-una breve relazione che contenga:

-descrizione del problema e delle ipotesi adottate

-l'algoritmo e le caratteristiche della soluzione

-discussione la complessità/efficienza stimata

- dati di esempio ed eventuali test comparativi sulle prestazioni

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.

Ambiente di sviluppo

Accenno ad android.

Il progetto si basa sulla libreria OpenCv, una libreria open-source, scritta in C++, per la computer vision, ovvero per la trasformazione dei dati provenienti da una fotocamera o videocamera.

\*approfondire il fatto che è scritto in c++, come si interfaccia con java, dire che c'è l'sdk android

Introduzione

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

* utilizzo di OpenCV in ambiente Android
* riconoscimento di un volto
* spostamento del volto in modo interattivo
* ricostruzione e rimpiazzamento dello sfondo dell'area lasciata libera dal volto

Come analizzato nella seguente sezione, 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.

Per l'interazione tra la camera e la libreria abbiamo utilizzato la classe "CameraBridgeViewBase", la quale controlla quando la fotocamera può essere attivata , processa i frame e chiama i listener per apportare eventuali modifiche ai frame.

Della classe CameraBridgeViewBase abbiamo utilizzato l'interfaccia "CvCameraViewFrame", che è la rappresentazione astratta di un singolo frame.

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.

Per la face detection abbiamo utilizzato CascadeClassifier, classe per la detection di un qualsiasi tipo di oggetto. Il costruttore di tale classe prende in input un file in cui è contenuto il classificatore dell'oggetto da riconoscere. Nel nostro caso abbiamo cercato il classificatore per il riconoscimento frontale delle facce e caricato quindi il documento xml "lbpcascade\_frontalface".

riconosciuto il volto (se ci sono più volti, viene riconosciuto il volto più grande), questo viene salvato in un oggetto della classe rect, il rettangolo che abbiamo utilizzato per il salvataggio del volto prende delle dimensioni ....NON mi è chiaro che gioco c'è tra rect matofrect e mat.

Per decidere dove spostare il volto basta attivare(???) l'evento onTouch(). A questo punto vengono salvate le coordinate del punto toccato nello schermo, che rappresentano il punto dove posizionare il centro della maschera con il viso. La zona di destinazione dell'immagine è rappresentata dal rettangolo "destFaceROI", il quale viene costruito a partire dal suo punto (0,0). Abbiamo ricavato questo punto rispetto al punto toccato nello schermo, considerando l'eventualità che le coordinate del rettangolo da creare potessero essere esterne allo schermo e decidendo in questo caso di settare il valore della coordinata "sballata"al punto più prossimo dello schermo.

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.