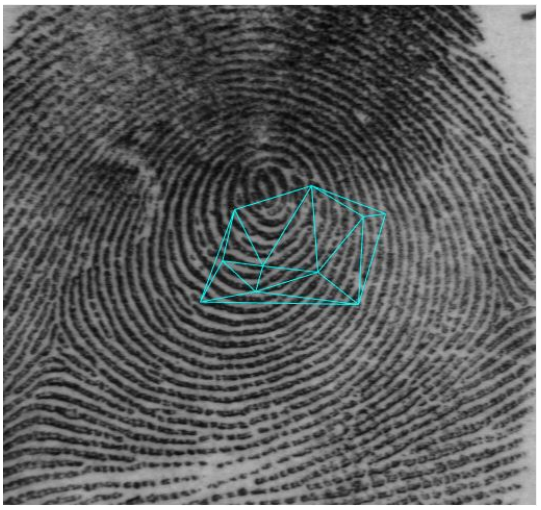
- f1 = $f0014\_06W$ $s1 = f0014\_06WP5$
- f2 = $f0024\_06W$ $s2 = f0024\_06WP5$
- f3 = $f0252\_01W$ $s3 = f0252\_01WP5$
- f4 = $f0278\_03W$ $s4 = f0278\_03WP5$

Le impronte s1,s2,s3,s4 sono le impronte perturbate. La perturbazione è stata eseguita sul 75% delle righe dei file CSV, con deviazione standard ($sigma\_x$ e $sigma\_y$) uguale a 5.

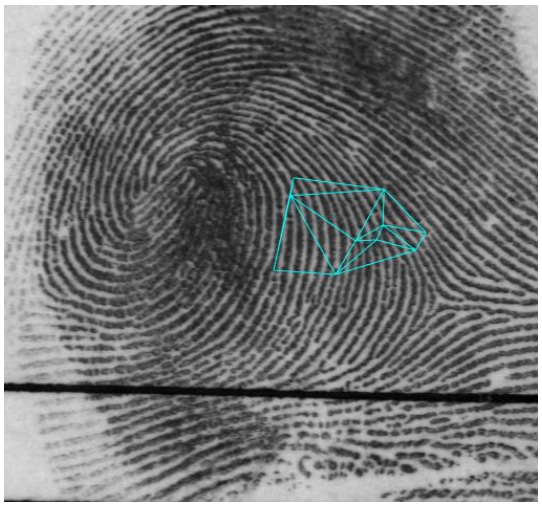
Si osserva dalla ***Tabella 2.*** che i valori dell'Edit Distance tra la stessa impronta perturbata e non perturbata, come ad esempio $f1s1$, sono molto più bassi (ottimali), rispetto tutti gli altri confronti.



(a) $f0252\_01W$.



(b) $f0252\_01WP5$.



(c) $f0024\_06W$.

(d) $f0024\_06WP5$.

Tabella 3. Confronto impronte perturbate Loop: 10 minutiae

	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	3	3	3	1	3	2	3
f2	3	0	4	4	3	2	3	4
f3	3	4	0	3	2	4	1	4
f4	3	4	3	0	4	4	3	0
s1	1	3	2	4	0	5	4	2
s2	3	2	4	4	5	0	5	2
s3	2	3	1	3	4	5	0	4
s4	3	4	4	0	2	2	4	0

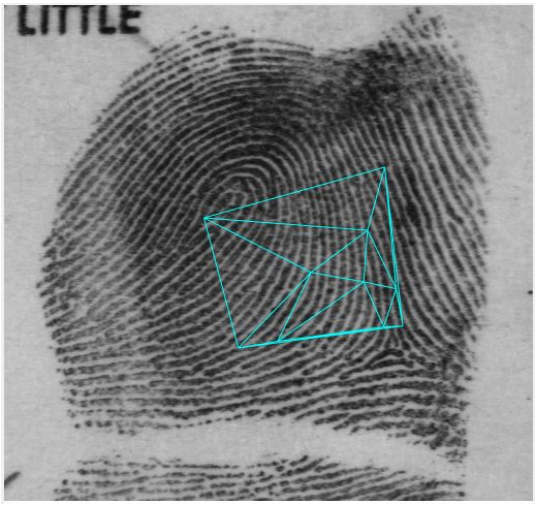
*Figura 5.9: **Tabella 3.** Confronto impronte perturbate Loop, 10 nodi*

In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

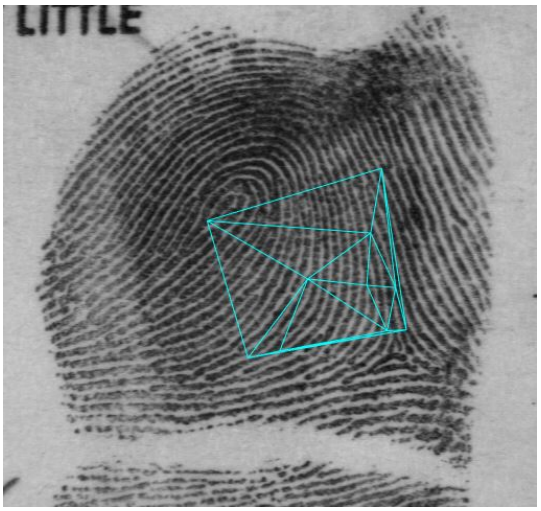
- f1 = $f0506\_10L$ $s1 = f0506\_10L$
- f2 = $f0602\_07L$ $s2 = f0602\_07L$
- f3 = $f0750\_10L$ $s3 = f0750\_10L$
- f4 = $f0041\_09L$ $s4 = f0041\_09L$

Le impronte s1,s2,s3,s4 sono le impronte perturbate. La perturbazione è stata eseguita sul 75% delle righe dei file CSV, con deviazione standard ($sigma\_x$ *e* $sigma\_y$) uguale a 5.

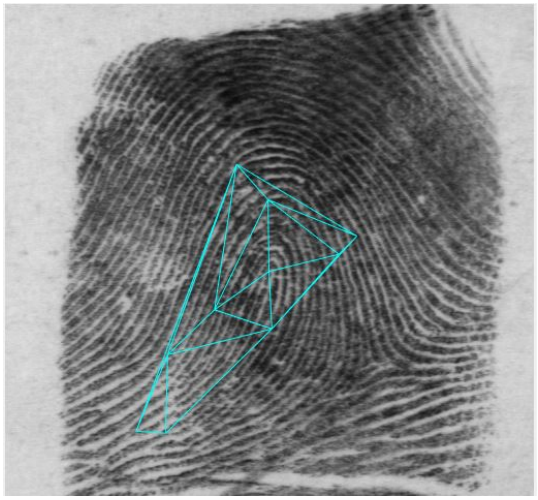
Si osserva dalla ***Tabella 3.*** che i valori dell'Edit Distance tra la stessa impronta perturbata e non perturbata, come ad esempio $f1s1$, sono molto più bassi (ottimali), rispetto tutti gli altri confronti.



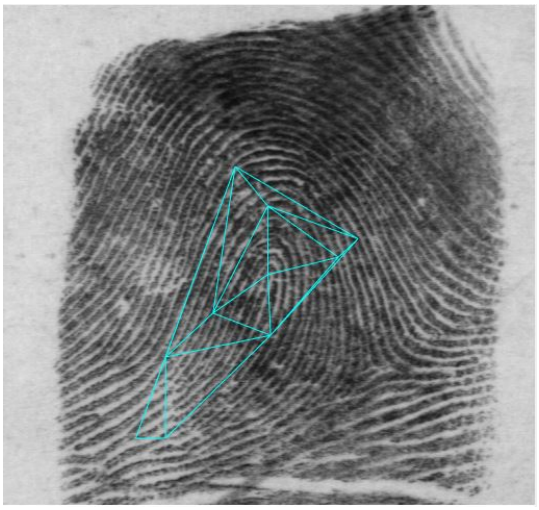
(a) $f0750\_10L$.



(b) $f0750\_10LP5$.



(c) $f0506\_10L$.



(d) $f0506\_10LP5$.

Tabella 4. Confronto impronte perturbate Arch: 20 minutiae

	f1	f2	f3	f4	s1	s2	s3	s4
f1	0	3	3	3	1	3	2	3
f2	3	0	4	4	3	2	3	4
f3	3	4	0	3	2	4	1	4
f4	3	4	3	0	4	4	3	0
s1	1	3	2	4	0	5	4	2
s2	3	2	4	4	5	0	5	2
s3	2	3	1	3	4	5	0	4
s4	3	4	4	0	2	2	4	0

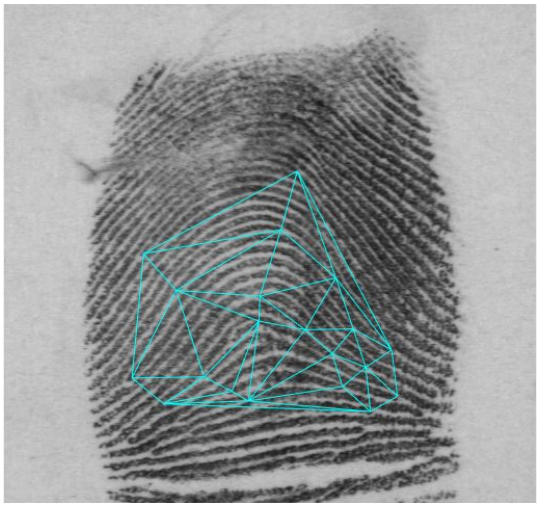
*Figura 5.10: **Tabella 4.** Confronto impronte perturbate Arch, 20 nodi*

In questa tabella osserviamo che le impronte sono suddivise in f1,f2,f3,f4 e s1,s2,s3,s4. Dove:

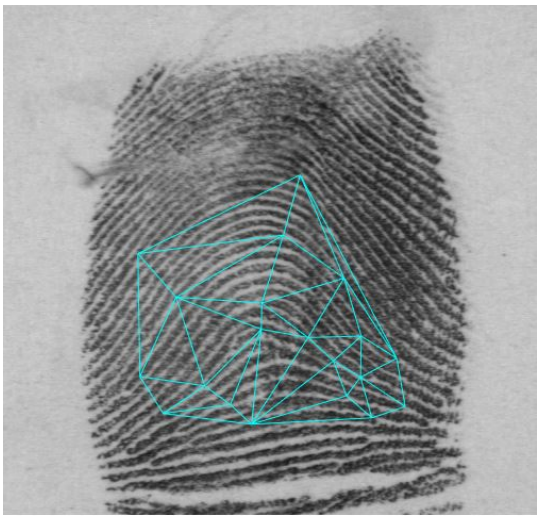
- f1 = $f0005\_03A$ $s1 = f0005\_03AP5$
- f2 = $f0031\_02A$ $s2 = f0031\_02AP5$
- f3 = $f0517\_10A$ $s3 = f0517\_10AP5$
- f4 = $f0315\_03A$ $s4 = f0315\_03AP5$

Le impronte s1,s2,s3,s4 sono le impronte perturbate. La perturbazione è stata eseguita sul 75% delle righe dei file CSV, con deviazione standard ($sigma\_x$ *e* $sigma\_y$) uguale a 1.

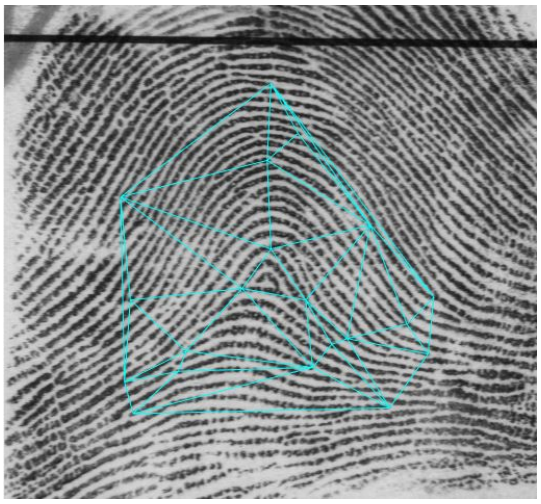
Si osserva dalla ***Tabella 4.*** che i valori dell'Edit Distance tra la stessa impronta perturbata e non perturbata, come ad esempio $f1s1$, sono molto più bassi (ottimali), rispetto tutti gli altri confronti.



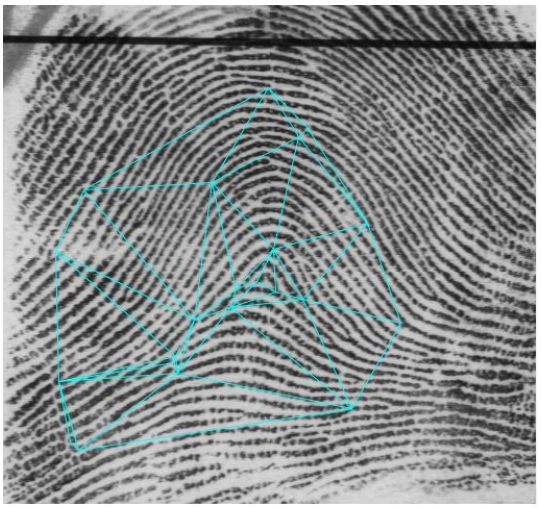
(a) $f0517\_10A$.



(b) $f0517\_10AP5$.



(c) $f0005\_03A$.



(d) $f0005\_03AP5$.

Conclusioni

Con questo elaborato si è voluto esporre un'innovativa proposta in tema di confronto dattiloscopico. Come primo step della trattazione, viene svolto un focus sull'importanza storica ed anatomica della comparazione, per poi concentrare l'attenzione sulle applicazioni utilizzate nella fase di "*matching*" delle impronte digitali. A tale riguardo, occorre precisare che ci si è serviti di due tecniche di fondamentale rilevanza: la *triangolazione di Delaunay* e l'*Edit Distance*. Con l'ausilio della prima, utilizzando come vertici dei triangoli i punti minutiae estratti dall'impronta, vengono elaborati dei grafi che si mettono a confronto tramite l'Edit Distance. Più il valore dell'Edit Distance risulterà essere basso, maggiore sarà la probabilità che due impronte appartengano allo stesso individuo. L'Edit Distance calcola, difatti, il percorso più breve per uniformare un grafo all'altro attraverso delle operazioni "pesate": le cosiddette "*operazioni di edit*".
I confronti sono stati eseguiti sia tra coppie di impronte appartenenti allo stesso individuo, sia tra coppie di impronte appartenenti a soggetti differenti. Osservando le tabelle e le relative immagini riportate nel **Capitolo 5**, si può affermare di aver ottenuto significativi risultati pur avendo impiegato un numero esiguo di minutiae.
Nel presente lavoro, sono state utilizzate 48 impronte digitali, appartenenti a 24 persone. Dalle impronte, provenienti dal database "*NIST Special Database 4 Gray Scale Images of Fingerprint Image Groups (FIGS)*", sono state estratte dapprima 10 minutiae ed in una fase successiva 20 minutiae. Su ciascuna sono stati poi costruiti i grafi e sono stati analizzati i valori finali dell'Edit Distance. Dal matching è emerso che coppie corrispondenti alla medesima persona presentavano un valore di Edit Distance più basso rispetto agli altri confronti e tra stesse impronte digitali, tale valore, risultava essere zero. Inoltre è necessario specificare che ci si è limitati ad un numero non elevato di minutiae, in quanto, provando questa metodologia di matching su un totale di 30 minutiae, i valori finali risultavano essere alquanto instabili.

In futuro, in un'ottica di perfezionamento della tecnica, si potrebbe procedere anzitutto ad un miglioramento dell'algoritmo finale per il confronto tra i grafi, così da ottenere valori di Edit Distance più accurati e, in aggiunta, all'estrazione di un numero di minutiae più elevato. Inoltre, per ricavare esiti più rigorosi, si potrebbe affinare l'algoritmo per l'estrazione di punti minutiae, così da non rendere più necessario il ricorso alla funzione "Ginput" (funzione che permette di selezionare manualmente le minutiae su un'impronta digitale), cercando in tal modo di evitare possibili errori causati dall'occhio umano.

Bibliografia

[1] Davide Maltoni, Dario Maio, Anil K Jain, and Salil Prabhakar. Handbook of fingerprint recognition. Springer Science Business Media, (2009).

[2] Anil K. Jain, Arun A. Ross, Karthik Nandakumar Introduction to Biometrics. Springer Science Business Media, LLC (2011).

[3] Fred Hobson, Barbara Ladd. The Oxford Handbook of the Literature of the U.S. South (Oxford Handbooks) 1st Edition (2016).

[4] Michael H. Ross. Histology: A Text and Atlas, Lww; 7th International edizione (1 gennaio 2015).

[5] Marina L. Gavrilova C.J. Kenneth Tan Mir Abolfazl Mostafavi. Transactions on Computational Science XIV Special Issue on Voronoi Diagrams and Delaunay Triangulation, Springer-Verlag Berlin Heidelberg (2011).

[6] George, Paul-Louis Borouchaki. Delaunay Triangulation and Meshing : Application to Finite Elements. Paris: Hermes, (1998).

[7] Franco P. Preparata, Michael Ian Shamos. Computational Geometry An Introduction. Springer-Verlag New York Inc.(1985).

[8] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars. Computational Geometry, Springer-Verlag Berlin Heidelberg (2008).

[9] Dario Maio and Davide Maltoni. Direct gray-scale minutiae detection in fingerprints. Pattern Analysis and Machine Intelligence, IEEE Transactions (1997).

[10] Nalini K Ratha, Shaoyun Chen, and Anil K Jain. Adaptive flow orientation based feature extraction in fingerprint images (1995).

[11] Michael Kass and Andrew Witkin. Analyzing oriented patterns. Computer vision, graphics, and image processing (1987)

[12] B.G. Sherlock and D.M. Monro. A model for interpreting fingerprint topology, (5 January 1993).

[13] Bebis, G., Deaconu, T and Georiopoulous, M. Fingerprint identification using Delaunay triangulation, ICIIS99, Maryland, (1999).

[14] Pedro Ribeiro Mendes Junior, Antonio Carlos de Nazare Junior, David Menotti. A Complete System for Fingerprint Authentication using Delaunay Triangulation.

[15] Nalini K Ratha, Shaoyun Chen, and Anil K Jain. Adaptive flow orientation based feature extraction in fingerprint images (1995).

[16] Hong, Yifei Wan, and Anil Jain. Fingerprint image enhancement: algorithm and performance evaluation (1998).

[17] Zhou, Gu and Zhang. Singular Points Analysis in Fingerprints Based on Topological Structure and Orientation Field (2007).

[18] Ratha, Chen and Jain . Adaptive flow orientation-based feature extraction in fingerprint images (1995).

[19] Lin Hong, Yifei Wan, and Anil Jain. Fingerprint image enhancement: algorithm and performance evaluation (1998).

[20] Karu and Jain. Fingerprint classification (1996).

[21] Gonzalo Navarro. A Guided Tour to Approximate String Matching, University of Chile (2001).

[22] Daniel Jurafsky, James H. Martin. Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (2018).

[23] Philip Bille. Asurvey on tree edit distance and related problems (2004).

[24] Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux, Sébastien Adam. New binary linear programming formulation to compute the graph edit distance. Pattern Recognition, Elsevier,(2017).

[25] Kaspar Riesen, Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching (2008).

[26] Peter Hart, Nils J. Nilsson and Bertram Raphaela. Formal Basis for the Heuristic Determination of Minimum Cost Paths (1968).

[27] D. Justice, A. Hero. A Binary Linear Programming Formulation of the Graph Edit Distance (2006).