Fingerprint Recognition

Progetto di Sistemi Multimediali

Mennella Antonio mat. N46003696

Mennillo Valerio mat. N46003768

A.A. 2019/2020

Indice

Introduzione

L'impronta digitale è una traccia lasciata dai polpastrelli delle dita su una superficie liscia. Le proprietà di un'impronta digitale sono 2:

Immutabilità

Le impronte si formano definitivamente nel feto all'ottavo mese di gravidanza e non cambiano per tutta la vita. In caso di graffi o tagli, la pelle dei polpastrelli ricresce con le stesse caratteristiche. Modificarle chirurgicamente è quasi impossibile: un medico riconoscerebbe a occhio nudo che la cresta originaria è stata sostituita da una cicatrice.

Individualità

L'impronta digitale cambia per ogni persona e ha dunque la caratteristica di unicità. Anche i gemelli omozigoti hanno la caratteristica di avere impronte distinte. In realtà ad oggi l'uguaglianza di due impronte appartenenti a due persone diverse non è stata mai provata.

Queste proprietà fanno sì che le impronte siano strumento per identificare persone.

Nell'ambito giudiziario quando una persona non è riconoscibile in alcun modo dalla legge, le vengono prelevate oltre a informazioni generiche anche le impronte digitali in modo che egli sia facilmente identificabile in seguito.

La rilevazione delle impronte digitali è prevista per ciascun cittadino di età maggiore o uguale a 12 anni. Le impronte digitali verranno scritte in sicurezza all'interno della propria CIE (Carta d'Identità Elettronica) e non depositate in nessun altro luogo.

A livello pratico, l'operatore comunale utilizza un dispositivo di rilevazione (sensore) su cui il cittadino è invitato a poggiare le proprie dita, al fine di acquisire le impronte.

Le impronte digitali devono essere acquisite a partire dal dito indice della mano destra e a seguire, dal dito indice della mano sinistra.

Il rilevamento delle impronte digitali rientra nelle operazioni urgenti sui luoghi e tra gli accertamenti tecnici non ripetibili. La coincidenza di molti punti dell'impronta costituisce prova certa sull'autore del reato anche in assenza di ulteriori elementi di prova.

Tuttavia anche se il processo di riconoscimento delle impronte digitali prevede l'analisi delle caratteristiche principali quali terminazioni e biforcazioni, in questo progetto abbiamo proposto un metodo alternativo di riconoscimento, avente vantaggi e svantaggi.

Nonostante sia campo di esperti la comparazione e identificazione di impronte digitali, la nostra curiosità ci ha spinto a provare un metodo alternativo che si è rivelato piuttosto soddisfacente e ci ha fatto scoprire un nuovo contesto applicativo dei Sistemi Multimediali.

Buona lettura

Tecnologie Utilizzate

L'analisi delle impronte ci ha inevitabilmente fatto discutere sull'ambiente e tecnologie da utilizzare. Le alternative erano tante ma ci siamo concentrati su **MATLAB** (**MAT**rix **LAB**oratory) data la sua predisposizione all'analisi, alla rappresentazione, alla gestione e alla manipolazione di immagini (matrici) e le loro caratteristiche.

Nello specifico, abbiamo avuto a disposizione un toolbox integrato in MATLAB, **Image Processing Toolbox**, che ci ha facilitato lo svolgimento di queste operazioni.

Alcune funzioni richiamate da **Image Processing Toolbox 10.3**:

Image read
```
A = imread(FILENAME)
```

Image show
```
imshow(I)
```

Morphological operations on binary image
```
BW2 = bwmorph(BW1,OPERATION)
```

Image Closing operation
```
IM2 = imclose(IMG,SE)
```

ecc.

Oltre ai toolbox forniti automaticamente da MATLAB, abbiamo avuto bisogno di strumenti per estrarre le feature caratteristiche delle immagini contenenti le impronte. **Computer Vision System Toolbox** è stata la nostra scelta data la sua versatilità e flessibilità.

Computer Vision System Toolbox 8.2 ci ha fornito:

Feature detection
```
POINTS = detectSURFFeatures(IMG)
```

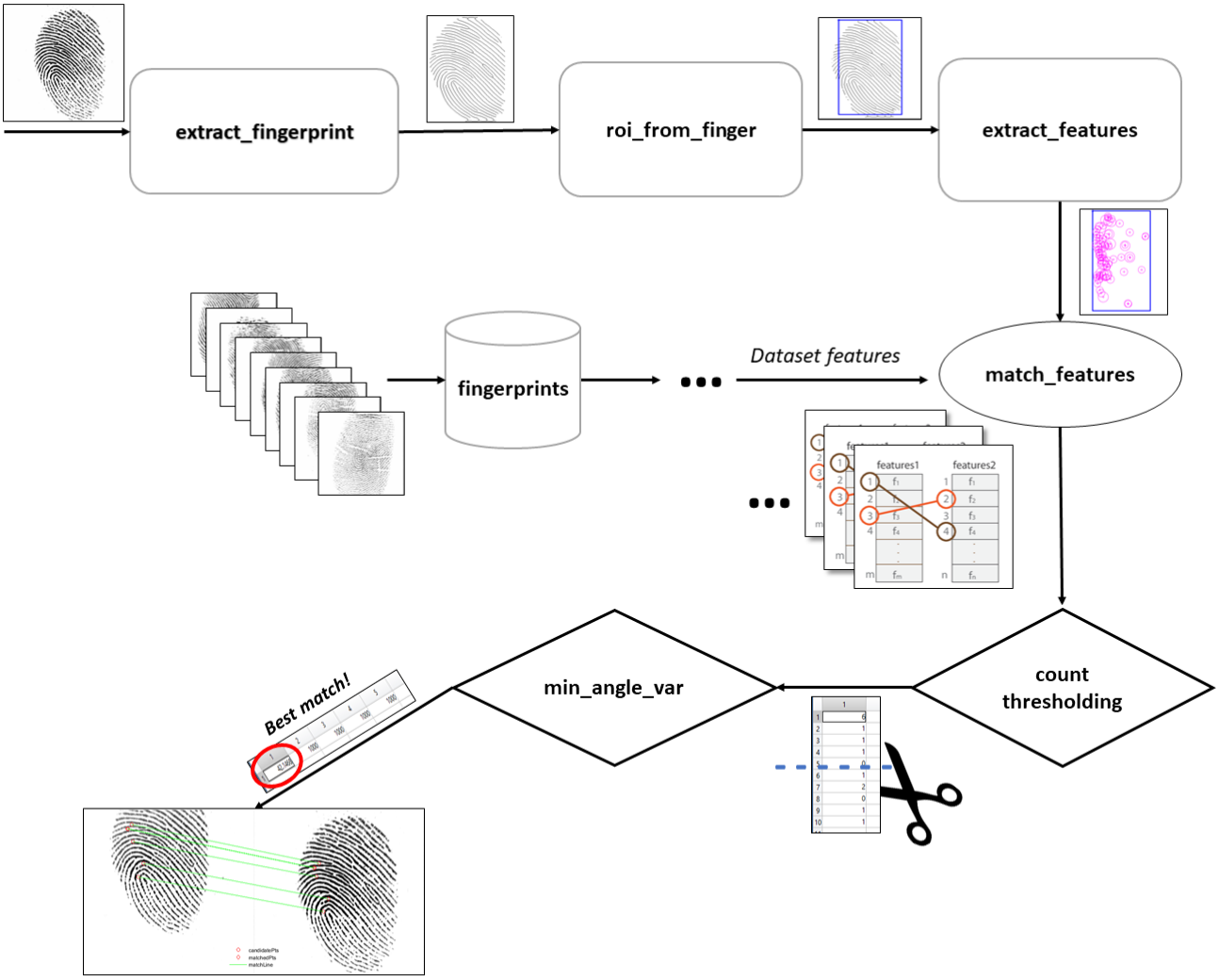
Feature extraction
```
[FEATURES, VALID_POINTS] = extractFeatures(IMG, POINTS)
```

Feature matching
```
indexPairs = matchFeatures(features1, features2)
```

ecc.

Fingerprint Recognition Process

Come già specificato dall'introduzione, questo risulta essere un metodo alternativo per la detection delle caratteristiche di una impronta digitale. Illustriamo quindi il processo che abbiamo ideato e seguito.



extract_fingerprint

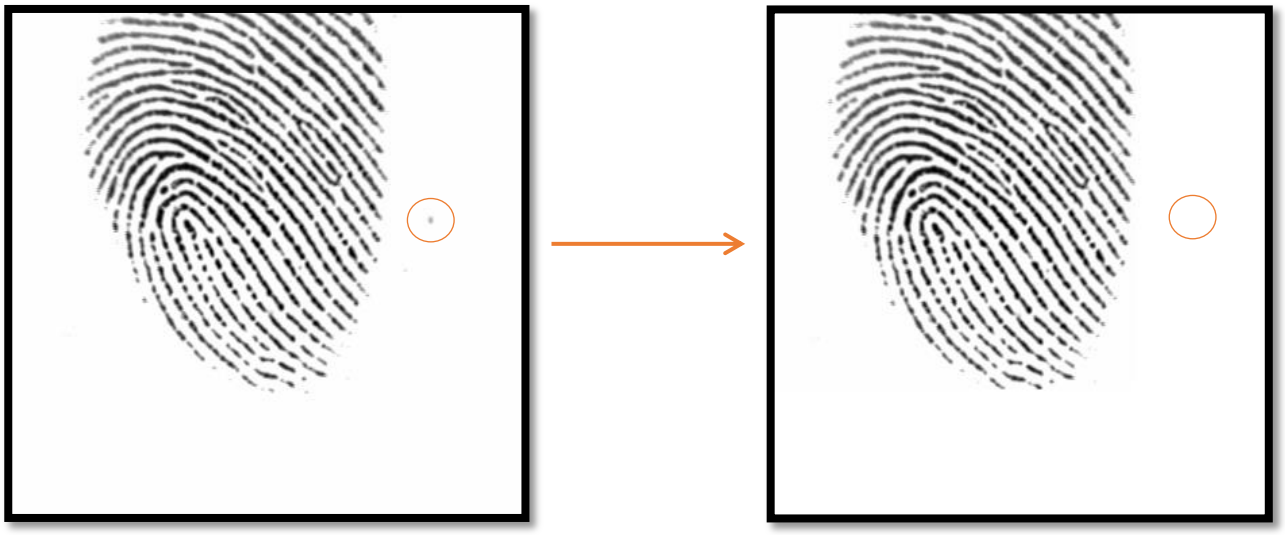
Questa funzione, data l'immagine contenente l'impronta digitale in input su livelli di grigio, ci fornisce in output la corrispondente immagine binaria dopo varie operazioni.

Durante questa fase, le operazioni fatte sono operazioni di enhancement e masking.

Più in particolare, le fasi sono:

1. *Noise reduction*
2. *Image enhancement*
3. *Rifinitura maschera*
4. *Thinning*

La prima fase mette in risalto l'impronta evitando di prendere in considerazione rumori superflui. La ROI viene determinata contando il numero di punti di ampiezza bassa (nero su bianco) sull'immagine in scala di grigio lavorando su asse orizzontale e poi verticale. Se il numero è maggiore di un certo valore, si impone tutta la riga (o colonna) bianca.



Esempio lungo l'asse verticale:

```
for y=1:55
    if numel(find(img(y,:)<200)) < 8
       img(1:y,:) = 255;
       yt=y;
       break
    end
end
```

La seconda fase è l'enhancement dell'immagine. Un tipo di enhancement fatto ad hoc per l'analisi delle impronte digitali. Volevamo basare inizialmente il nostro progetto su tecniche di segmentazione per riuscire ad estrarre dall'immagine l'impronta corrispondente. Tali prove si sono rivelate poco efficaci date varie problematiche. Ad esempio due impronte corrispondenti alla stessa persona risultavano topologicamente molto diverse data la variabilità della procedura di acquisizione dell'impronta.

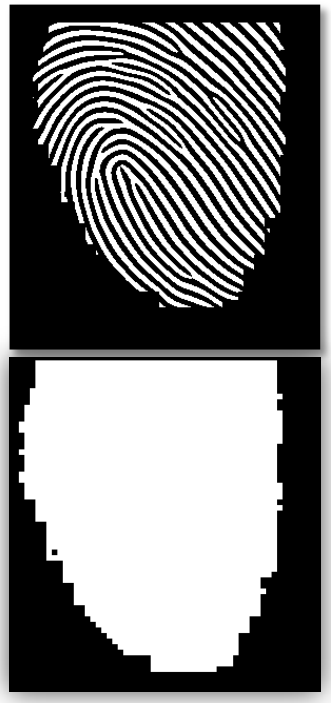
Dunque abbiamo deciso di utilizzare, dopo ricerche, alcune funzioni MATLAB, implementate da Vahid K. Alilou, le quali ci hanno permesso di arrivare ad una più che sufficiente robustezza alla variabilità.

Queste funzioni MATLAB (contenute nella cartella *FExtraction*), nonostante la robustezza, andavano modificate opportunamente per il nostro progetto. Il loro obiettivo era quello di detectare le terminazioni e biforcazioni delle impronte mentre il nostro non comprende questo tipo di analisi (utilizzeremo SURF).

Come risultato di questa operazione otteniamo l'immagine binaria (*binim*) dell'impronta digitale (calcolata dalla ROI della fase 1) e una maschera non rifinita (*mask*).

```
function [ binim, mask] = enhance_func( img )
    addpath('./FExtraction/');
    [enhimg] =  fft_enhance_cubs(img, -1);
    blksze = 5;  thresh = 0.085;
    [normim, mask] = ridgesegment(enhimg, blksze, thresh);
    oimg2 = ridgeorient(normim, 1, 3, 3);
    [~, medfreq] = ridgefreq(normim, mask, oimg2, 32 , 5, 5, 15);
    binim = ridgefilter(normim, oimg2, medfreq.*mask, 0.5, 0.5, 1) > 0;
end
```

img

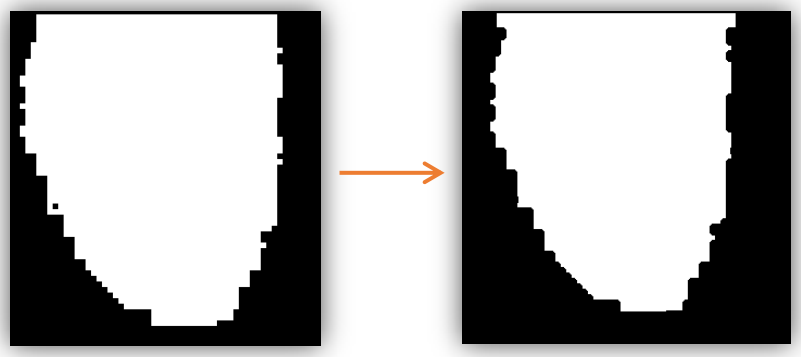


binim

mask

Finalmente siamo giunti alla terza fase:

In questa fase dopo uno scorrimento a blocchi sulla maschera, ricavata precedentemente, per ottenere una maschera più compatta si applicano operazioni di closing. Il motivo per il quale si fa un'operazione di closing (dilatazione->erosione) è quella di ottenere una sotto-maschera che non prenda in considerazione i punti incerti dell'impronta (quelli ai bordi). Di solito i bordi potrebbero contenere informazioni di terminazioni e biforcazioni utili per classificare un'impronta. Nel nostro caso, invece, consideriamo solamente una sotto-maschera centrale dell'impronta in modo da applicare lì una feature detection! Si utilizza un elemento strutturante (SE) a forma di disco con raggio 5. La chiusura viene applicata 3 volte (numero valutato empiricamente).



mask *out*

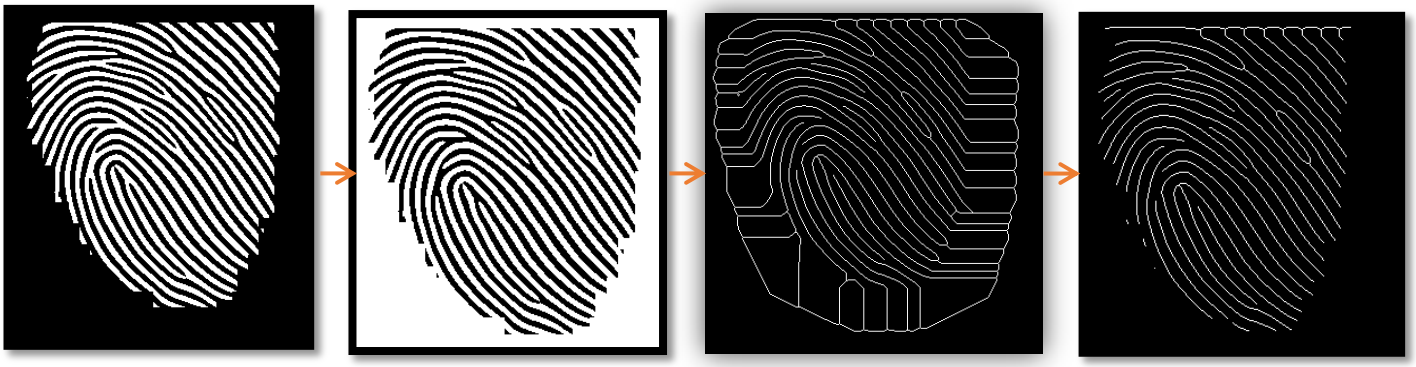
```
b=strel('disk',5);
out=imclose(mask,b);
out=imclose(out,b);
out=imclose(out,b);
```

L'ultima fase del primo blocco prevede il *thinning* della immagine binaria (*binim*) invertendone i colori (nero->background, bianco->impronta). Operazione di thinning viene effettuata "infinite" volte per rendere le linee più sottili possibili.

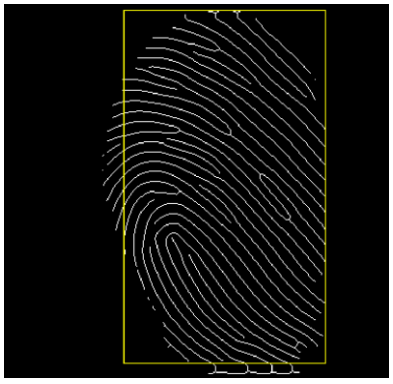
```
%immagine negativa (era analogo imcomplement)
inv_binim = (binim == 0);
%operazione di thinning (Infinite volte) sull'impronta
thinned =  bwmorph(inv_binim, 'thin',Inf);
```

L'output di questa fase sarà il prodotto tra la maschera e l'immagine *thinnata*.

```
%risultato = prodotto punto punto tra maschera (out) e thinned image
%(thinned)
ret=out.*thinned;
```



binim *inv_binim* *thinned* *ret*



roi_from_finger

Restituisce un rettangolo corrispondente ad una precisa area da prendere in considerazione nella fase di feature detection. L'algoritmo ha la stessa struttura dell'algoritmo precedente per la riduzione del rumore. Esso però viene calcolato dall'immagine binaria thinnata.

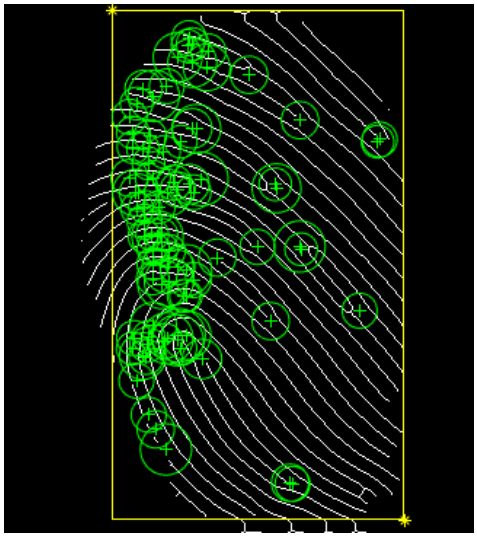
```
[x1,y1,width,height]=roi_from_finger(c_img);
```

L'immagine a sinistra mostra una region of interest (rettangolo giallo) su immagine binaria thinnata.

extract_features

Arrivati a questo punto, alla regione calcolata precedentemente tramite l'algoritmo *roi_from_finger*, viene applicata una sequenza di funzioni MATLAB: *detectSURFFeatures* e successivamente *extractFeatures.*

detectSURFFeatures: funzione MATLAB (integrata in **Computer Vision System Toolbox**), la quale sfrutta l'algoritmo SURF per la detection delle feature presenti nell'immagine dell'impronta digitale.



extractFeatures: altra funzione MATLAB (integrata in **Computer Vision System Toolbox**) la quale, dopo aver estratto i punti di interesse, crea un vettore di feature (descrittori). La funzione ritorna anche il numero di punti validi corrispondenti ai descrittori.

L'immagine a sinistra mostra il risultato ottenuto da questo blocco.

```
c_points = detectSURFFeatures(c_img,'ROI',[x1,y1,width,height]);
[c_f,c_vpts] = extractFeatures(c_img,c_points,'Method','SURF');
```

match_features

Prima di discutere di come avviene la fase di matching delle features tra due impronte, è necessario considerare prima un *dataset* di impronte "campione". Le immagini contenute nel dataset seguiranno lo stesso processo visto precedentemente (a partire da *extract_fingerprint)* per il calcolo delle proprie features (sarà compito della funzione *initialize()*). Le features poi verrano salvate in una matrice Matlab e nella cartella dell'esecuzione del programma, nel caso sia il first run. In particolare il dataset che abbiamo utilizzato è FVC2002 (**Second International Competition for Fingerprint Verification Algorithms**) da cui abbiamo estrapolato 1 immagine per ogni impronta. Abbiamo poi scelto casualmente dal dataset delle impronte con cui testare il nostro programma (Test set). Le immagini fornite dal dataset sono tutte di risoluzione uguali.

Questo blocco prende in input le feature dell'impronta da confrontare e le feature delle impronte del dataset. Con un approccio iterativo effettua il confronto. Ad ogni iterazione si crea un vettore indexPairs che riassume le corrispondenze degli indici. In realtà indexPairs viene gestito come un Cell Array in Matlab. Questo vuol dire che in ogni cella c'è un tipo di dato (nel nostro caso matrici). Questo ci è tornato utile in quanto quando una nuova impronta deve essere classificata, creiamo un Cell Array di dimensioni pari al numero di impronte nel Dataset e associamo alla posizione i-esima in indexPairs gli indici delle rispettive fingerprints (quella del TestSet e DataSet) corrispondenti al match.

Esempio di indexPairs:

Andando ad aprire la prima matrice dell'esempio troviamo gli indici delle rispettive impronte corrispondenti al match!

	1
1	*6x2 uint32*
2	[10,109]
3	[37,128]
4	[17,99]
5	[]
6	[26,18]
7	[8,14;49,82]
8	[]
9	[17,56]
10	[34,81]

indexPairs

	1	2
1	7	5
2	17	19
3	32	53
4	39	37
5	45	66
6	48	57

indexPairs{1}

match degli indici dei rispettivi vettori di feature tra impronta input e impronta 1 del Dataset.

Una volta avuti gli indici corrispondenti al match possiamo applicare un thresholding e la matrice con più "Counts" (corrispondenze) viene designata come la matrice della coppia vincente. Così però non è stato nella pratica. Per arrivare a questo ragionamento, spieghiamo l'idea di introdurre SURF e la sua utilità nella nostra applicazione.

```
fprintf("Computing img "+cnddt_img_name+" and "+uniqueFingerprints(i)+".. ");
indexPairs{i} = matchFeatures(c_f,valid_points{i,1},'Unique',true,'MaxRatio',0.4);
```

SURF

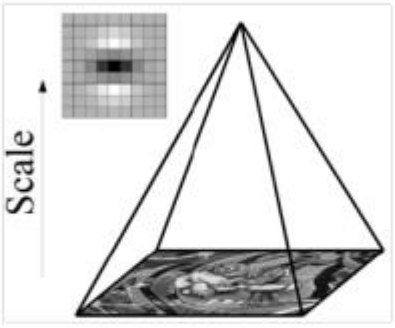
L'algoritmo SURF, acronimo di "*SpeedUp Robust Features*", è un descrittore in grado di rilevare e descrivere i keypoints in breve tempo. SURF deriva dall'algoritmo SIFT (*Scale-Invariant Feature Trasform*), il quale è più robusto ma allo stesso tempo più lento. In SIFT l'analisi dell'immagine richiede molti più secondi rispetto all'algoritmo SURF, infatti esso calcola un descrittore composto da 128 valori causando un grosso overhead nella fase di matching.
Con SURF otteniamo un descrittore più piccolo (64 valori), quindi uno *SpeedUp* nella fase di rilevazione e descrizione dei punti.
Tuttavia avremo una minore robustezza in caso di forti cambi di illuminazione e di rotazione del soggetto.

fase di detection

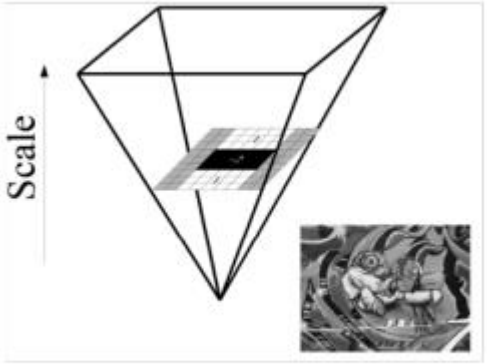
L'algoritmo SURF per aumentare significativamente la velocità di localizzazione dei punti si basa sull'analisi dell'immagine integrata, da essa si calcola il determinante della matrice hessiana. Tramite questo processo si è in grado di far risaltare i cambi di intensità luminosa, evidenziando gli angoli ma non i bordi degli oggetti, tale processo è velocizzato tramite l'utilizzo di kernel che eseguono una media pesata dell'intensità del pixel (convoluzione).

rappresentazione scale-space

I punti di interesse vengono rilevati con fattori di scala differenti, tramite lo Scale-space, perché la ricerca delle corrispondenze spesso richiede il confronto di immagini in cui le feature sono viste a distanze diverse. L'algoritmo dello Scale-space è implementato generalmente (nelle SIFT) come una piramide di immagini, ottenuta applicando funzioni di tipo Gaussiano alle diverse immagini mentre, nell'algoritmo SURF, grazie all'impiego dei Box-filter e l'integrale dell'immagine, non è più necessario applicare il medesimo filtro all'immagine filtrata e scalata, ma si applica il filtro a ogni dimensione direttamente sull'immagine originale. L'analisi Scale-Space viene quindi effettuata con l'up-scaling del filtro piuttosto che il ridimensionamento iterativo dell'immagine filtrata. I vari layer della piramide sono ottenuti quindi filtrando l'immagine con filtri via via sempre più grandi.



SIFT



SURF

il descrittore SURF

Il Descrittore SURF si calcola a partire dall' assegnazione dell'orientamento. Per ogni keypoint estratto dalla precedente fase viene assegnato un orientamento in modo da raggiungere l'invarianza rispetto alla rotazione dell'immagine.
La prima fase riguarda il calcolo della risposta wavelet di Haar nelle direzioni verticale e orizzontale, calcolata ancora una volta sull'immagine integrale. La direzione dominante è stimata calcolando la somma di tutte le risposte con una finestra scorrevole. Il vettore con modulo maggiore tra quelli calcolati con le diverse finestre è l'orientamento principale del punto di interesse.

componenti del descrittore

Una volta individuato l'orientamento, la determinazione del descrittore avviene costruendo una griglia quadrata centrata nel punto di interesse e orientata lungo l'orientamento principale.
La griglia viene divisa in sotto regioni 4x4 (in modo da preservare il maggior numero di info spaziali).
Per ognuna di queste regioni viene calcolata la risposta wavelet di Haar nelle due direzioni. Quindi le risposte calcolate verranno sommate a tutte le altre risposte ottenute nella sotto regione.
Inoltre verranno estratti anche i valori assoluti delle somme (precedentemente calcolate) per preservare l'informazione riguardante l'intensità dei pixel.
Otteniamo dunque un descrittore di 4 dimensioni. Calcolando questo su ogni blocchetto della sotto regione 4x4, fanno un totale di 64 valori.

count_thresholding & min_angle_variance

Essendo SURF non perfettamente flessibile alla rotazione, abbiamo ritenuto opportuno fornire a SURF un supporto riguardo a tale problematica, optando per un matching a cascata di 2 blocchi.

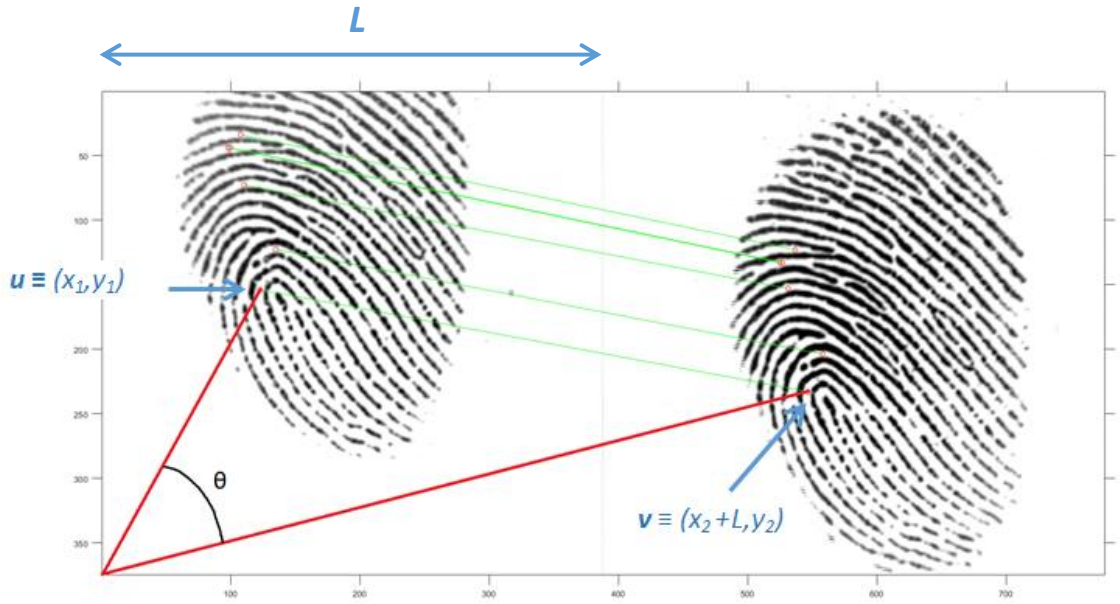
```
matchedIndex=find((matchCount)>=max(matchCount)/2);
fprintf("Count Thresholding!!! removed "+ (length-size(matchedIndex,1)) +" fingerprints!!!\n");
min_index_varianza=(matchedIndex(1));

for i=1:size(matchedIndex,1)
    if(varianza(1,matchedIndex(i))<varianza(min_index_varianza))
       min_index_varianza=matchedIndex(i);
    end
end
matchedIndex=min_index_varianza;
```

count_thresholding

Nel primo blocco estrapoliamo le coppie di immagini col maggior *matchCount* (si prendono le coppie con conteggio di match più alto della metà del massimo conteggio di match).

min_angle_variance

Una volta filtrate le soluzioni "tagliando" quelle con matchCount troppo basso, il secondo blocco impone il calcolo della varianza di tutti gli angoli, composti dalle rette passanti per l'origine e per i punti corrispondenti ai match. Per aggiungere chiarezza, alleghiamo un'immagine esplicativa.



Questa operazione viene iterata su ogni coppia di punti. Siamo sicuri al 100% che questa corrispondenza sia 1 a 1 (e non 1 a molti) grazie all'utilizzo del parametro *Unique* nella funzione matchFeatures.

Una volta calcolati tutti gli angoli, si calcola lo scostamento dalla media (varianza). Se la varianza è bassa vuol dire che tutti i punti hanno un angolo simile (in pratica le linee verdi che uniscono i match tendono ad essere parallele). Se la varianza invece è alta vuol dire che l'angolo θ è molto variabile e quindi, ipotizzando che le impronte siano della stessa risoluzione, esse molto probabilmente non corrispondono.

$$\cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \; ||\mathbf{v}||}$$

Siano u e v due punti. Riusciamo ad ottenere θ tramite arcocoseno del risultato di questa espressione.

L corrisponde alla lunghezza della prima immagine (viene utilizzato come offset per prendere l'angolo giusto per la seconda immagine).

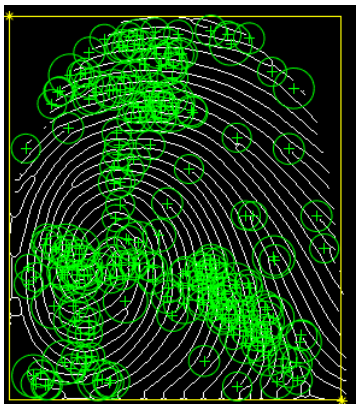
```
function y = variance_angle(matchedPoints1,matchedPoints2,img1)

    length=matchedPoints1.Count;
    if(length==0 || length==1 || length==2)
       y=1000;
    else
        C=zeros(length,1);
        for i=1:length
            x1=matchedPoints1.Location(i,1);
            y1=matchedPoints1.Location(i,2);
            x2=matchedPoints2.Location(i,1);
            y2=matchedPoints2.Location(i,2);
            A=[x1,x2+size(img1,2)];
            B=[y1,y2];
            C(i)=angle_between_vectors(A,B);
        end

        %più le linee sono parallele più sarà bassa la varianza =>
        %corrispondenza pratica!
        y=var(C);
    end
end
```

applicazioni pratiche

- Esempio 1

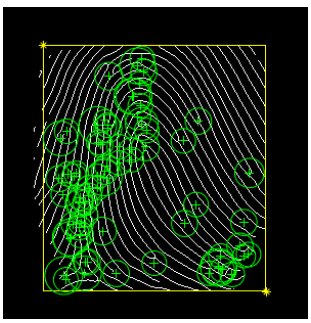
I keypoints delle immagini già presenti nel dataset vengono calcolati nel first run del programma (si veda funzione *initialize*). Qualora fossero già stati calcolati il programma lo capirebbe e restituirà una stringa di conferma. Questo esempio rappresenta esattamente il caso in cui il numero di match non è informazione sufficiente per discriminare l'impronta corrispondente. Ci viene in aiuto la varianza degli angoli.

```
Valid points have already been calculated! returning them...
Computing img c8_4.tif and 1.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 393.0509
Computing img c8_4.tif and 2.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c8_4.tif and 3.tif.. numero di corrispondenze: 7
varianza media di tutti i punti: 50.3243
Computing img c8_4.tif and 4.tif.. numero di corrispondenze: 9
varianza media di tutti i punti: 187.5502
Computing img c8_4.tif and 5.tif.. numero di corrispondenze: 10
varianza media di tutti i punti: 214.7541
Computing img c8_4.tif and 6.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 228.4133
Computing img c8 4.tif and 7.tif.. numero di corrispondenze: 8
varianza media di tutti i punti: 299.1008
Computing img c8 4.tif and 8.tif.. numero di corrispondenze: 5
varianza media di tutti i punti: 23.1572
Computing img c8_4.tif and 9.tif.. numero di corrispondenze: 5
varianza media di tutti i punti: 234.2206
Computing img c8_4.tif and 10.tif.. numero di corrispondenze: 5
varianza media di tutti i punti: 351.6782
Count Thresholding!!! removed 3 fingerprints!!!
L'immagine c8_4.tif ha il miglior numero di match (5) con
l'impronta 8.tif totalizzando una varianza di 23.1572!
```



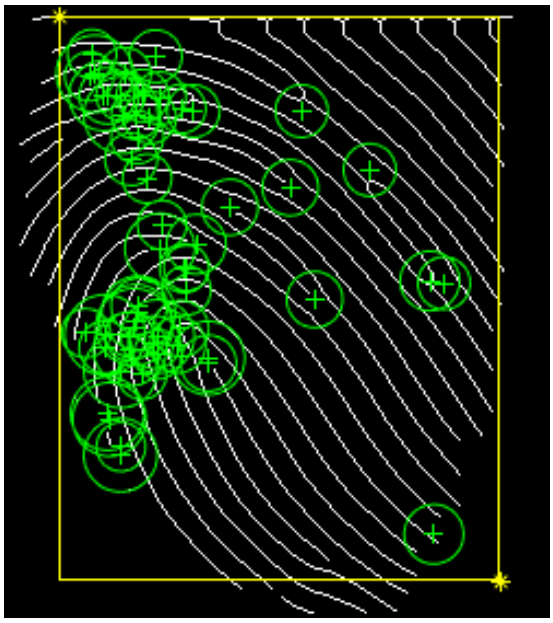
- Esempio 2

```
Valid points have already been calculated! returning them...
Computing img c7 3.tif and 1.tif.. numero di corrispondenze: 0
varianza media di tutti i punti: 1000
Computing img c7_3.tif and 2.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c7_3.tif and 3.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 148.7409
Computing img c7_3.tif and 4.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c7 3.tif and 5.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c7_3.tif and 6.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c7 3.tif and 7.tif.. numero di corrispondenze: 4
varianza media di tutti i punti: 7.7224
Computing img c7_3.tif and 8.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c7_3.tif and 9.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c7_3.tif and 10.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Count Thresholding!!! removed 7 fingerprints!!!
L'immagine c7_3.tif ha il miglior numero di match (4) con
l'impronta 7.tif totalizzando una varianza di 7.7224!
```



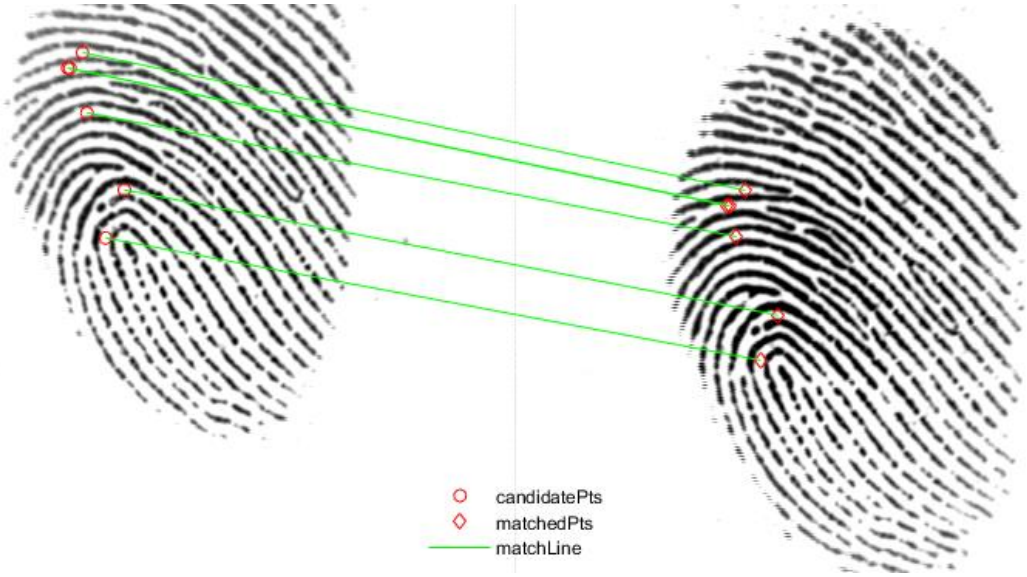
- *Esempio 3 (initialize+feature match)*
  Simulo first run rimuovendo valid_points.mat

```
It appears nobody ever calculated valid_points on fingerprints!
Doing it by myself!
1.tif...
2.tif...
3.tif...
4.tif...
5.tif...
6.tif...
7.tif...
8.tif...
9.tif...
10.tif...
Computing img c1 2.tif and 1.tif.. numero di corrispondenze: 6
varianza media di tutti i punti: 42.1468
Computing img c1_2.tif and 2.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c1_2.tif and 3.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c1_2.tif and 4.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c1_2.tif and 5.tif.. numero di corrispondenze: 0
varianza media di tutti i punti: 1000
Computing img c1_2.tif and 6.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c1 2.tif and 7.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c1_2.tif and 8.tif.. numero di corrispondenze: 0
varianza media di tutti i punti: 1000
Computing img c1_2.tif and 9.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c1_2.tif and 10.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Count Thresholding!!! removed 9 fingerprints!!!
L'immagine c1 2.tif ha il miglior numero di match (6) con
l'impronta 1.tif totalizzando una varianza di 42.1468!
```



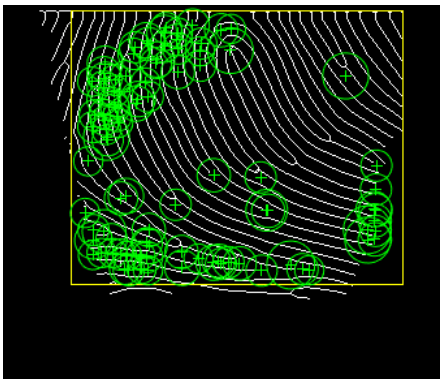
La ragione per cui la varianza viene saturata a 1000 è data dal fatto che il numero di corrispondenze è troppo piccolo per poter considerare un match come valido (si veda implementazione funzione variance_angle precedentemente esposta).

Si pensi ad esempio a 2 immagini con pochissimi match. La loro media sarà informazione poco significativa data la scarsità di dati. All'aumentare dei match, la media e quindi la varianza assumeranno valori altamente informativi.



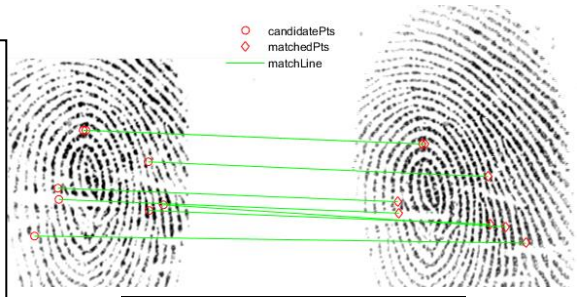
- *Esempio 4*

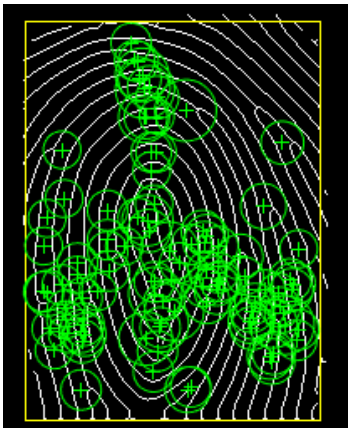
```
Valid points have already been calculated! returning them...
Computing img c10_6.tif and 1.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 145.7274
Computing img c10_6.tif and 2.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c10_6.tif and 3.tif.. numero di corrispondenze: 5
varianza media di tutti i punti: 370.7698
Computing img c10_6.tif and 4.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c10_6.tif and 5.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c10_6.tif and 6.tif.. numero di corrispondenze: 4
varianza media di tutti i punti: 50.8399
Computing img c10_6.tif and 7.tif.. numero di corrispondenze: 4
varianza media di tutti i punti: 41.2522
Computing img c10_6.tif and 8.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 81.3734
Computing img c10_6.tif and 9.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c10_6.tif and 10.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 30.9035
Count Thresholding!!! removed 4 fingerprints!!!
L'immagine c10_6.tif ha il miglior numero di match (3) con
l'impronta 10.tif totalizzando una varianza di 30.9035!
```



- *Esempio 5*

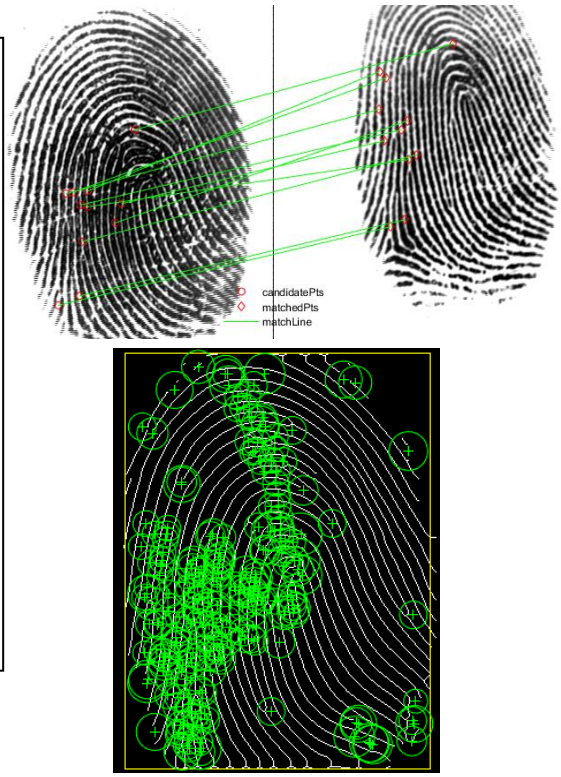
```
Valid points have already been calculated! returning them...
Computing img c9_1.tif and 1.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img c9_1.tif and 2.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c9_1.tif and 3.tif.. numero di corrispondenze: 4
varianza media di tutti i punti: 52.2896
Computing img c9_1.tif and 4.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 262.8539
Computing img c9_1.tif and 5.tif.. numero di corrispondenze: 0
varianza media di tutti i punti: 1000
Computing img c9_1.tif and 6.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c9_1.tif and 7.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img c9_1.tif and 8.tif.. numero di corrispondenze: 4
varianza media di tutti i punti: 142.1558
Computing img c9_1.tif and 9.tif.. numero di corrispondenze: 8
varianza media di tutti i punti: 19.5919
Computing img c9_1.tif and 10.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Count Thresholding!!! removed 7 fingerprints!!!
L'immagine c9 1.tif ha il miglior numero di match (8) con
l'impronta 9.tif totalizzando una varianza di 19.5919!
```





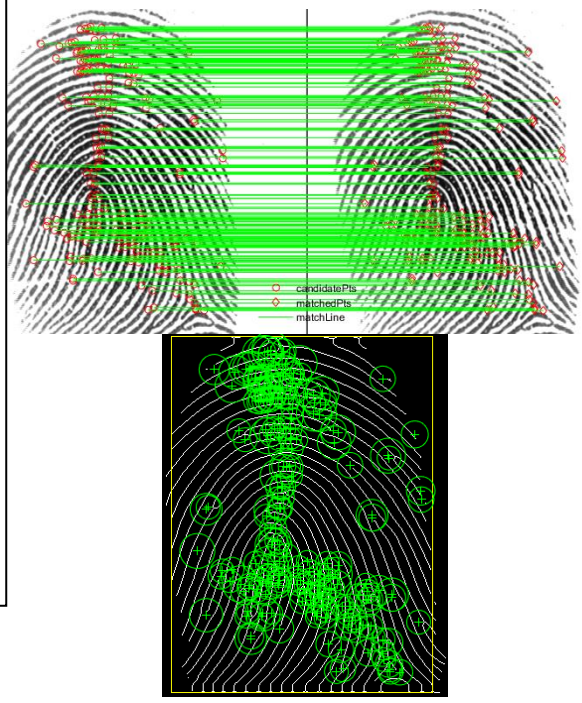
- *Esempio 6*

```
Valid points have already been calculated! returning them...
Computing img c3_6.tif and 1.tif.. numero di corrispondenze: 5
varianza media di tutti i punti: 543.6537
Computing img c3_6.tif and 2.tif.. numero di corrispondenze: 11
varianza media di tutti i punti: 392.739
Computing img c3_6.tif and 3.tif.. numero di corrispondenze: 11
varianza media di tutti i punti: 45.8005
Computing img c3_6.tif and 4.tif.. numero di corrispondenze: 9
varianza media di tutti i punti: 190.1361
Computing img c3_6.tif and 5.tif.. numero di corrispondenze: 10
varianza media di tutti i punti: 216.0025
Computing img c3_6.tif and 6.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 552.5523
Computing img c3_6.tif and 7.tif.. numero di corrispondenze: 6
varianza media di tutti i punti: 361.4555
Computing img c3_6.tif and 8.tif.. numero di corrispondenze: 6
varianza media di tutti i punti: 132.9782
Computing img c3 6.tif and 9.tif.. numero di corrispondenze: 8
varianza media di tutti i punti: 191.8917
Computing img c3_6.tif and 10.tif.. numero di corrispondenze: 6
varianza media di tutti i punti: 368.3338
Count Thresholding!!! removed 2 fingerprints!!!
L'immagine c3 6.tif ha il miglior numero di match (11) con
l'impronta 3.tif totalizzando una varianza di 45.8005!
```



- *Esempio 7*

```
Valid points have already been calculated! returning them...
Computing img d2.tif and 1.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Computing img d2.tif and 2.tif.. numero di corrispondenze: 202
varianza media di tutti i punti: 9.2443
Computing img d2.tif and 3.tif.. numero di corrispondenze: 0
varianza media di tutti i punti: 1000
Computing img d2.tif and 4.tif.. numero di corrispondenze: 2
varianza media di tutti i punti: 1000
Computing img d2.tif and 5.tif.. numero di corrispondenze: 6
varianza media di tutti i punti: 131.3456
Computing img d2.tif and 6.tif.. numero di corrispondenze: 4
varianza media di tutti i punti: 186.6399
Computing img d2.tif and 7.tif.. numero di corrispondenze: 3
varianza media di tutti i punti: 18.164
Computing img d2.tif and 8.tif.. numero di corrispondenze: 7
varianza media di tutti i punti: 274.6992
Computing img d2.tif and 9.tif.. numero di corrispondenze: 6
varianza media di tutti i punti: 120.8178
Computing img d2.tif and 10.tif.. numero di corrispondenze: 1
varianza media di tutti i punti: 1000
Count Thresholding!!! removed 9 fingerprints!!!
L'immagine d2.tif ha il miglior numero di match (202) con
l'impronta 2.tif totalizzando una varianza di 9.2443!
```



Nell'ultimo esempio testiamo l'algoritmo con un'immagine presente nel dataset. Si noti il valore relativamente basso della varianza e valore alto di corrispondenze.

Riferimenti

https://it.wikipedia.org/wiki/Impronta_digitale

https://www.laleggepertutti.it/299221_impronte-digitali-ultime-sentenze

https://www.cartaidentita.interno.gov.it/modalita-acquisizione-impronte/

http://bias.csr.unibo.it/fvc2002/

http://www.dr-alilou.ir/

https://amslaurea.unibo.it/6855/1/Andrea_Annovi_tesi.pdf

http://tesi.cab.unipd.it/33854/1/Tesi.pdf

Ringraziamenti

Questo progetto ci ha dato la possibilità di toccare con mano un mondo nuovo ovvero quello dei Sistemi Multimediali. Con questo esame, siamo stati dolcemente slegati dai vincoli didattici e grazie al prof. Rinaldi siamo stati catapultati, tramite un progetto di livello applicativo, in un contesto reale. Abbiamo avuto la libertà di scelta del progetto il ché inizialmente ci ha sorpresi. Tale libertà, ci ha concesso di scegliere un argomento per noi interessante e stimolante e per questo lo ringraziamo.

Valerio Mennillo
Antonio Mennella