“Algoritmo di Posture Recognition per App Android”

Corso di “Tecniche Audiovisive”, Prof. G.Orlandi

Alessandro Moschella, Matr.1375639

Indice

Capitolo 1

Introduzione

Gli avanzamenti tecnologici degli ultimi decenni hanno portato a una larga diffusione di smartphone più o meno performanti, ad un costo generalmente molto abordabile, il che ha permesso di aprire scenari, fino a qualche tempo fa impensabili, soprattutto perchè usufruibili dalla massa.

In parallelo, un campo in rapida espansione è la cosiddetta Human-Computer Interaction (HCI), che si occupa dell'interazione tra utenti e computer, per sviluppare sistemi interattivi affidabili, e che facilitino le attività umane. La sua evoluzione è quindi indirizzata verso lo sviluppo di un tipo di comunicazione macchina-utente più intuitivo, naturale e adatto ai comuni comportamenti umani.

Una delle modalità di comunicazione che da tempo risulta essere oggetto di sfide nel settore dell'HCI è il movimento delle mani ("Hand Gesture"), in quanto questa risulta essere il tipo di comunicazione non verbale più utilizzata dalle persone, per imporre comandi, indicare azioni o per la lingua dei segni, utilizzata dalle persone sorde.

I gesti in generale possono essere statici, nel qual caso vengono anche chiamati "posture", o dinamici cioè formati da opportune combinazioni di posture.

Esempio di utilizzo delle posture è la dattilologia, o "alfabeto manuale", che è la rappresentazione manuale delle lettere utilizzate nella scrittura; essa sfrutta le dita della mano per formare combinazioni che hanno la stessa funzione dei caratteri della scrittura.

Attraverso l'alfabeto manuale è possibile segnare ciascuna lettera utilizzando esclusivamente le posizioni e i movimenti delle dita della mano, per poter comunicare nomi propri (geografici o di persona), nomi non conosciuti e parole di lingue estere.

In questo lavoro verrà presentata una implementazione del riconoscimento delle posture, che potrà essere un ottima base per lo sviluppo di applicazioni sempre più consistenti, che si estendano al riconoscimento dei gesti e quindi anche di frasi e/o comandi.

La sfida principale è stata quella di integrare questo sistema di riconoscimento di posture in una Applicazione per smartphone provvisto di S.O. Android.

Obiettivo finale è quello di creare un prodotto finale efficiente, pur essendo integrato in dispositivi con processori di potenza limitata, e video camere a risoluzione non elevata.

Capitolo 2

Android

Android è un sistema operativo per dispositivi mobili sviluppato da Google Inc., in particolare è un insieme di strumenti e librerie per la composizione di applicazioni mobili.

Il sistema operativo è basato sul Kernel Linux così da creare applicazioni in grado di rispondere in modo del tutto immediato all'utente, e può essere considerato una distribuzione GNU/Linux per sistemi embedded.

Essendo stato principalmente progettato per smartphone e tablet, che hanno a disposizione un hardware limitato, si è scelto di utilizzare una macchina virtuale diversa da quella Sun, nonostante le applicazioni vengano comunque scritte in linguaggio di programmazione Java.

La macchina virtuale appositamente creata e ottimizzata per l'ambiete a disposizione, la Dalvik Virtual Machine (DVM) è in grado di eseguire il codice contenuto all'interno di un file di estensione .dex ottenuto a partire dal byte-code Java.

In sostanza Android è caratterizzato dall'essere open, dato che:

- utilizza tecnologie Open Source, quali il Kernel Linux così come le librerie e le API che non solo sono sfruttate per la sua realizzazione, ma anche per la creazione delle Applicazioni;
- il codice Android è completamente reperibile online, così che ognuno possa contribuire al suo miglioramento e dare il proprio contributo;
- la licenza scelta dalla Opend Handset Alliance è la Open Source Apache License 2.0, la quale permette ai vendors di costruire su Android le proprie estensioni anche proprietarie, senza legami che potrebbero limitarne l'uso; ciò ha un importanza rilevante dal momento che non vi è alcuna royality da pagare da parte dei produttori che vogliono adottare Android sui propri dispositivi.

2.1 Computer Vision

L'insieme dei processi che mirano a creare un modello approssimato del mondo reale (3D) partendo da immagini bidimensionali (2D) è detto Visione Artificiale.

La capacità di visione, intesa come pura acquisizione di immagini, è attualmente considerabile come un problema risolto, dato che le capacità visive dei sistemi ottici e i relativi sensori, hanno superato le possibilità dell'occhio umano per quanto riguarda sensibilità, velocità e risoluzione.

Lo step successivo della computer vision, è l'interpretazione e l'utilizzazione delle informazioni acquisite.

Questo risulta essere a tutt'oggi un problema complesso per un sistema automatico, in quanto la conversione di una immagine in informazioni oggettive non può avvenire in maniera banale come per il cervello umano.

Un sistema di visione artificiale è dato dall'integrazione di componenti ottiche, elettroniche e meccaniche che permettono di acquisire, registrare ed elaborare immagini sia all'interno che all'esterno dello spettro della luce visibile.

E' proprio attraverso l'elaborazione delle immagini stesse che è possibile poi focalizzarsi sull'estrazione di determinate caratteristiche quali la ricerca di determinati oggetti e parti del corpo, la ricostruzione della scena, etc.

2.2 Libreria OpenCV

Nell'ottica della visione artificiale è necessario avere una base comune di strumenti analitici su cui operare.

Il primo di questi è una libreria che raccoglie da una parte le funzionalità degli algoritmi più usati, e dall'altra una serie di formati di rappresentazione dei dati secondo standard aperti e condivisi. Le librerie OpenCV (Open Computer Vision) nascono appunto per questo scopo.

Il progetto è nato da un gruppo di ricerca sponsorizzato da Intel, e da questo ha ereditato alcuni aspetti dell'Intel Image Processing Librery (IPL), avvelendosi poi di svariati contributi sia da ricercatori del MIT e docenti della Berkeley University.

Uno dei punti di forza della libreria è l'utilizzo della licenza in stile BSD (Berkeley Software Distribution, prima licenza LINUX), famiglia di licenze software permissive delle quali molte vengono considerate Open Source, che a grandi linee premette la libera ridistribuzion sia in forma sorgente che binaria, anche all'interno di prodotti commerciali, a condizione di mantenere le note di copuright e di non usare il nome di Intel a scopo promozionale di prodotti derivati.

Con il termine "libreria grafica" si identificano in genere almeno tre famiglie di librerie con scopi differenti:

- i toolkit, librerie primitive per la creazione di oggetti grafici di interfaccia;
- librerie di rendering e multimedia, come DirectX e OpenGL, orientate alla massima performance nella creazione di effetti poligonali o vettoriali, il cui

utilizzo più comune è teso all'ottenimento di elevate prestazioni grafiche sfruttate nei videogame o nelle applicazioni multimediali;

- librerie di gestione hardware grafico come digitalizzazioni e frame graber.

Le librerie OpenCV, anche se includono alcune funzionalità tipiche di ciascuna delle famiglie citate, non fanno parte di nessuna di essa in particolare.

L'utilizzo primario è infatti quello collegato alla visione artificiale, il cui obiettivo principale è quello di estrarre dati significativi dalle immagini, tali da essere trattabili automaticamente.

Le funzioni presenti la libreria coprono le più svariate esigenze: trattamento di immagini, funzioni matematiche ottimizzate ed un completo pacchetto di algebra matriciale.

Tutte le funzioni della libreia sono accomunate dal prefisso "cv", mentre i nomi delle classi usate hanno prefisso "Cv", ad eccezione della IplImage, ereditata dalla libreria Ipl.

La libreria contiene inoltre diverse strutture dati come CvMat, ovvero matrici numeriche, IplImage per il buffer immagine, CvMemStorage per i buffer dinamici, CvSeq per le liste dinamiche, grafi e alberi con le relative funzione di gestione.

Sono inoltre presenti funzioni per il disegno diretto di linee, ellissi, poligonali, testo etc.

Nel caso in cui vi fosse la necessità di ricorrere a librerie deidicate, specifiche per la gestione dell'hardware adottato, le funzioni della libreria permettono un facile scamnio di dati con oggetti provenienti da altre librerie.

Tra le funzioni più avanzate,openCV mette a disposizione molti degli algoritmi più raffinati oggi reperibili.

Uno dei problemi più comuni affrontati nel trattamento delle immagini riguarda la delimitazione di oggetti, o in generale la loro identificazione a scopo di misura, come ad esempio nel riconoscimento dei caratteri (OCR) oppure nella trasformazione di immagini da raster a vettoriali per i quali è presente l'algoritmo di Canny per l'identificazione degli spigoli dell'immagine.

Questo algoritmo è attualmente considerato uno dei migliori disponibili allo scopo, ma oltre ad esso troviamo i filtri di Sober e di Laplace così come una completa serie di funzioni di classificazione geometrica dei contorni.

Le funzioni di classificazione vengono anche utilizate per un altro dei comuni problemi di analisi, ovvero la blob analisi, cioè l'identificazione di aree omogennee, procedimento molto utilizzato nell'analisi di immagini ottenute al di fuori del campo visibile.

E' inoltre disponibile la trasformata di Hough per l'identificazione di pattern e che permette tra le altre cose, l'identificazione di linee rette e di pattern regolari all'interno di un'immagine.

Per quanto riguarda invece l'analisi di immagini in movimento (motion detection, object tracking,etc) sono presenti applicazioni degli algoritmi di Lucas e Kanade, e di Horn e Schunk, utili per il calcolo dei flussi ottici per la rilevazione del movimento e per il block marching (rilevazione di elementi uguali

in posizioni diverse tra immagini successive); è presente inoltre un classificatore basato sul metodo di Haar ed ottimizzato per il riconoscimento di volti umani e utilizzabile anche per il riconoscimento di oggetti dopo una adeguata istruzione.

Esiste infine un'implementazione del filtro di Kalman e dell'algortirmo di condensation, utilizzari nella riduzione del rumore e nei problemi di previsione.

Capitolo 3

Posture Recognition

L'approccio utilizzato in questo lavoro per lo sviluppo del sistema di Posture Recognition ha dato maggiore importanza al peso dell'algoritmo, alla sua velocità unita alla sua efficienza, dal momento che, come detto in precendeza, i dispositivi per cui sono pensati possono essere poco performanti.

Per sviluppare l'algoritmo è stato preso in considerazione l'alfabeto manuale, mostrato in figura3.1, del quale si sono considerati solo le lettere dalla A alla F, visto che i rispettivi gesti sono statici.

Le restanti lettere dell'alfabeto invece non sono state considerate, perchè alcune di esse hanno come corrispondenti, dei gesti dinamici (ovvero composti da una sequenza di posture), e il loro studio esula dagli scopi del progetto.

La modalità operativa che si è scelta di adoperare, si esplica nel trattamento del frame corrente, senza la necessità da parte dell'utente di scattare o salvare fotografie del gesto da rilevare.

In questo modo è stato possibile creare una App che in real time scansiona la postura della mano e ne restituisce la corrispondente lettera in sovraimpressione.

Interfaccia grafica e funzionamento App Android Innanzitutto per quanto riguarda l'aspetto visivo e funzionale che si è scelto di dare alla App, possiamo dire che questo è stato concepito per complicare il meno possibile il lavoro all'utente che utilizzerà l'App, tenendo ovviamente presenti i vincoli da rispettare per il corretto funzionamento del Posture Recognition.

In particolare la App si presenterà con i seguenti layout:

- Innanzitutto sarà necessario settare uno sfondo, per mezzo della fotocamera principale dello smartphone, e che preferibilmente abbia colore uniforme nero/scuro, fig.3.2;
- Una volta settato lo sfondo, e mantenendo la fotocamera focalizzata su di esso, sarà automaticamente avviato l'algoritmo di riconoscimento delle posture. A questo punto, una volta che l'utente avrà sovrapposto una certa configurazione delle dita della mano sullo sfondo stesso, verrà visualizzata

Figura 3.1: Alfabeto manuale (fonte web).

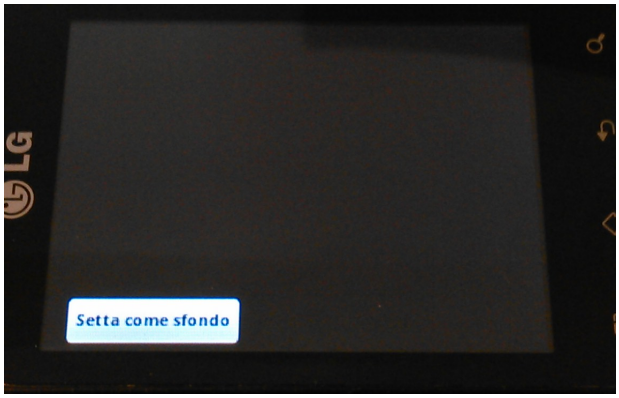

Figura 3.2: Interfaccia dell'App Android per il settaggio dello sfondo.

in real time la lettera corrispondente al gesto, in caso di matching (vedi esempio fig.3.3).

- E' stata inoltre aggiunta la possibilità di archiviare un nuovo gesto, associandolo alla corrispondente lettera, così da aggiornare l'archivio delle posture, come visibile sempre in fig.3.3.

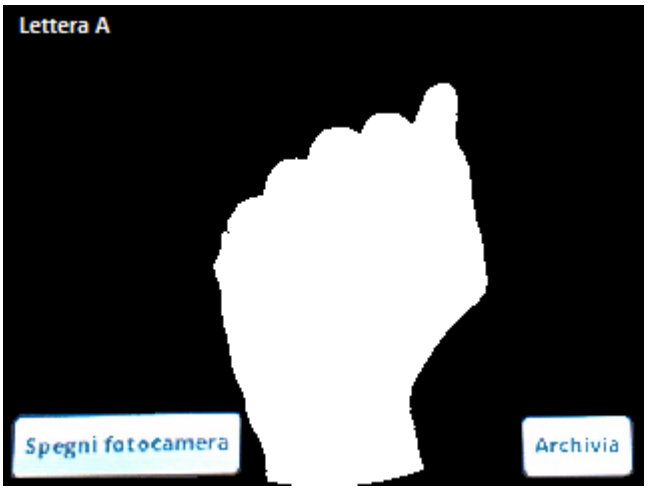

Figura 3.3: Interfaccia dell'App Android durante la fase di matching.

E' da sottolineare che all'avvio della fotocamera (all'interno dell'App), la successione di frame sarà visualizzata in scala di grigi, mentre dopo aver settato il background le immagini visualizzate in real-time saranno binarie, ovvero: tutto ciò che è uguale o simile allo sfondo impostato, è visualizzato nero, mentre tutto ciò che è diverso dal background è visualizzato di bianco.

3.1 Strategia utilizzata: algoritmo

Il sistema di riconoscimento delle posture è stato strutturato in modo da seguire una serie di passi, come descritto in figura 3.4.

Dalla figura 3.4 infatti si evidenziano i seguenti step:

1. All'avvio dell'App verrà richiesto di settare uno sfondo, attraverso l'uso della fotocamera principale dello smartphone, e che non dovrà mutare durante l'utilizzo dell'App stessa. Questo background dovrà essere, come già accennato, preferibilmente scuro/nero, non riflettente e possibilmente collocato in un ambient mediamente illuminato, per favorire il rilevamento successivo della mano.

2. Una volta settato lo sfondo si avvierà automaticamente la procedura dello Skin Detection per l'identificazione della mano, vista come "oggetto" esterno che entra nel campo visivo della fotocamera.

3. A questo punto sono possibili due opzioni:

   (a) nel caso in cui siano state già archiviate delle lettere (sottoforma di gesti), verrà automaticamente avviata la procedura per il Posture Recognition, che ovviamente farà capo al database di posture;

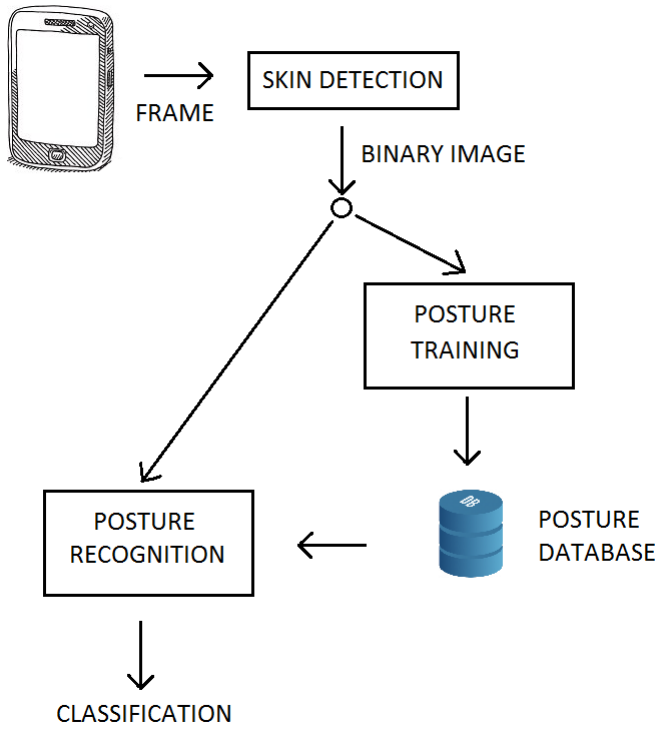

Figura 3.4: Schema blocchi del sistema

il risultato di questo matching sarà visualizzato su schermo in tempo reale.

(b) nel caso in cui si voglia archiviare l'attuale gesto ed associarlo ad una determinata lettera basterà utilizzare il bottone presente in basso a destra e seguire le istruzioni per il salvataggio automatico.

3.2 Adattamento ad Android

Per poter adattare la struttura dell'algoritmo ad un prodotto munito di Sistema Operativo Android, cercando di alleggerire il più possibile il codice prodotto, si sono studiate nei dettagli le potenzialità che lo stesso sistema offre.

3.2.1 OpenCV Manager

All'interno della nostra applicazione Android è possibile utilizzare la libreria OpenCV, senza l'utilizzo del codice nativo, ma appoggiandoci a quello che viene definito "OpenCV Manager" che è il pacchetto fornito dal Market di Android,

avente il compito di mettere a disposizione delle applicazioni la miglior versione delle OpenCV disponibile al momento.

OpenCV Manager, una volta installato sul dispositivo in cui sarà testata/utilizzata l'App, è quindi un servizio per Android mirato alla gestione delle librerie OpenCV, che permette di condividere le librerie tra le varie applicazioni presenti sullo stesso dispositivo.

Inoltre, sempre nell'ottica "risparmio", OpenCV Manager offre i seguenti vantaggi:

- minor utilizzo di memoria, dal momento che tutte le applicazioni utilizzano lo stesso pacchetto binario e non hanno librerie al loro interno;
- ottimizzazione dell'hardware per tutte le piattaforme utilizzate;
- tutti i pacchetti OpenCV vengono pubblicati sul Market;
- aggiornamenti regolari e bug fix.

3.2.2 Inizializzazione Asincrona (Async initialization)

In fase di programmazione, per collegare la nostra App alle librerie di OpernCV abbiamo due possibilità:

1. modalità statica: tutti i file binari di OpenCV sono collegati e inseriti nel pacchetto della nostra applicazione. Questo metodo è utilizzato principalmente per scopi di sviluppo, ma sconsigliato per realizzare codice destinato alla "produzione", poichè non sfrutta le potenzialità offerte da OpenCV Manager in termini di leggerezza e velocità.
2. modalità asincrona (async): è quella consigliata, dal momento che utilizza il serivizio automatico di Android per scaricare e gestire la libreria OpenCV, ovvero utilizza OpenCV Manager per accedere alle librerie OpenCV installate esternamente all'App. L'applicazione creata con questa modalità funziona quindi con il gestore OpenCV in modo ascincrono, e nel codice java dell'App sarà presente la chiamata di callback "OnManagerConnected", che verrà effettuata in un thread UI quando termina l'inizializzazione. Notare che non è consentito utilizzare le chiamate OpenCV o caricare librerie native indipendenti, prima di aver effettuato la chiamata di callback.

3.3 Skin detection

Torniamo adesso ad occuparci dello schema a blocchi di figura 3.4, ed in particolare della skin detection.

Il rilevamento della pelle, in generale, può essere fatto in due modalità:

- background subtraction

- skin color detection

Quella da noi scelta è stata la background subtraction, poichè la seconda, è notevolmente influenzata dall'illuminazione, dalle ombre etc, ragion per cui i parametri da settare avrebbero dovuto essere compresi in range più o meno larghi che avrebbero potuto facilmente invalidare la precisione della detection, ancor di più avendo a che fare con smartphone non performanti.

Con la background subtraction invece si può ovviare a questi problemi, e anche evitare la scelta dello spazio dei colori per rilevare la pelle (RGB, HSI etc).

Nello specifico, una volta che è stato settato lo sfondo, il processo di acquisizione dei frame continua e per ciascuno di essi, preventivamente convertito in scala di grigi, viene fatta la differenza (in valore assoluto) pixel a pixel con il background.

Per poter discriminare meglio l'eventuale mano presente, viene a questo punto accentuato il contrasto della "immagine differenza", trasformando questa da scala di grigi a una in bianco&nero.

La soglia di luminosità, il cui range va da 0 a 255, è stata settata a 24, a valle di test empirici.

Per questa fase è stato necessario utilizzare due funzioni delle libreria di OpenCV, in particolare:

absdiff della libreria Core, per la differenza fra background e frame.

threshold della libreria Imgproc, che serve per ottenere un'immagine bi-livello a partire da una in scala di grigi.

3.4 Training di Archiviazione

La fase di Posture Training è fondamentale ai fini del Posture Recognition, poichè in questa vengono archiviati i gesti relativi alle lettere desiderate.

E' da sottolineare che per motivi pratici, avendo a disposizione uno smartphone con memoria interna minima, questa archiviazione è stata effettuata su memoria esterna SD card.

Questa fase essendo compiuta a valle dello Skin Detection, sfrutta il file precendentemente descritto, ovvero una immagine-frame in cui lo sfondo è stato completamente reso nero e la mano interamente bianca. A questo punto i passi principali in cui si articola questa fase sono i seguenti:

1. all'immagine-frame bianca&nera ottenuta dalla Skin-detection, viene applicata una maschera circolare, mostrata in figura 3.5, allo scopo di non considerare completamente il braccio, potenzialmente presente nell'inquadratura, e concentrarci solo sulla zona centrale-superiore dell'immagine stessa, dove è più probabile venga posizionata la mano.

2. attraverso la pressione del bottone "Archivia", l'utente viene invitato a digitare la lettera da associare al gesto rappresentato e il salvataggio in memoria esterna avviene automaticamente.

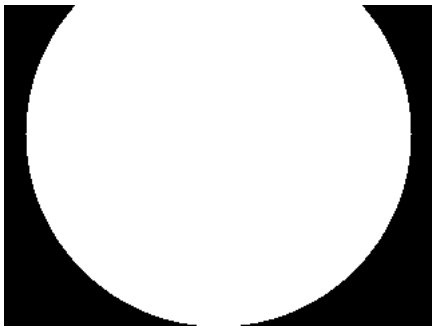

Figura 3.5: Maschera circolare

Questa procedura operativa permette di memorizzare immagini in archivio, da 320x240 pixel con un peso oscillante tra 1 e 2KB, contro i 30KB medi, delle immagini in scala di grigi con background subtraction effettuato.

3.5 Matching - Posture Recognition

Questa fase dell'algoritmo sfrutterà entrambi gli step precendenti e si svilupperà secondo una metodologia di lavoro scelta "mediando" i vari estrattori di caratteristiche utilizzati in diversi lavori accademici [Post1],[Post2].

Per riuscire ad effettuare un matching veloce tenendo conto che le potenzialità degli smartphone non sono sempre le migliori, si è deciso di non creare ex-novo algoritmi estrattori/classificatori, ma di sfruttare le funzioni messe a disposizione dalle librerie di OpenCV stesse, che sicuramente sono ottimizzate in questa ottica.

Innanzitutto, quello che viene effettuato dall'App è caricare immediatamente, all'avvio della stessa, tutte le immagini presenti in archivio.

Contorni In ciascuna di queste immagini caricate viene effettuata la ricerca dei contorni presenti, e di questi, solo il contorno con area più grande viene considerato di interesse, perchè sicuramente relativo alla sagoma della mano.

Le funzioni utilizzate per questo passo sono:

findContours della libreria Imgproc, applica l'algoritmo Suzuki85 [Suzuki85], e opportunamente settata restuisce tutti i contorni esterni presenti in una immagine binaria, senza approssimazioni, ovvero memorizzando tutti i punti per ciascuno di essi.

contourArea della libreria Imgproc, calcola l'area di un contorno utilizzando la formula di Green.

Una volta scansionato tutto l'archivio di immagini, ed estratto un contorno per ciascuno di esse, li memorizzo run-time uno ad uno in appositi array, associando

ad ogni contorno la relativa lettera, sempre per motivi di velocità del matching successivo.

Matching A questo punto, posto che l'utente ad ogni avvio della fotocamera dovrà settare uno sfondo su cui successivamente sovrapporre la mano in determinate posture, i successivi frame acquisiti saranno confrontanti con tutti questi contorni recuperati. E' evidente che per ciascuno di questi nuovi frame acquisiti run-time verrà compiuta la stessa procedura di cui sopra, per la ricerca del contorno con area maggiore, per poi poterlo confrontare con quelli recuperati dall'archivio. Questo confronto fra contorni, si è pensato di farlo cercando di gestire il più possibile gli inevitabili movimenti, cambiamenti di scala, rotazioni e traslazioni della mano da rilevare, per cui si è utilizzata una funzione che consideri tutti queste variabilità:

matchShapes della libreria ImgProc, compara due contorni utilizzando i momenti di Hu di un'immagine, che appunto godono delle proprietà di:

- invarianza alla traslazione;
- invarianza ai cambiamenti di scala;
- invarianza alla rotazione.

  Ricordiamo infatti che, in generale, nella computer vision il momento di un'immagine è una particolare media dell'intensità dei pixel componenti l'immagine, e quindi i momenti di una immagine indicano un insieme di caratteristiche dell'immagine stessa; nel loro insieme quindi i momenti riescono a determinare univocamente l'immagine, e addirittura un piccolo sottinsieme di essi è già significativo per descrivere l'immagine stessa. Oltre all'esistenza dei momenti spaziali e dei momenti centrali, Hu ha dimostrato nel 1962 [Hu] che esistono 7 momenti, che godono delle tre proprietà suddette, e che possono essere costruiti a partire dai momenti centrali normalizzati fino al terzo ordine.

A valle dei confronti fra il frame considerato e tutti i contorni archiviati, viene selezionato quello più simile (se esistente), e quindi visualizzata in sovraimpressione la lettera corrispondente al gesto.

E' da sottolineare che la soglia di similitudine è stata settata opportunamente a 0.07, a valle di test empirici, in modo da equilibrare la probabilità di corretta di rivelazione e quella dei falsi allarmi.

Capitolo 4

Risultati e sviluppi futuri

Seguendo l'algoritmo di cui sopra (Cap.3), si è prodotta una Applicazione Android sufficientemente performante, che permette di unire una moderata velocità di elaborazione ad un soddisfacente livello di matching.

Come accennato infatti nel capitolo 3, i test sono stati effettuati su un sottoinsieme dell'alfabeto manuale, ovvero dalla A alla F, sia per motivi di tempo, che per motivi pratici, poichè alcune delle lettere successive sono formate da una sequenza di posture, e ciò implica l'estensione dell'algoritmo prodotto, alla gestione di sequenze di frame.

4.1 Risultati

Per effettuare i test sulle performance dell'App, si è operata una fase di training in cui sono state archiviate un certo numero di posture per ciascuna lettera, in modo da poter gestire eventuali variazioni delle dita.

A valle di ciò i risultati ottenuti dai matching delle lettere dalla A alla F sono mostrati, a titolo di esempio, nelle figure: 4.1, 4.2, 4.3, 4.4, 4.5, 4.6.

Come si vede, attraverso il settaggio delle soglie di luminosità e di similitudine, i matching pervenuti sono soddisfacenti, e la velocità del match stesso è abbastanza immediato.

4.2 Conclusioni

L'algoritmo di Posture Recognition, descritto in questo lavoro, ha complessivamente fornito dei risultati soddisfacenti tenendo conto dell'hardware non altamente performante di cui si disponeva, sia in termini di fotocamera che di processore.

Questo quindi va sicuramente testato con dispositivi che offrono maggiori prestazioni.

L'algoritmo inoltre potrà essere facilmente esteso anche al rilevamento delle "lettere dinamiche" (ovvero le lettere rappresentate da gesti composti da sequen-

ze di posture), semplicemente considerando anche gruppi di frame, anzichè solo quello corrente, in modo da poter sviluppare poi dei veri e propri riconoscitori di parole.

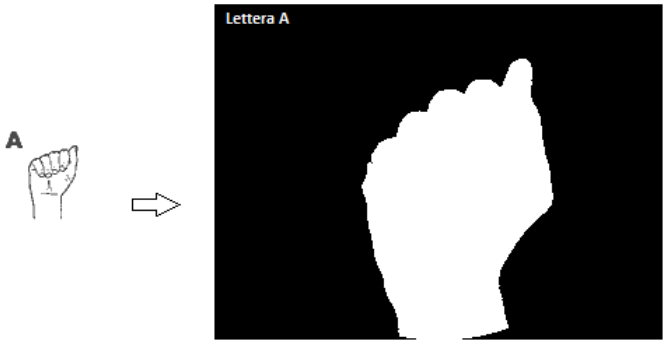

Figura 4.1: Matching lettera A

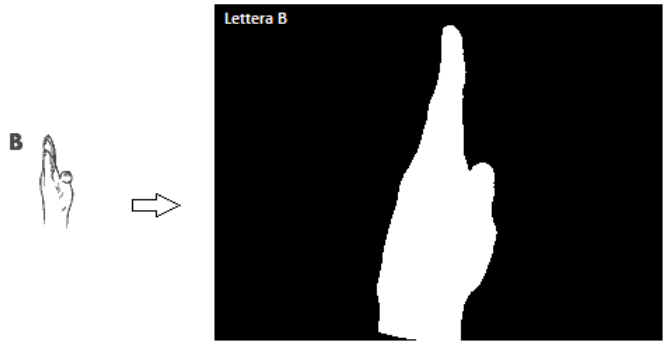

Figura 4.2: Matching lettera B

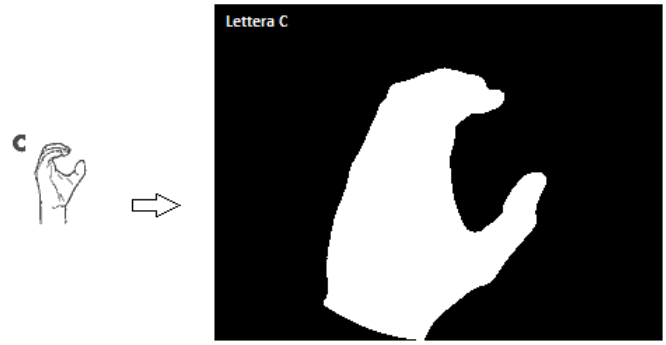

Figura 4.3: Matching lettera C

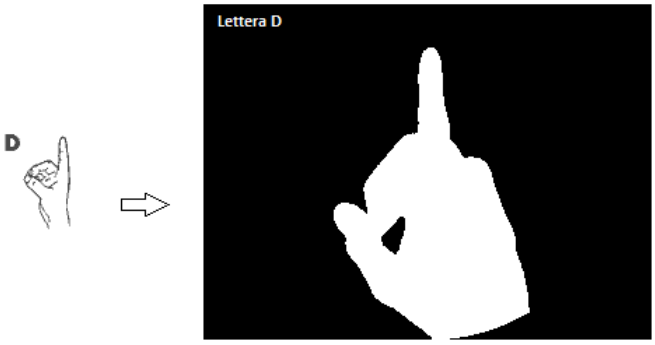

Figura 4.4: Matching lettera D

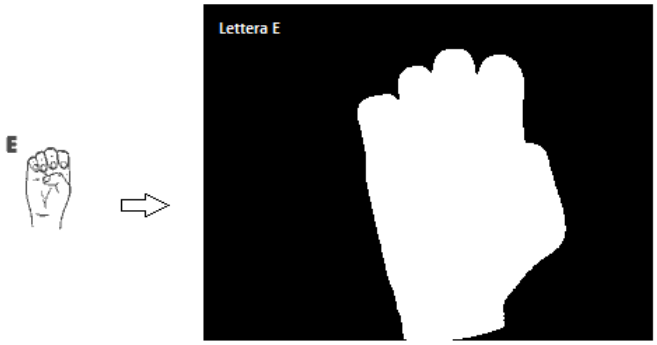

Figura 4.5: Matching lettera E

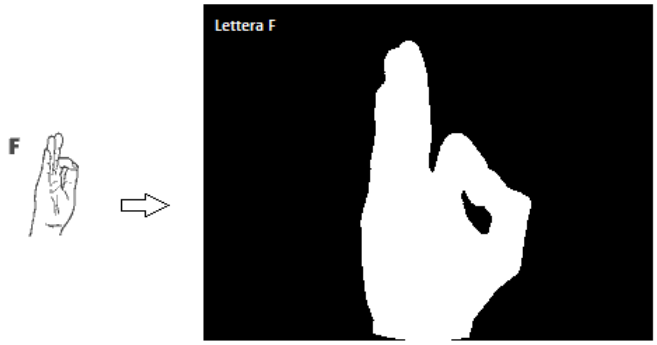

Figura 4.6: Matching lettera F

Bibliografia

[Post1] L. Tarrataca, A.C. Santos, J.M.P. Cardoso, The Current Feasibility of Gesture Recognition for a Smartphone using J2ME. Proceedings of the 2009 ACM symposium on Applied Computing, Pages 1642-1649.

[Post2] R.C.B. Madeo, S.M.Peres, D.B.Dias, C.Boscarioli, Gesture Recognition for Fingerspelling Applications: An Approach Based on Sign Language Cheremes, Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility (2010), Pages 261-262.

[Suzuki85] Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)

[Hu] M. Hu, Visual pattern recognition by moment invariants. IRE Trans. Information Theory, Vol. IT-8, Num. 2, 1962.