-discussione la complessità/efficienza stimata

Sommario

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

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.

Toccando il bottone che raffigura una fotocamera è possibile salvare in memoria il frame corrente.

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.

Descrizione del problema e delle ipotesi adottate

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

* [utilizzo di OpenCV in ambiente Android](#Import opencv|region)
* [riconoscimento volto](#face detection|region)
* [ricostruzione e rimpiazzamento dello sfondo dell'area lasciata libera dal volto](#background subtraction|region)
* [spostamento del volto in modo interattivo](#spostamento volto|region)

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.

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 che si è provveduto a ridimensionare in modo opportuno da contenere l'intera testa del soggetto (includendo fronte e capelli).

Una volta ottenuta l'area rettangolare che delimita la testa del soggetto è necessario distinguere i contorni del volto, per fare in modo di spostare solamente l'area del volto e non anche la parte di sfondo contenuta nella ROI.

Si è scelto di utilizzare la tecnica della Background Subtraction ( o Foreground Detection). Questa tecnica richiede il pre-salvataggio di un frame che rappresenti uno sfondo e il requisito che la fotocamera continui a puntare durante tutta la computazione la stessa area.

Questa tecnica prevede che per ogni frame corrente contenente un volto si computi la differenza assoluta di ogni pixel con il corrispondente pixel del frame background memorizzato.

Ogni pixel che risulta essere cambiato di un valore maggiore di una certa soglia (threshold), rappresenta un pixel del volto e quindi un pixel da spostare e da rimpiazzare con il vecchio pixel del frame background.

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.

Vedremo dettagliatamente in seguito come si è ottenuta questa corrispondenza.



L'algoritmo e le caratteristiche della soluzione

Il programma si compone delle sole due classi MainActivityCam e Boia.

La prima rappresenta l'unica activity dell'applicazione mentre la seconda il nucleo di elaborazione.

Complessità ed efficienza

per la ricostruzione dello sfondo, la nostra app prevede che inizialmente venga impostato un frame che rappresenti lo sfondo per la persona a cui "tagliare" la testa; all'onClick del bottone "**Background**",l'applicazione salva il frame corrente, che viene poi clonato in una nuova matrice dove andremo a lavorare.

Il metodo getBackgroundMask restituisce una matrice che rappresenta gli oggetti in foreground, dove ricercare il volto da "tagliare". Per far ciò si ha bisogno di calcolare la distanza tra la matrice che rappresenta il frame attuale e quella che rappresenta il frame salvato come background.

per il calcolo della distanza abbiamo utilizzato la classica distanza euclidea, con l'ausilio di una matrice di appoggio "diffImage", le cui celle sono state popolate dalla differenza in valore assoluto tra ogni elemento delle due matrici, ovvero tra ogni pixel dei frame.

per ogni cella abbiamo controllato che lo scalare rappresentante la differenza di colore delle due maschere avesse un valore più alto di una soglia stabilita e che tale cella si trovasse all'interno di un ellisse inscritta dentro il rettangolo del volto.

le celle che hanno superato le due condizioni le abbiamo inserite nella maschera di foreground e quindi abbiamo stabilito i pixel da ridisegnare nel frame.

 the array assignment is an O(1) operation because it only copies the header and increases the reference counter. The Mat.clone() method can be used to get a full (deep) copy of the array when you need it.