UNIVERSITÀ DEGLI STUDI DI PERUGIA

Dipartimento di Matematica e Informatica

Relazione

Tagliatore di teste

Applicazione Android

Lorenzo Franco Ranucci & Anna Pagoni

1 - Sommario

Android-Tagliatore di teste Scenario/Obiettivo: un'app basata su OpenCV riconosce la faccia di un individuo e la “taglia” ricostruendo lo sfondo in modo adattivo. L’utente può interagire con la propria “testa” spostandola nello schermo. Il tutto avviene in tempo reale attraverso la “preview” della fotocamera del dispositivo Android.

2 - Come funziona l'app

L'applicazione Tagliatore di Teste si compone di una sola schermata in cui l'utente visualizza il flusso di immagini catturate dalla fotocamera. Prima di essere mostrato, il flusso di immagini viene modificato in modo tale che, se viene rilevato un volto umano, esso viene “tagliato” dal resto del corpo e l'utente può in tempo reale spostarlo in qualsiasi punto dello schermo.
L'utente deve posizionare il proprio dispositivo Android in posizione salda e immutata per tutta la computazione. Una volta posizionato il dispositivo, l'utente deve registrare lo sfondo utilizzando il bottone con etichetta “background”. Da questo momento l'applicazione è in grado di riconoscere volti (uno alla volta) e spostarli sullo schermo. Tramite la seekbar con etichetta “Threshold” l'utente può regolare un valore di soglia per migliorare la qualità del taglio.
Tramite la seekbar “Quality” l'utente può decidere la qualità dell'immagine e quindi rendere più o meno performante la computazione.
Toccando il bottone che raffigura una fotocamera è possibile salvare in memoria il frame corrente.
In fine è presente un bottone per eseguire uno switch tra la fotocamera frontale e posteriore del telefono.

3 - Ambiente di sviluppo

L'applicazione è stata sviluppata per dispositivi mobile con sistema operativo Android, muniti di fotocamera.
Per lo sviluppo dell'applicazione è stato utilizzato il software Android Studio.

Sono state sfruttate le librerie OpenCV, libreria open source per la computer vision, campo che si occupa dei metodi per acquisire, processare e analizzare immagini.
Pur essendo scritte in linguaggio C++, l'organizzazione OpenCV offre un SDK per Android che include delle classi Java che permettono di interfacciarsi con i metodi nativi.

4 - Descrizione del problema e delle ipotesi adottate

Dovendo simulare il taglio di una testa, i problemi riscontrati sono stati molteplici.
In prima istanza si è resa necessaria una tecnica per il rilevamento del volto. Questo compito viene eseguito grazie agli strumenti già esistenti, testati ed efficienti del SDK OpenCV.

In secondo luogo, una volta ottenuto il riquadro contenente il volto è stato necessario ridimensionarlo per fare in modo di contenere l'intera testa.

Tramite una tecnica che rientra nel campo della Background Subtraction (o Foreground Detection) si è svolto il compito di “ritagliare” il contorno della testa.

Background Subtraction è un termine che racchiude diverse tecniche molto utilizzate nel campo del trattamento digitale delle immagini, è usata in diverse applicazioni per rilevare oggetti in movimento nella scena video quando la camera è statica, come per esempio in sistemi di video sorveglianza, optical motion capture e multimedia.
La difficoltà che queste tecniche spesso riscontrano è che il campo di utilizzo prevede solitamente riprese video dalla lunga durata in cui la determinazione dello sfondo è resa difficoltosa da cambiamenti di luce e introduzione di oggetti statici che entrano a far parte dello sfondo in un momento posteriore al tempo zero di inizio ripresa video.
La Background Subtraction presenta i seguenti passi:

- Modellazione e inizializzazione del background
- Background maintenance
- Foreground Detection

Essendo Tagliatore di teste, un'app pensata per lavorare sulle riprese video per breve tempo, si è scelto di inizializzare lo sfondo con un immagine priva di oggetti in movimento e impostata direttamente dall'utente.

Per quanto riguarda la background maintenance, essa non risulta come un passaggio indispensabile per la nostra applicazione, infatti l'utilizzo di un'app per dispositivi mobili in norma non ha una durata prolungata e soprattutto non prevede la presenza di vari soggetti in movimento. Per questo motivo si è scelto di non dedicare molte risorse computazionali e tempo di calcolo a questo aspetto e si è così utilizzata la tecnica Blind Maintenance Rules, ovvero si è trattato ogni pixel con la stessa regola.
La regola per l'aggiornamento di un pixel si ispira all'algoritmo della media Gaussiana, calcolando la varianza di un pixel negli ultimi N frame.
Si è quindi scelto eseguire questi calcoli in un thread secondario, per non rallentare la user experience.
Per la foreground detection, si è scelto di calcolare la distanza di ogni pixel appartenente al frame corrente dal corrispondente appartenente allo sfondo.

5 - L'algoritmo e le caratteristiche della soluzione

Nello sviluppo dell'app del "Tagliatore di teste" sono state riscontrate le seguenti problematiche:

- utilizzo di OpenCV in ambiente Android
- riconoscimento volto
- background subtraction
- spostamento interattivo del volto

5.1 - Utilizzo di OpenCV in ambiente Android

OpenCV fornisce degli strumenti di sviluppo così utili nel campo del trattamento digitale delle immagini da rendere il suo utilizzo in questo progetto praticamente indispensabile.

La libreria OpenCV possiede un package riservato completamente alle classi dedicate allo sviluppo in ambiente Android. Tali classi devono la loro esistenza a due esigenze, quella di includere le librerie in un progetto Android e quella di fornire un'interfaccia per l'utilizzo (dei dati) della fotocamera di un dispositivo che utilizzi tale sistema operativo.

A discrezione dello sviluppatore le librerie possono essere incluse nel progetto sia in modo statico che dinamico. Nel primo caso, da noi scelto, le librerie vengono fisicamente inserite all'interno del progetto e quindi dell'archivio in formato *apk* utilizzato per la diffusione dell'app, rendendo le sue dimensioni maggiori. Mentre la seconda alternativa costringe l'utente al download supplementare dell'app OpenCV Manager, la quale fornisce il linking delle librerie.
Si è scelto di agevolare l'user experience, a discapito delle dimensioni del progetto e della possibilità di aggiornamenti delle librerie, tenendo conto che comunque ad oggi, Luglio 2015, pur essendo stata rilasciata la versione 3.0 delle librerie, OpenCV Manager supporta fino alla versione 2.4.9.

Come detto, l'app si propone di lavorare in tempo reale nella fase di preview della fotocamera, questo è un comportamento comune ai software che rientrano nella categoria della computer vision. Per questo motivo OpenCV offre delle classi riservate all'ambiente android per interfacciarsi con l'input device (la fotocamera) e l'output device (lo schermo).
CameraBridgeViewBase e la sua specializzazione *JavaCameraView* sono le due classi da noi utilizzate per assolvere questo compito.
La classe *CameraBridgeViewBase* permette di settare un listener che implementi l'interfaccia *CameraBridgeViewBase.CvCameraViewFrame* a cui la vista si fa carico di inviare tutti i frame ricevuti dalla fotocamera e si aspetta di ricevere tali frame eventualmente modificati da poter mostrare a schermo come output.
Abbiamo quindi impostato la nostra main activity come tale listener.

In OpenCV le immagini sono rappresentate con la stessa struttura delle matrici, l'interfaccia sopra citata fornisce due metodi per convertire i frame in matrici, in quanto la classe Mat può essere utilizzata per memorizzare sia immagini in scala di grigi che immagini a colori, in particolare noi abbiamo utilizzato il metodo per lavorare con frame a colori.

5.2 - Riconoscimento volto

Tra i tanti strumenti che OpenCV fornisce, uno dei più utilizzati ed efficienti è quello del riconoscimento di oggetti grazie alla classe CascadeClassifier.
Questa classe sfrutta i file classifier, in formato xml, in cui vengono definiti i parametri che identificano un oggetto.
Nel SDK di OpenCV troviamo un esempio di face detection e quindi un file classifier, che definisce le regole per identificare un volto, che abbiamo quindi potuto sfruttare nel nostro progetto.
Tramite questo strumento si è potuto così ottenere una così detta Region Of Interest (ROI) rettangolare contenente il volto cercato.
Si è scelto in primo luogo di ridimensionare l'area in modo opportuno da contenere l'intera testa del soggetto (includendo fronte e capelli) e poi ottenere l'ellisse inscritto a tale rettangolo per rendere il taglio dal corpo meno accentuato.

5.3 - Background Subtraction

Una volta ottenuta l'area rettangolare che delimita la testa del soggetto è necessario distinguere i contorni del volto, in modo tale da spostare solamente l'area del volto e non anche la parte di sfondo contenuta nella ROI.
Come già visto si è scelto di utilizzare una tecnica rientrante nella categoria della Background Subtraction.
Il background viene inizializzato dall'utente stesso, tale background deve poi poter essere aggiornato, in caso di tempi di esecuzione prolungati, per poter registrare cambiamenti di luce o inserimenti di oggetti statici.

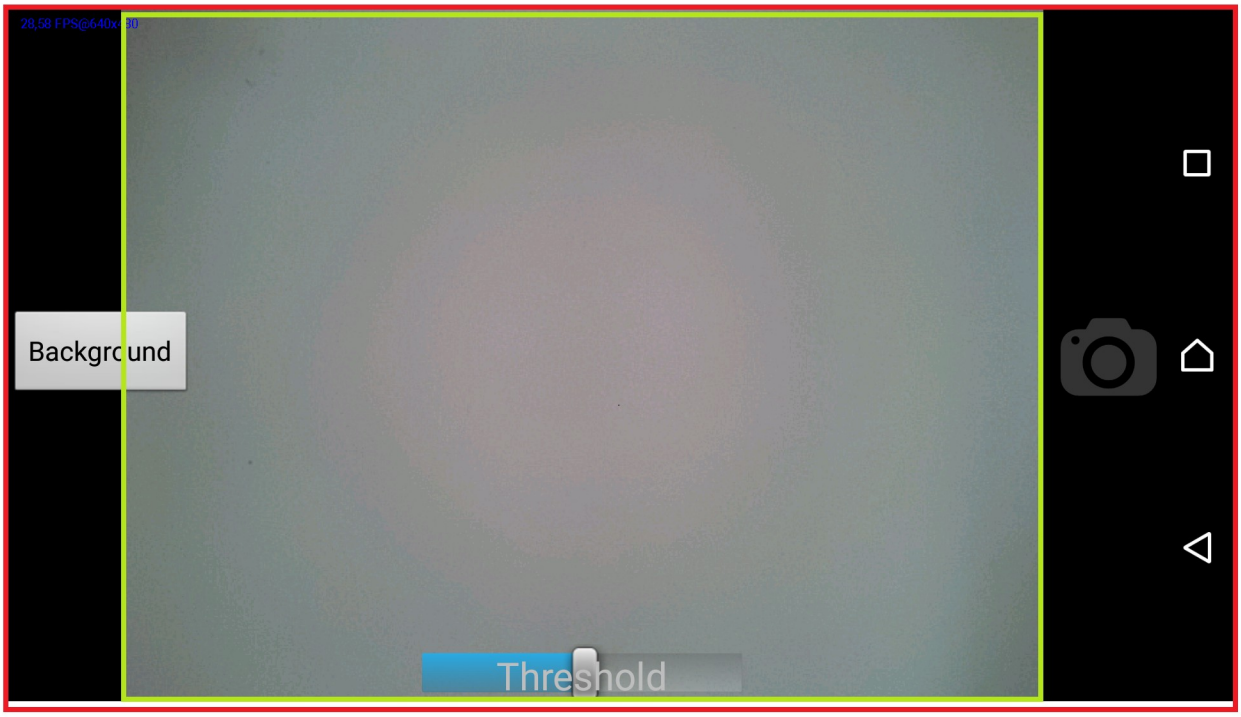
Tale aggiornamento viene eseguito in un thread secondario, esso riceve ad intervalli regolari dal core del programma il frame corrente. Ogni frame ricevuto viene confrontato pixel a pixel con i frame precedentemente ricevuti; qualora un pixel si mostri per niente o leggermente alterato viene considerato un pixel di background e quindi aggiornato nel background corrente.
La giusta calibrazione del valore di Adaption Rate, che indica la frequenza con cui lo sfondo viene aggiornato, e del valore di threshold utilizzato per valutare la distanza tra pixel sono fondamentali per evitare che oggetti in movimento, ma temporaneamente statici, vengano considerati come parte dello sfondo.

Per quanto riguarda invece la Foreground Detection si è scelto di confrontare pixel a pixel, l'area del frame contenente il volto, con la corrispondente area del background. In questo modo si ottiene una maschera binaria, in cui i pixel non settati a 0 rappresentano i pixel appartenenti al volto.
Questa tecnica risulta molto dispendiosa in termini di tempo, nel caso in cui l'area da esaminare sia molto vasta (volto in prossimità della fotocamera o risoluzione video elevata), si è scelto così di sotto-dimensionare i frame, in base alle impostazioni dell'utente, prima del calcolo della distanza pixel a pixel.

La distanza di un pixel (sui tre canali RGB) dal suo corrispondente viene confrontata con un valore soglia (threshold) anch'esso impostato dall'utente in un range opportuno; tale valore determina quali pixel fanno parte del foreground e quali no.

5.4 - Spostamento interattivo del volto

In fine il volto ottenuto deve poter essere spostato in un punto dello schermo scelto dall'utente. Per soddisfare questo comportamento la nostra attività principale implementa il listener per il tocco della vista *JavaCameraView*. In questo caso la difficoltà principale è stata quella di ricavare il pixel dell'immagine in cui centrare lo spostamento del volto pur disponendo solamente delle coordinate del punto dello schermo in cui ha cliccato l'utente. Infatti le dimensioni del video mostrato a schermo non coincidono con le dimensioni della vista che lo contiene o dello schermo o della matrice di pixel che la rappresenta. Come visibile nell'immagine di seguito, quello che si cerca di ottenere sono le coordinate di un punto relative all'area delimitata dalla cornice di colore verde mentre quello che si ottiene tramite il metodo implementato dal listener sono le coordinate del punto toccato dall'utente relative all'area delimitata dalla cornice rossa.



In OpenCV le immagini vengono solitamente convertite e rappresentate sotto forma di matrice. La nostra activity infatti riceve i frame catturati dalla fotocamera sotto forma di oggetti della classe Mat. Le dimensioni della matrice sono uguali alla risoluzione del frame catturato dalla fotocamera. Tuttavia lo schermo di un dispositivo Android può avere dimensioni variabili, quindi un frame prima di essere mostrato a schermo, deve essere scalato di un valore opportuno.
Ottenendo tale valore di scala, è possibile calcolare la grandezza in pixel dell'immagine mostrata a schermo (l'area con cornice verde). Avendo così la dimensione della vista (cornice rossa) e dell'immagine (cornice verde) è possibile ora determinare a partire da qualsiasi tocco sullo schermo, le coordinate del punto toccato in relazione all'immagine.

Il punto ottenuto, se interno all'area dell'immagine, rappresenta così le coordinate in cui centrare la destinazione del volto da spostare.
Si è utilizzata la funzione di OpenCV “copyTo” che permette di copiare una matrice (il frame del volto e il frame di background replace) in un'altra matrice (il frame originale). Tale funzione permette di utilizzare una maschera binaria, delle dimensioni del frame da copiare, che stabilisce quali pixel copiare e quali no. La maschera binaria, come visto, è determinata tramite il processo di Background Subtraction esaminato in precedenza.

6 - Complessità ed efficienza

La complessità dell'algoritmo va calcolata sul metodo “decapitate”. Esso riceve un frame dalla fotocamera, lo modifica e lo mostra a schermo. Quindi attraverso questo metodo passano tutti i frame mostrati a schermo, e tutti i frame catturati dalla fotocamera durante la modifica di un frame precedentemente catturato vengono saltati. Quindi da questo metodo dipende il frame rate dell'applicazione.

Quattro sono le operazioni che influiscono sulla velocità di questo metodo:

- face detection
- foreground detection
- copia della sotto-matrice del volto
- copia della sotto-matrice di background-replace

La face detection viene eseguita tramite funzioni della libreria OpenCV e non è quindi immediato il calcolo della complessità per questa operazione, si può comunque notare empiricamente che essa non penalizza in modo evidente il frame rate.
Le altre tre operazioni invece sono influenzate dalla grandezza delle sotto-matrici trattate.
Comunque anche le funzioni di copia delle matrici hanno beneficiato del metodo copyTo della libreria OpenCV e quindi pur essendo influenzate dalla grandezza dei frame, non è possibile dimostrarne precisamente la complessità.
Il *bottleneck* dell'applicazione è quindi costituito dal metodo “getBackgroundMask”, che restituisce la maschera utilizzata per la foreground detection. Esso esamina ogni pixel della sotto-matrice contenente il volto ed ha quindi complessità O(nm) con n e m corrispondenti al numero di pixel orizzontali e verticali dell'immagine. Per questo si è scelto di sotto-dimensionare tale matrice in base alle impostazioni dell'utente.

In fine è trascurabile il dispendio di tempo per le operazioni di background maintenance, esse infatti, pur avendo costo quadratico, vengono eseguite su un thread separato ad intervalli di tempo molto prolungati.