Dalty. Applicazione Android per la correzione di immagini per soggetti "color blind"

Corso di Sistemi Operativi II - Prof. D. Riccio

Studente: Andrea Sorrentino N97/221

9 luglio 2016

Indice

1 Introduzione

Dalty é un'applicazione Android che si propone come scopo la modifica dei colori in un'immagine al fine di migliorare la percezione del colore in soggetti "color-deficient".
L'essere umano é in grado di percepire il colore tramite tre classi di *fotorecettori* (o coni) presenti nella retina:

- long-wavelength (L)
- middle-wavelength (M)
- short-wavelength (S)

Pertanto, la normale visione umana é detta *tricromatica*. La modifica di una di queste tre classi di coni determina la CVD (Color Vision Deficiency). Ci sono tre tipi di CVD:

1. *dichromacy*
2. *trichromacy*
3. *achromatopsia*

In particolare per la 1 si individuano tre ulteriori specializzazioni:

- *protanopia*
- *deuteranopia*
- *tritanopia*

L'applicazione in questione si occupa di fornire, nel suo piccolo, una soluzione per questi ultimi tre casi. Il funzionamento dell'algoritmo utilizzato é spiegato in dettaglio nella sezione 2.2.1, mentre per una comprensione approfondita dell'applicazione si rimanda alla sezione 2 ed alle sue sottosezioni.

2 Descrizione dell'applicazione

L'applicazione, come giá accennato, consente la modifica dello spettro di colori di un'immagine in modo da facilitare la visualizzazione della stessa a soggetti "color blind". All'avvio dell'app, é proposta all'utente la seguente schermata: Come é facilmente intuibile, l'utente puó selezionare la 'patologia' desiderata mediante uno spinner posto nella parte

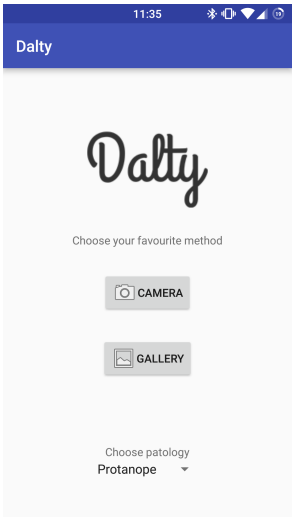

Figura 1: Schermata principale dell'applicazione

bassa dello schermo. Sulla base di questa scelta l'algoritmo modificherá il suo flusso, adattandosi all'input utente. Una volta selezionata la patologia é opportuno scegliere il metodo di acquisizione dell'immagine:

1. dalla fotocamera
2. dalla gallery

Nel caso si scelga la prima opzione, l'applicazione lascerá il controllo alla fotocamera predefinita dello smartphone, l'utente acquisirá la foto, che sará quindi salvata, processata dall'algoritmo(sezione 2.2.1) e successivamente mostrata a video in una image view. Il procedimento appena descritto é esposto graficamente nell'immagine seguente:

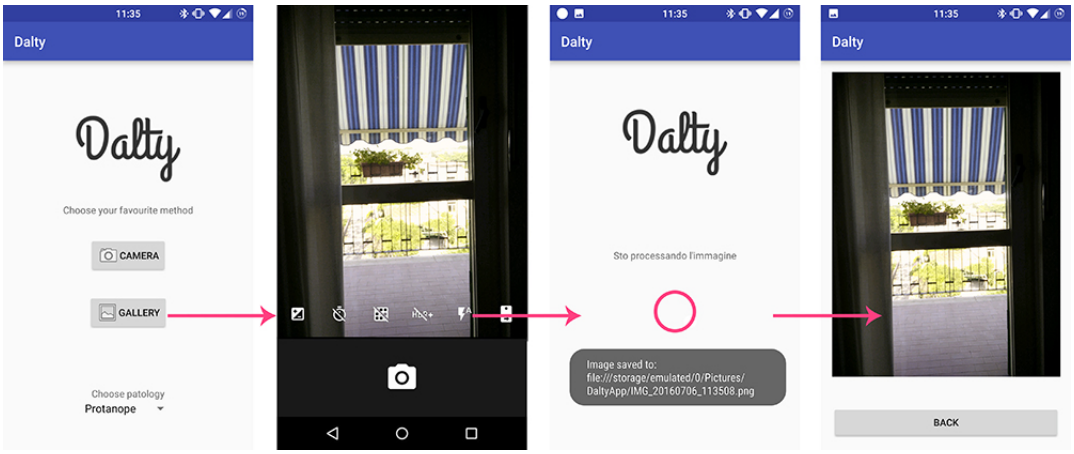

Figura 2: Flow dell'applicazione per selezione immagine tramite fotocamera

Allo stesso modo, in caso l'utente scelga la seconda modalitá, l'applicazione lascerá il controllo alla gallery predefinita dello smartphone, l'utente selezionerá la foto, che sará quindi processata dall'algoritmo(sezione 2.2.1) e successivamente mostrata a video in una image view. Il procedimento appena descritto é esposto graficamente nell'immagine seguente:

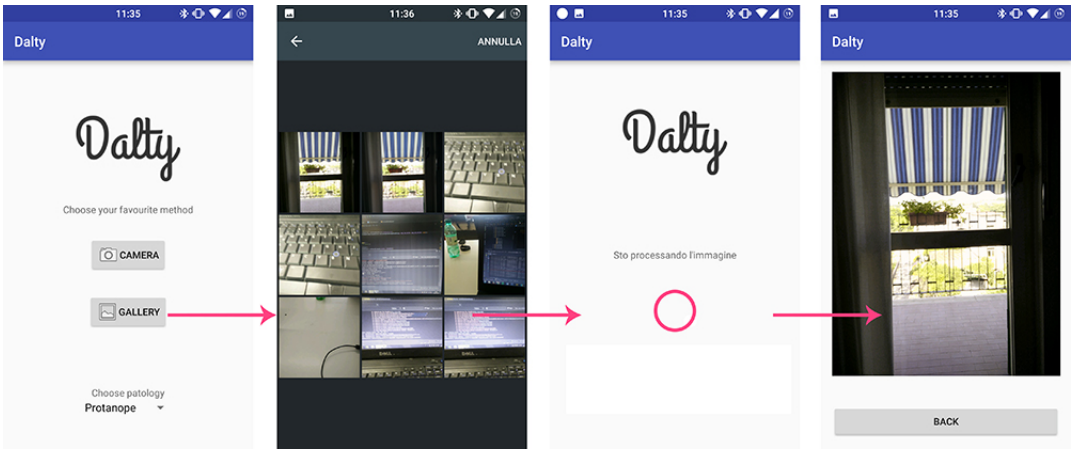

Figura 3: Flow dell'applicazione per selezione immagine da gallery

Chi usa l'app ha, ovviamente, la possibilitá di tornare indietro e ripetere il procedimento ad oltranza. É bene notare, inoltre, che l'utente, una volta lanciato il processing dell'immagine, puó continuare ad utilizzare normalmente lo smartphone. Infatti, tutta la parte di elaborazione viene effettuata in un IntentService, separato quindi dalla presenza a schermo dell'activity principale.

2.1 La MainActivity

La MainActivity é tra i componenti principali del sistema. Si occupa, infatti, di gestire i vari stati dell'applicazione, in particolar modo a livello grafico, oltre a coordinare la logica dell'app in generale. Per controllare al meglio le dinamiche interne del sistema, si é scelto di implementare una versione piú leggera del design pattern state , riuscendo fondamentalmente cosí ad ovviare ad una serie di difficoltá che l'interazione con l'utente e la comunicazione con il Service che lavora in background comportano. Cosí facendo, l'activity risulta essere in grado di gestire in modo autonomo le tre schermate dell'interfaccia, con particolare attenzione per quanto riguarda le transizioni tra i cambi di stato dell'app.

Per quanto riguarda la schermata principale la MainActivity gestisce le dinamiche dello spinner ed implementa delle callback per i button *"CAMERA"* e *"GALLERY"*. Tali metodi semplicemente si occupano di lanciare i rispettivi intent per la gestione dell'operazione corrispondente.

Il metodo *"onActivityResult"*, oltre ad essere un handler per il processing del risultato di tali intent, lancia anche il Service per l'elaborazione dell'immagine 2.2. Similmente, il metodo *"onReceiveResult"* controlla che il risultato dell'operazione sia conforme alle aspettative ed in caso positivo consente di visualizzare l'immagine processata in una ImageView, a patto che l'activity sia visibile a schermo. É possibile che non ci si trovi in questa situazione, in tal caso il metodo procede a salvare il path dell'immagine in questione. Quest'ultima sará poi caricata e visualizzata a tempo debito, quando l'utente riattiverá l'app.

Ovviamente, la MainActivity si occupa anche di fornire all'utente un feedback costante, sia esso tramite notifiche, messaggi toast, o modifiche all'interfaccia.

2.2 Il DaltonizeService

La classe DaltonizeService estende un IntentService, cosí facendo si rende piú snella la gestione delle operazioni che é necessario effettuare al di fuori del thread di UI. Si é scelto di implementare un IntentService invece di un Service convenzionale perché, oltre ad avere un lifecycle gestito quasi tutto in modo automatico, risulta piú adatto allo scopo dell'app che richiede semplicemente un'elaborazione in background di tipo momentaneo. Al contrario un Service sarebbe stato piú idoneo in caso di necessitá di monitoraggio continuo nel tempo di uno o piú aspetti del dispositivo. L' IntentService riceve in input:

- Un oggetto per la comunicazione con l'activity
- Il path del file da processare
- La patologia selezionata

Innanzitutto viene caricata in memoria l'immagine, sottoforma di Bitmap. Siccome lo spazio in memoria é limitato, si fa uso della classe di utility, ImageResizer, che mette a disposizione metodi per il ridimensionamento delle immagini, in particolare il limite per il caricamento delle immagini é stato stabilito in modo empirico a $1500 * 1500$, per cui si controlla che l'immagine da caricare non abbia piú di 2250000 pixel totali, se ció dovesse accadere l'immagine sarebbe ridimensionata in modo opportuno e caricata in memoria. In caso di malfunzionamento di tale procedura, si invia un messaggio di errore all'activity e si termina il Service. Il Service fa uso della libreria *opencv* per manipolare le immagini con semplicitá ed in particolare della classe Mat. Entrando nello specifico vengono istanziate due Mat (src e dst) delle stesse dimensioni e con lo stesso contenuto della bitmap appena caricata. Si applica quindi l'algoritmo di cui si forniscono i dettagli alla sezione 2.2.1. Al termine la Mat dst viene convertita nella Bitmap $destBitmap$ e viene salvata in persistenza in formato PNG (formato looseness che consente di preservare i colori senza attuare alcuna compressione dell'immagine). A questo punto, se le operazioni descritte sono andate a buon fine, si invia un messaggio all'activity contenente il path della nuova immagine e si termina il Service.

2.2.1 L'algoritmo alla base

L'algoritmo di 'Daltonization' si puó scindere in due parti fondamentali. Per prima cosa viene analizzata l'immagine e, sulla base di algoritmi comunemente accettati in letteratura, si simula la patologia selezionata. Successivamente si applica all'immagine 'simulata' una procedura che consente di migliorare la percezione dei colori per persone "color-blind".
Entrando piú nel dettaglio, si analizzano i singoli pixel dell'immagine e per ogni pixel si estraggono i rispettivi canali R,G,B. Una volta ottenuti i canali si convertono i valori dallo standard RGB a LMS (grazie ad una matrice di conversione comunemente accettata). A questo punto si simula la patologia moltiplicando il vettore LMS per una matrice (diversa a seconda della patologia), ottenendo cosí il vettore $LMSp$ che viene poi convertito in $RGBp$ grazie ad un'altra matrice(l'inversa di quella usata per convertire da RGB ad LMS). Lo step successivo prevede di ricavare una matrice di errore $E = RGB - RGBp$, che consenta di shiftare i colori nello spettro di quelli visibili dal soggetto in questione. Si moltiplica poi la matrice di errore E per un'altra matrice (diversa a seconda della patologia, ed anch'essa comunemente accettata) in modo da ottenere $Emod$, una matrice di compensazione dei colori. Quest'ultima é infine aggiunta al vettore originale RGB in modo da ottenere una versione daltonizzata dell'immagine. Si fornisce di seguito una versione in pseudocodice dell'algoritmo:

Algorithm 1 Daltonization algorithm

procedure DALTONIZE
  for ogni pixel p dell'immagine img **do**
    $RGB \leftarrow estraiCanali(p)$
    $LMS \leftarrow convertiToLMS(RGB)$
    $LMSp \leftarrow LMS * Msim$//Msim=matrice per simulare la patologia
    $RGBp \leftarrow convertiToRGB(LMS)$
    $E \leftarrow RGB - RGBp$
    $Emod \leftarrow E * Mcor$//Mcor=matrice per compensazione colori
    $RGB \leftarrow RGB + Emod$
  end for
  return img
end procedure

Per una spiegazione piú approfondita dell'algoritmo e per comprenderne appieno i dettagli si rimanda al paper *"Intelligent modification for the daltonization process of digitized paintings"(Christos-Nikolaos Anagnostopoulos, George Tsekouras, Ioannis Anagnostopoulos Christos Kalloniatis)* link al paper

3 Use Case

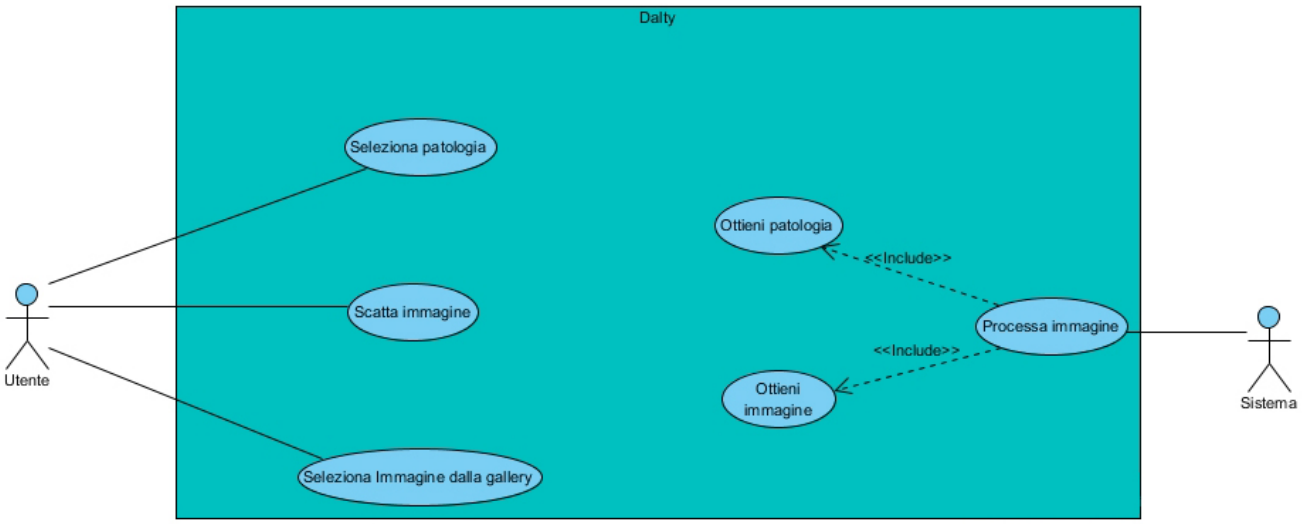

Figura 4: Use case dell'applicazione

4 Testing

L'applicazione é stata testata con una buona varietá di immagini, molte delle quali presenti in letteratura e che consentono di verificare il corretto funzionamento dell'algoritmo implementato. Un'esempio di immagine testata é il seguente, che é direttamente ripreso dal paper precedentemente citato alla sezione 2.2.1.

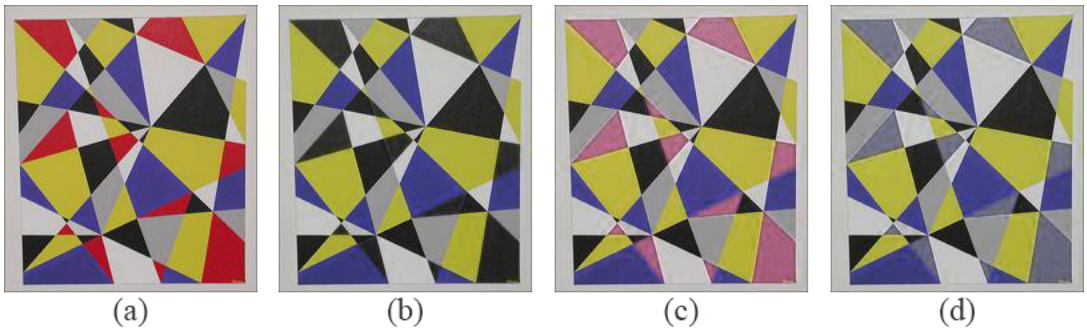

Figura 5: (a)Immagine originale, (b) visione dal punto di vista del soggetto color-blind, (c) immagine processata dall'applicazione, (d) visione dal punto di vista del soggetto color-blind della

É stato possibile anche far testare a dei soggetti "color-blind" l'applicazione in modo da valutarne l'efficacia e l'utilitá. Questi ultimi hanno espresso pareri positivi riguardanti sia il flusso di controllo che l'outcome dell'elaborazione. Nonostante ció, sono state proposte dai tester delle migliorie riguardanti la terza schermata che dovrebbe consentire all'utente un'interazione maggiore con l'immagine processata.