Palmy - android palmistry application

Orlando Aprea N97/226,
Andrea Sorrentino N97/221,
Alessandro Serrapica N97/213

Universitá degli studi di Napoli Federico II - Corso di Visione Computazionale

September 27, 2017

Abstract

La lettura del palmo é un argomento che ha ottenuto notevole risalto negli ultimi anni per interessi commerciali ma anche di ricerca.
Questo progetto, nell'ambito dello sviluppo di una applicazione mobile, si prefissa lo scopo di individuare ed analizzare le principali linee del palmo della mano, mediante algoritmi di visione computazionale. Ció consentirá, infine, di mostrare, in accordo con la lettura del palmo della mano tradizionale cinese, informazioni circa il carattere e il futuro delle persone.
In questo documento mostreremo i vari approcci allo sviluppo di questa applicazione, le problematiche sorte, i risultati ottenuti ed i futuri sviluppi.

1 Introduzione

L'identificazione e la lettura del palmo della mano, negli ultimi anni, é stata una delle caratteristiche biometriche che ha ottenuto piú risalto. Questo perché é stato dimostrato che le linee che caratterizzano il palmo della mano di una persona si formano tra il terzo e il quinto mese di gravidanza, restando immutate per tutta la vita. Inoltre, ogni individuo ha una conformazione unica del palmo e questo aspetto ha dato adito a ricerche mirate ad utilizzare il palmo della mano come sistema di riconoscimento biometrico affidabile.
Questo progetto, come detto, si propone di identificare le principali tre linee della mano e, in accordo con la tradizione cinese, identificare e descrivere il carattere delle persone o predirne il futuro. Ció rappresenta, ovviamente, un pretesto per approfondire una tematica complessa come il riconoscimento di caratteristiche biometriche, avvalendosi di algoritmi di visione computazionale unitamente a strumenti quali ad esempio OpenCV.
Secondo la tradizione cinese, le tre principali linee della mano sono :

- Linea del cuore : fornisce informazioni sulla sfera affettiva dell'utente e la sua predisposizione ad una vita familiare o a relazioni stabili e durature.
- Linea della testa o dell'intelligenza : descrive le attitudini e le capacitá lavorative.
- Linea della vita : fornisce informazioni circa la salute, la longevitá e la qualitá stessa della vita.

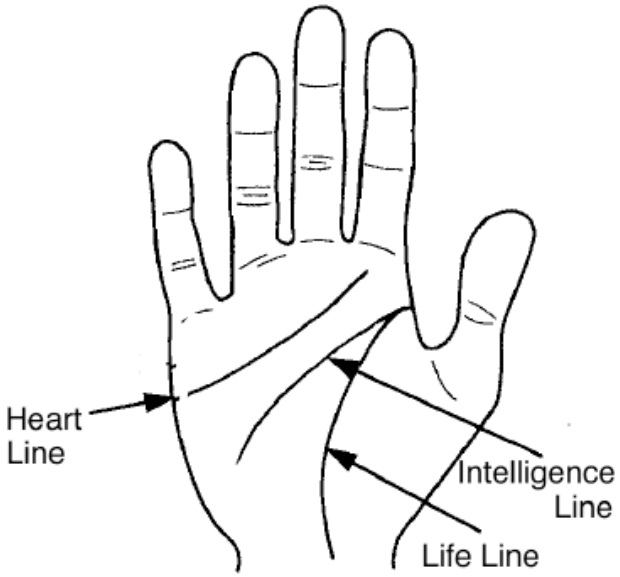

Figure 1: **Le principali linee della mano**

L'analisi delle suddette linee si effettua in termini di lunghezza e di angolazione. Come detto, essendo di nostro interesse l'identificazione delle linee e non l'interpretazione chiromantica delle stesse é stato realizzato un piccolo 'database' d'esempio in grado di fornire diverse predizioni basandosi appunto su lunghezza e angolazione delle linee identificate.

2 Analisi del problema

Le caratteristiche del palmo della mano elencate precedentemente, se da una parte hanno, come detto, attirato l'attenzione di molti ricercatori, dall'altra costituiscono una serie di ostacoli che é necessario aggirare per un riconoscimento accurato.
Innanzitutto, si é constatato come le condizioni di luce siano determinanti per una corretta individuazione delle linee e per la gestione del rumore. Data la natura del progetto é stato necessario creare delle condizioni ottimali per la cattura dell'immagine originale che vanno ad alleviare la problematica. Si é deciso infatti di catturare immagini con il palmo della mano ad una distanza di circa 20cm dal dispositivo uti-

lizzando il flash della fotocamera in modalitá torcia in modo da illuminare in modo omogeneo l'intera regione d'interesse. Inoltre, data l'alta risoluzione delle foto catturate, sará poi applicato un taglio, come suggerito dalla UI dell'applicazione, al fine di andare a lavorare su di un'immagine dalle dimensioni minori che contenga esclusivamente il palmo della mano. L'immagine risultante avrá una dimensione di 500 per 500 pixel.
Come detto, peró, anche la natura stessa del palmo ha comportato diverse problematiche. L'etereogeneitá dei palmi e le differenti conformazioni delle linee sia per forma che per spessore hanno impedito di effettuare delle assunzioni su di essi a livello algoritmico. É stato necessario quindi implementare degli algoritmi capaci di evidenziare le linee con maggiore risalto per poi andare successivamente ad analizzare solo quelle di reale interesse.
Le caratteristiche intrinseche della pelle umana hanno comportato delle problematiche in termini di rumore. Nel tentativo di evitarle sono stati implementati degli algoritmi volti alla pulizia dell'immagine e all'identificazione di quei falsi positivi che avrebbero potuto pregiudicare l'identificazione.
Infine, la decisione di sviluppare il tutto come applicazione mobile, ci ha spinti a trovare soluzioni computazionalmente efficienti al fine di garantire tempi di risposta rapidi a fronte delle numerose operazioni e trasformazioni che vengono effettuate sull'immagine originale. In tal senso, come si vedrá in seguito, é stata implementata una tecnica innovativa nel riconoscimento delle principali linee del palmo della mano, ulteriormente modificata nello sviluppo di questo progetto, che garantisce buoni risultati in tempi brevi nonostante si vada a catturare immagini ad alta risoluzione.

3 Workflow dell'applicazione

Lo sviluppo di un'applicazione non puó prescindere da uno studio adeguato sull'efficacia dell'interfaccia utente sviluppata e della sua usabilitá in termini pratici. Sebbene questo non sia l'obiettivo primario di questo progetto, si é cercato di creare un'esperienza utente semplice ed al contempo accattivante. Il risultato dello studio effettuato é stato un workflow estremamente lineare e intuitivo, come illustrato nella figura 2

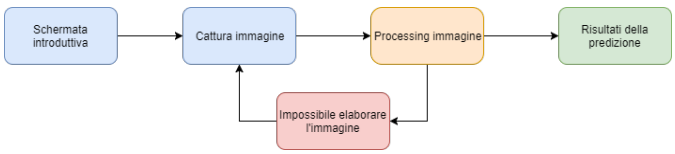

Figure 2: **Workflow dell'applicazione** - I 5 step principali

Di seguito saranno mostrate nello specifico le varie schermate dell'interfaccia utente e verranno spiegate e contestualizzate le scelte effettuate nella progettazione delle stesse.

Schermata introduttiva La prima schermata ha come scopo quello di creare un environment adeguato alle finalitá del progetto e che coinvolga da subito l'utente. Si é cercato di raggiungere questi obiettivi introducendo un logo moderno e delle spiegazioni brevi e chiare. Inoltre, ponendo al centro dell'UI il pulsante per scattare la foto, si é potuta esaltare l'affordance dello stesso.

Figure 3: **Schermata introduttiva**

Cattura dell'immagine In questa fase l'utente é invitato a scattare una foto. Questa schermata é, ovviamente, la piú importate dal punto di vista prestazionale. Infatti, una foto scattata male puó avere potenzialmente un impatto molto negativo sulle operazioni di processing attuate sull'immagine. Per 'suggerire' all'utente la giusta modalitá di acquisizione della foto sono stati inseriti alcuni elementi grafici pertinenti. In ogni caso sono stati effettuati dei controlli che consentono di comprendere se l'utente ha effettivamente inquadrato la mano in modo corretto o meno.

Figure 4: **Schermata per la cattura dell'immagine**

Processing dell'immagine L'operazione di processing avviene in genere in un tempo molto breve. Tuttavia, quest'ultimo dipende anche dalle caratteristiche hardware del dispositivo. Ció comporta tempi di attesa differenti a seconda del device utilizzato. Per risolvere il problema dell'attesa é mostrata all'utente una dialog activity con una infinite progress bar. Questo espediente consente di mantenere alta l'attenzione dell'utente durante il processing dell'immagine.

Figure 5: **Schermata mostrata durante il processing dell'immagine**

Dialog errore processing immagine In caso di errata inquadratura del palmo della mano, oppure di mancato riconoscimento delle linee principali, é mostrato all'utente un feedback. L'utente é invitato a ripetere l'operazione.

Figure 6: **Schermata di errore**

Schermata dei risultati L'ultima schermata é predisposta a mostrare l'esito dell'elaborazione dell'immagine. Si é scelto di dare un'impostazione molto semplice a questa activity fornendo una chiara visione dei risultati, mediante la sovrapposizione delle linee ricavate sull'immagine originale e tramite una descrizione testuale delle caratteristiche delle linee stesse.

4 Approcci risolutivi per l'estrazione delle linee

La letteratura riguardante il riconoscimento delle linee nel palmo della mano é molto estesa. Nel panorama analizzato sono stati individuati due filoni di ricerca principali:

- Il primo basato su approcci di tipo edge detection (Canny, Sobel,Prewitt, etc)

Figure 7: **Schermata per mostrare i risultati**

- Il secondo fondato su tecniche molto usate in ambito di feature extraction (come il morphological TOP-HAT filtering)

Le tecniche di cui sopra, presentano vantaggi e svantaggi e per verificarne l'efficacia sono state implementate entrambe. Dopo una fase di testing é stato preferito il secondo filone di ricerca e sono state individuate delle criticitá cui si é tentato di porre rimedio in questo progetto.

4.1 Canny edge detection

Di seguito verrá introdotta la prima tecnica implementata per l'estrazione delle linee.
L'algoritmo Canny edge detection appartiene alla famiglia degli edge detectors : algoritmi in grado di riconoscere bordi all'interno di un'immagine.
L'algoritmo prevede tre fasi principali :

- Preprocessing dell'immagine in ingresso
- Riconoscimento dei bordi candidati
- Riconoscimento dei bordi definitivi

Preprocessing dell'immagine in ingresso Al fine di ridurre il piú possibile la quantitá di rumore e di errori di rilevamento si é reso necessario trasformare l'immagine originale nella sua equivalente in scala di grigi.
Inoltre é stato applicato un filtro Gaussiano con lo scopo di ridurre il rumore all'interno dell'immagine, smussando i bordi poco rilevanti.

Riconoscimento dei bordi candidati Tramite il calcolo dell' intensitá del gradiente dell'immagine vengono riconosciuti i bordi candidati valutando il contrasto tra i pixel.

Riconoscimento dei bordi definitivi Utilizzando il gradiente calcolato in precedenza per ogni pixel tale valore viene confrontato con delle soglie di tolleranza denominate 'High-Threshold' e 'LowThreshold'. Se il gradiente del pixel é al di sopra dell' 'HighThreshold' tale pixel viene accettato come facente parte di un bordo, se al di sotto del 'LowThreshold' viene rigettato, mentre se é all'interno dell' intervallo tra le

due soglie viene accettato solo se connesso a pixel con gradiente superiore all' 'HighThreshold'.

4.1.1 Utilizzo dell' algoritmo nell'applicativo

All'interno del palmo della mano, salvato in un'immagine 500x500 pixel, se assumiamo che ogni linea abbia uno spessore di 41 pixel e lunghezza paragonabile all'altezza dell'immagine (come nella pressoché totatilá dei palmi umani), é facile notare che circa il 24% dell'immagine sará dedicato alla rappresentazione delle linee.
Sia 'x' la percentuale dell'immagine che rappresenta le linee, avremo dunque:

$$X = \left[\frac{41*3*500}{500*500}\right] * 100 = 24.6$$

Partendo da questa assunzione ci si é resi conto che un corretto output di Canny con bordi di larghezza di 41 pixel dovesse avere una percentuale di bordi rispetto all'immagine tra il 20% e il 27%.
Ció é stato possibile mediante la seguente implementazione:

begin
  $highThreshold \longleftarrow colorMean + colorStandardDeviation$;
  $lowThreshold \longleftarrow colorMean - colorStandardDeviation$;
  $linePercentage \longleftarrow 0$;
  while $20 \leq linePercentage$ **and** $27 \geq linePercentage$ **do**
    $result \longleftarrow Canny(inputImg, highThreshold, lowThreshold)$;
    $result \longleftarrow enlarging(result)$;
    $linePercentage \longleftarrow calculateLinePercentage(result)$;
    if $linePercentage < 20$ **then**
      $highThreshold \longleftarrow highThresholdLowering()$;
    end
    if $linePercentage > 27$ **then**
      $highThreshold \longleftarrow highThresholdUpping()$;
    end
  end
  $result \longleftarrow Canny(inpuImg, highThreshold, lowThreshold)$;
  $result \longleftarrow enlarging(result)$;
  $result \longleftarrow thinning(result)$;
end

All'interno del precedente pseudo-codice, vengono tarati i valori di high e low Threshold da utilizzare all'interno di Canny per rilevare il giusto quantitativo di bordi necessario per individuare le linee della mano. Una volta fatto ció viene eseguito nuovamente Canny con i valori di threshold corretti e viene poi effettuata l'operazione di enlarging e di thinning per ridurre il numero di linee spezzate e il rumore attorno ad esse.

Questa soluzione é stata peró scartata a causa di alcune problematiche sorte :

Figure 8: **Risultato dell'algoritmo Canny Edge Detection**

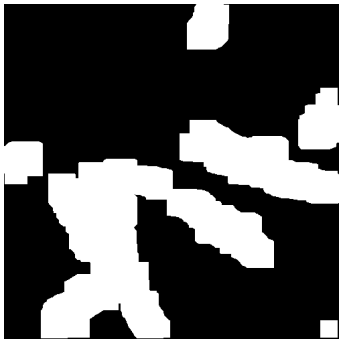

Figure 9: **Risultato dell'algoritmo Enlarging**

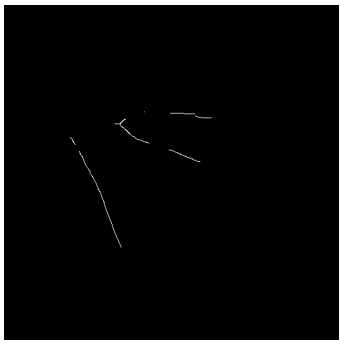

Figure 10: **Risultato dell'algoritmo Thinning**

- Performance non adeguate all'uso su dispositivi mobile.
- Difficile distinzione tra veri bordi e quelli causati dal rumore.

4.2 Morphological TOP-HAT Filtering

Nella seguente sottosezione viene presentato l'algoritmo che si é scelto di utilizzare facendo riferimento al paper [1]. Quest'ultimo lavora su immagini di 150x150 px, mentre (come giá accennato) nel presente elaborato si é scelto di operare con immagini di 500x500 px, scalando ove necessario i parametri delle operazioni effettuate. Ulteriori modifiche attuate saranno evidenziate durante la spiegazione dell'algoritmo. Come giá descritto nel paper di riferimento, é possibile scindere questo algoritmo in due fasi:

- Preprocessing dell'immagine in ingresso
- Estrazione delle linee principali

A sua volta ogni fase é divisa in piú step che andiamo ad analizzare singolarmente.

4.2.1 Fase di preprocessing

In questa fase sono applicate all'immagine operazioni atte ad evitare di catturare, nella fase successiva, linee che di fatto costituiscono rumore ai fini di analisi. Si susseguono, quindi, i seguenti passi.

Contrast enhancement L'acquisizione del palmo della mano avviene ad una distanza di circa 20cm dal device, con flash attivo in modalitá 'torcia'. Se da un lato ció consente di poter fruire di un'illuminazione generalmente omogenea, dall'altro questo stratagemma comporta anche l'acquisizione di immagini con un basso contrasto. Per ovviare a questo problema si é utilizzata una tecnica di 'stretching dell'istogramma' che consente di aumentare il contrasto e di migliorare cosí la visibilitá delle linee del palmo. La funzione opencv di cui si é fatto uso per raggiungere questo obiettivo é *'equalizeHist'* ed il risultato é mostrato in figura 11.

Figure 11: **Immagine contrastata** - risultato dell'operazione di equalizeHist

Bilateral filtering Il paper di riferimento opera su un dataset di immagini, del database PolyU, catturate in situazioni ottimali. L'applicazione al mondo reale della tecnica proposta porta con sé dei fattori di perturbazione che possono disturbare il riconoscimento delle linee. Per ovviare a questa situazione e riportare l'algoritmo in condizioni ottimali é stato introdotto questo nuovo step. Si applica quindi un'operazione di bilateral filtering, un particolare tipo di filtro di smoothing che ha la caratteristica di riuscire a preservare la nitidezza dell'immagine nei pressi degli edge, sfocando le aree rimanenti. Il risultato di tale operazione é mostrato in figura 12.

Figure 12: **Bilateral filtering** - immagine risultante dalla prima operazione di smoothing

Gaussian filtering A questo punto si riprende il normale workflow descritto nel paper [1]. Si introducono quindi altre operazioni di smoothing. La prima di queste é un filtro gaussiano che consente di rimuovere parte del rumore e la maggior parte delle linee non utili. Il risultato di quest'operazione é mostrato in figura 13.

Figure 13: **Gaussian filtering** - immagine risultante dalla seconda operazione di smoothing

Median filtering L'ultima operazione di smoothing é data dall'applicazione di un filtro mediano che consente di rimuovere la maggior parte del rumore lasciato dalle operazioni precedenti. Il risultato di tale operazione é mostrato in figura 14.

Figure 14: **Median filtering** - immagine risultante dalla terza operazione di smoothing

Negative image Dopo aver effettuato operazioni di normalizzazione e rimozione del rumore dall'immagine originale, si converte quest'ultima nella corrispondente negativa in modo da avere le regioni di interesse con un'intensitá maggiore [1]. Il risultato é mostrato in figura 15.

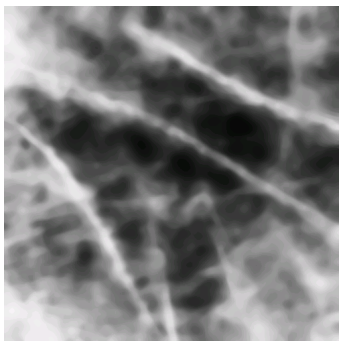

Figure 15: **Negative image**

4.2.2 Fase di estrazione

Una volta preparata l'immagine si passa alla fase di estrazione in cui si applicano operazioni di Top-Hat filter e contrast enhancement a partire dall'immagine negativa. A questa fase sono state apportate aggiunte sostanziali rispetto al flusso proposto nel paper di riferimento [1].

Top-Hat filter Si utilizza un filtro Top-Hat sull'immagine negativa per correggere l'illuminazione non omogenea. Il filtro Top-Hat calcola "l'apertura morfologica" di un immagine e poi sottrae l'immagine risultante all'immagine originale. Si tenga presente che nell'adattare l'operazione all'immagine 500x500 px si é utilizzato uno structuring element con forma ovale e con un raggio di 7 punti, il risultato di tale operazione puó essere consultato in figura 16.

Figure 16: **Top-Hat filter result**

Contrast enhancement In questa fase si é fatto uso nuovamente della tecnica di 'stretching dell'istogramma' che consente di aumentare il contrasto in modo da aumentare la visibilitá dell'immagine . La funzione opencv di cui si é fatto uso per raggiungere questo obiettivo é come in precedenza *'equalizeHist'* ed il risultato é mostrato in figura 17.

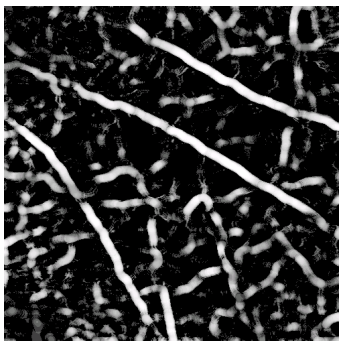

Figure 17: **Immagine risultato del Contrast enhancement**

Convert to binary image Questo step prevede la conversione dell'immagine da greyscale a binary. É necessario per poter effettuare il passo successivo, ovvero calcolare le componenti connesse presenti nell'immagine. La 'binarizzazione' dell'immagine é ottenuta applicando una funzione di thresholding, per cui tutti i valori di grigio superiori alla soglia di 150 (valore di intensitá compreso tra 0 e 255) sono trasformati in bianco, mentre quelli al di sotto della soglia in nero. Il risultato di tale operazione é visibile in figura 18.

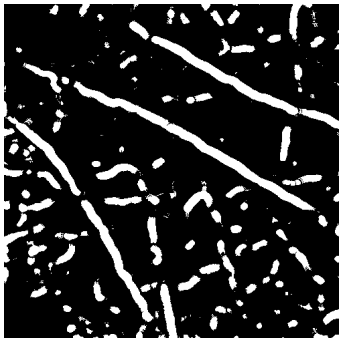

Figure 18: **Binary image**

Connected component labeling #1 Per rimuovere il rumore presente nell'immagine e le linee che non sono rilevanti ai fini di analisi si é utilizzato un algoritmo basato sulla ricerca delle componenti connesse in un immagine. Nell'utilizzo di tale algoritmo, messo a disposizione da opencv come 'connectedComponentsWithStats', le componenti connesse sono identificate tramite la verifica degli 8 pixel adiacenti a quello attualmente analizzato (8-neighborhood). Le componenti connesse risultanti che hanno un'area inferiore a 300 pixel sono scartate e trattate come rumore. Questo é l'ultimo step proposto nel paper di riferimento [1]. I risultati di tale operazione sono mostrati in figura 19.

Figure 19: **Componenti connesse #1** - rimozione del rumore mediante l'algoritmo di Connected component labeling

Enlarging Molte volte le linee principali, come conseguenza delle operazioni precedenti, possono risultare interrotte. Dopo un'attenta analisi, si é deciso di introdurre un'operazione morfologica di enlarging il cui scopo é proprio quello di collegare le linee eventualmente interrotte in modo erroneo ai passi precedenti. Un enlarging di 4 px é applicato in questa fase. Il risultato é apprezzabile in figura 20.

Thinning & Connected component labeling #2 L'operazione attuata al passo precedente ha il pregio di collegare linee interrotte, tuttavia, é ancora presente un eccessivo livello di rumore nell'immagine. Per ovviare a questo problema si effettua un'operazione morfologica di thinning e si riapplica l'algoritmo della ricerca delle componenti connesse con una soglia ridotta a 180 pixel. In tal modo é possibile ottenere un'immagine molto piú pulita in cui sono apprezzabili le linee principali ed il rumore é quasi assente. Il risultato di quest'ultimo step é mostrato in figura 21.

Figure 20: **Immagine risultato dell'enlarging**

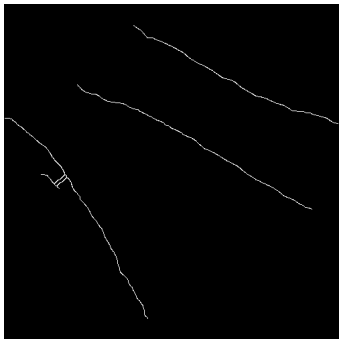

Figure 21: **Thinning and Connected component labeling #2 result**

5 Approcci risolutivi per la separazione delle linee

Una volta effettuata la fase di estrazione, ció che si ottiene é un insieme di linee, scelte tra le piú marcate rilevate dall'algoritmo all'interno della foto di input. A causa dell'elevata eterogeneitá dei palmi umani, delle condizioni di luce non completamente controllabili, del livello qualitativo dell'immagine non omogeneo, ma dipendente dal dispositivo in uso e dalla possibile presenza di rumore e/o piccoli errori di rilevamento, riconoscere tra le linee risultanti, quelle di effettivo interesse é un operazione non banale. Di seguito sono riassunti tutti gli approcci tentati per ottenere un esatto riconoscimento e raggruppamento dei segmenti ottenuti, nelle tre entitá di interesse (linea della vita, del cuore, della testa) per l'applicativo.
Come mostrato nella figura 22, é stata utilizzata la tecnica della trasformata di Hough per determinare quali punti rilevati dalle tecniche illustrate in precedenza, appartenessero effettivamente a linee. Tale strategia sostanzialmente si occupa di mappare un problema di "pattern detection" in un problema piú semplice di "peak detection". L'algoritmo procede col verificare la quantitá di punti che fanno parte di una specifica retta di equazione: $y = m * x + n$ e nel riconoscere i picchi formati dai punti che hanno i parametri corrispondenti a rette effettivamente esistenti nellímmagine originale. Tramite una soglia di threshold é possibile valutare correttamente i picchi rilevati, decidendo quali di essi rappresentano effettivamente parametri di punti appartenenti a linee.

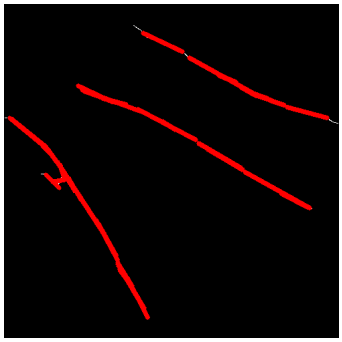

Figure 22: **Immagine risultato di Hough**

5.1 Clustering k-means

Una delle prime tecniche sviluppate é basata sul clustering k-means. Nello specifico é stato ipotizzato che un algoritmo di clustering fosse idoneo a risolvere il task della separazione delle 3 linee principali, tuttavia questa assunzione si é rivelata non veritiera e questa tecnica ha lasciato il posto ad altri branch di sviluppo piú promettenti in termini di risultati. É stato implementato un clustering k-means basato su 3 feature principali che produce in output 3 cluster, in teoria corrispondenti alle linee da individuare. Le feature su cui si basa l'algoritmo sono:

- Slope dei segmenti prodotti da Hough
- Posizione del punto medio dei segmenti prodotti da Hough rispetto all'asse x
- Posizione del punto medio dei segmenti prodotti da Hough rispetto all'asse y

Di seguito una descrizione non formale dell'algoritmo implementato:

input : k = 3, insieme di vettori $x_1...x_n$ a 3 dimensioni
Posizionare i centroidi $c_1...c_k$ *in posizioni random*;
repeat
  foreach *vettore* x_i **do**
    trova il centroide c_j piú vicino (distanza euclidea);
    assegna il punto x_i al cluster c_j;
  end
  foreach *cluster* $i = 1...k$ **do**
    calcolare il nuovo centroide c_j come media di tutti i vettori x_i assegnati al cluster al passo precedente.
  end
until *i centroidi non cambiano piú*;

La complessitá di quest'algoritmo é indicativamente O(#iterazioni * #clusters * #istances * #dimensions). I risultati ottenuti con questa tecnica peró non sono abbastanza soddisfacenti. Sebbene la procedura implementata funzioni correttamente essa non produce mai risultati perfetti. Quest'ultimo aspetto, che ha portato all'abbandono della stessa, é dovuto al fatto che un clustering di tipo k-means tende a produrre cluster di forma ovale che non sono conformi alle specifiche del problema. Inoltre il numero di feature individuate, sulla cui base fare clustering, risulta essere non sufficiente a discernere in modo netto le linee e quindi a produrre risultati soddisfacenti. Infatti, la linea del cuore e la linea della testa hanno per alcuni tratti lo stesso

slope e sono vicine in termini di location dei punti. Allo stesso modo la linea della vita e quella della testa sono talvolta congiunte e per brevi tratti hanno valori di slope simili. Nonostante tutto, lo sviluppo di quest'algoritmo é risultato, di fondamentale importanza per acquisire una conoscenza del dominio piú approfondita.

5.2 Clustering hierarchical single-link

Si é tentato di ricorrere, tra le varie strategie, anche ad un algoritmo di clustering di tipo gerarchico. Per clustering gerarchico si intende una particolare famiglia di algoritmi di clustering che mira a realizzare una gerarchia a livelli di cluster basandosi su una metrica di similaritá, stabilita a priori, tra gli elementi da agglomerare. La scelta é ricaduta sul single-link: un algoritmo di tipo agglomerativo, che seguendo una strategia di tipo bottom-up genera a partire da entitá distinte cluster sempre piú grandi, agglomerando ad ogni iterazione dell'algoritmo i due cluster con metrica di vicinanza maggiore. Tale algoritmo é stato scelto per una sua particolare caratteristica, spesso considerata un effetto collaterale, chiamata chaining. Per chaining si intende l'effetto per cui, essendo il single-link un algoritmo che usa un criterio di merge locale, non considera la forma e la similaritá generale dei cluster che vengono creati, ma si basa solo sulla massima similaritá locale producendo solitamente delle catene di elementi indefinitamente lunghe. Tale caratteristica, benché in altri ambiti possa essere negativa, risulta molto utile in quanto particolarmente vantaggiosa quando si tratta di accorpare segmenti di linee. La suddetta strategia é stata peró abbandonata a causa di alcune mancanze riscontrate in fase di testing.

- In primo luogo, una strategia di clustering di questo tipo, come tutte le strategie di clustering, risulta molto influenzabile dal rumore prodotto dall'elaborazione.
- Non é possibile sapere a priori se due linee tra le tre di interesse si intersecano in qualche punto, per cui, allo stesso modo, non é possibile stabilire a priori, se l'algoritmo debba fermarsi dopo aver generato due o tre cluster, in quanto linee che si intersecano vengono interpretate come una stessa linea.
- Un ultima difficoltá riscontrata, riguarda la complessitá computazionale dell'algoritmo, la quale oscilla tra O(n^3)(algoritmo di tipo naive che ad ogni n-esima iterazione, stabilisce i cluster da unire visitando una matrice di prossimitá di grandezza n*n) e O(n^2) (utilizzando la variante SLINK che fa a meno della matrice di prossimitá). Tali complessitá risultano particolarmente onerose, in caso di elevato numero di elementi da considerare, o scarso grado di performance del device in uso limitando notevolemnte la scalabilitá dell'intero applicativo.

.

5.3 Metodologia ad hoc

Dopo i vari approcci analizzati, tutti, come visto, con diverse problematiche, si é deciso di procedere con una metodologia ad hoc basata su tre funzioni, ciascuna delle quali in grado di individuare una specifica linea del palmo della mano.
Abbiamo giá visto quali siano le linee da individuare, pertanto possiamo passare a descrivere i relativi algoritmi. Da notare

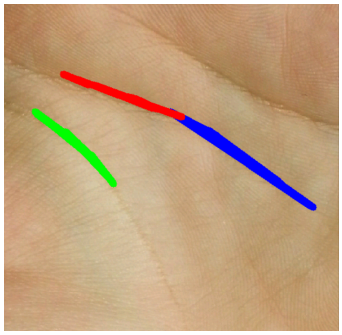

Figure 23: **Single link output** - risultato con errore

é che i seguenti algoritmi sono realizzati per identificare linee della mano sinistra dell'utente e si basano sui risultati della trasformata di Hough precedentemente effettuata.
Queste funzioni procedono nell'identificare le relative linee, cercando un punto di ancoraggio dal quale partire nell'analizzare i punti candidati a far parte della linea. Tale punto sará il primo punto trovato, in base alla regione di ricerca scelta, che come vedremo in seguito varia in base alla funzione, tra quelli selezionati dall'algoritmo di Hough. Una volta identificato il punto di ancoraggio, le funzioni procedono ispezionando un intorno del punto identificando cosí il prossimo punto della linea e procedendo finché ci sono punti candidati a far parte della linea. Ogni punto identificato e salvato verrá poi eliminato dall'immagine originale facilitando cosí le altre funzioni nell'individuare i relativi punti.
Le linee vengono restituite sottoforma di ArrayList in modo da poter compiere anche delle analisi di lunghezza e angolazione su di esse.
Procedendo dunque seguendo l'ordine dell'identificazione utilizzata nell'applicazione avremo :

- **leftLifeLineDetection**: questa funzione procede nell'identificare la linea della vita (la linea gialla), cercando un punto di ancoraggio nella parte inferiore dell'immagine e risalendo lunga la linea stessa.
- **leftHeartLineDetection**: questa funzione procede nell'identificare la linea del cuore (la linea verde), cercando il punto di ancoraggio a partire dall'angolo in alto a destra dell'immagine e procedendo verso il lato opposto dell'immagine.
- **leftHeadLineDetection**: questa funzione procede nell'identificare la linea della testa (la linea blu), partendo per la ricerca del punto di ancoraggio dall'angolo in alto a sinistra dell'immagine originale. Procederá poi nel ricercare i successivi punti scandendo l'immagine verso il lato opposto.

Tale metodologia é quindi risultata la piú idonea e quella in grado di offrire i migliori risultati a paritá di prestazioni rispetto le altre tecniche elencate.
Al fine di evitare errori causati dal rumore come falso positivo, si é deciso di ripetere le operazioni di identificazione per quelle linee che, una volta rilevate, risultano essere di dimensioni troppo ridotte per essere soddisfacenti. Il secondo tentativo, dato che le linee identificate e salvate vengono eliminate dall'immagine originale, garantisce l'esclusione dell'eventuale rumore permettendo quindi la scelta della linea corretta. Il risultato di tale operazione é visibile in figura 24.

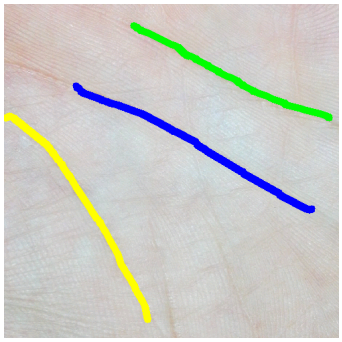

Figure 24: **Immagine risultato finale**

6 Sviluppi Futuri

Sebbene allo stato attuale il progetto offra buoni risultati per quello che é il suo obiettivo principale, ovvero l'identificazione delle linee del palmo della mano, si ritiene che esso possa crescere e migliorare sotto diversi aspetti. Di seguito una breve lista delle caratteristiche che verranno sviluppate o modificate per portare l'applicazione a compimento:

- **Miglioramento dell'interfaccia utente** rendendo l'applicazione piú user friendly e rendendo l'utente maggiormente coinvolto nell'esecuzione dell'analisi del palmo.
- **Aggiunta della lettura per la mano destra** in quanto, come é noto, secondo la tradizione cinese le linee delle due mani hanno interpretazioni diverse, quindi si rende necessario implementare la lettura della mano destra per realizzare un'applicazione completa.
- **Miglioramento del sistema di predizione** migliorando l'accuratezza delle predizioni ed i fattori caratterizzanti da analizzare. Come anticipato il 'database' attuale é molto limitato perché ritenuto non di effettivo interesse per questa fase progettuale.
- **Miglioramento delle performance** utilizzando strutture dati differenti che possano dare una maggior velocitá di analisi all'applicazione. Un semplice esempio potrebbe essere l'utilizzo nativo di OpenCV al posto della libreria android messa a disposizione, la quale risulta male ottimizzata.
- **Studio di ulteriori metodologie** al fine di ottenere un eventuale miglioramento per l'identificazione delle linee del palmo della mano.

7 Conclusioni

In conclusione é stato realizzato un software mobile che, attraverso alcune metodologie che combinano diversi algoritmi di visione computazionale, é in grado di ricavare con buone performance le principali linee del palmo della mano.
I test condotti evidenziano come da un punto di vista qualitativo i risultati siano completamente indipendenti dalla tipologia di smartphone e dalla relativa qualitá della fotocamera. Le performance, da un punto di vista prestazionale, sono invece minimamente affette garantendo tempi di risposta rapidi a prescindere dalla qualitá dello smartphone utilizzato.
Questa applicazione, come si é visto, si avvale di una metodologia di analisi dell'immagine completamente slegata dai canonici metodi di rilevazione dei bordi ed inoltre, tale metodologia, é stata modificata andando a rispondere in modo specifico ad esigenze implementative e progettuali.
Nonostante le difficoltá incontrate lo sviluppo di quest'applicazione ci ha permesso di toccare con mano, anche se in minima parte, il complesso mondo della visione computazionale facendoci comprendere quanto possa essere complicato analizzare ed estrapolare informazioni da scenari del mondo reale.

References

[1] J. A. Zia-uddin, Zahoor Jan and A. Abbasi, "A novel technique for principal lines extraction in palmprint using morphological top-hat filtering," *World Applied Sciences Journal*, vol. 31, July 2014.