UNIVERSITÀ DEGLI STUDI DI TRENTO

Report “Filtri S-M”

A.A. 2019/2020

Uez Davide

Calearo Luca

Stefanolli Paolo

Francesco Zoccatelli

SOMMARIO

1 Introduzione

Al giorno d'oggi scattare fotografie è un'azione comune, che ogni persona compie per condividerle con amici e conoscenti oppure per mantenere un ricordo di un momento . Con l'aumento di popolarità dei social network come Snapchat e Instagram i managers delle aziende produttrici di questi servizi hanno dovuto trovare nuove funzionalità per le loro app. Da qui l'idea di applicare delle immagini al volto delle persone inquadrate dalla telecamera: nascono così i filtri facciali.

1.1 Idea del progetto

L'idea del progetto è quella di implementare un' applicazione desktop che permetta all'utente di selezionare dei filtri da applicare a sequenze video. Sarà poi possibile effettuare uno screenshot per salvare la propria foto con il filtro applicato.

1.2 Fasi temporali del progetto

Le principali fasi del progetto sono state:

- Individuazione dei possibili filtri da implementare
- Implementazione dei filtri scelti
- Implementazione della GUI
- "Assemblaggio" GUI e filtri
- Stesura report finale

Per raggiungere questi obiettivi inizialmente siamo partiti dalla ricerca dei linguaggi e delle librerie necessari per lo sviluppo dei filtri (Python, OpenCV e dlib).

Una volta scelto l'ambiente di sviluppo ci siamo concentrati nell'identificazione dei filtri, pensando a dei filtri simpatici e belli da utilizzare. Finita questa fase abbiamo iniziato ad implementarli utilizzando solamente funzioni di opencv e dlib. Siamo passati poi alla scelta dei 5 filtri che compongono l'applicazione, scegliendo quelli che per noi sembravano migliori.

In parallelo abbiamo lavorato alla parte di identificazione e creazione dell'interfaccia grafica utilizzando Tkinter.

Abbiamo concluso poi con il "merging" della GUI e dei filtri, facendo effettuare dei test ad alcune persone.

È possibile trovare il contributo di ciascun membro del team nel file Excel al seguente link:

https://drive.google.com/drive/u/1/folders/1HEV5XLzW4STjbsqt-9HEW94Jd7KIkFIV

1.3 Materiale progetto

Il materiale del progetto può essere trovato nella cartella di Google Drive al seguente link:

https://drive.google.com/drive/folders/1NnoC50lsFeSnIzAu7d_MWq1bHN9DQ0fy?usp=sharing

2 Linguaggi e librerie utilizzate

Inizialmente siamo stati un po' indecisi riguardo ai linguaggi ed alle librerie da utilizzare, in quanto eravamo più indirizzati nello sviluppare un applicativo per mobile.

Oltre a questo l'idea iniziale di usare gli "haarcascades" per localizzare un volto nella schermata si è rilevata avere scarse prestazioni, ed è stato deciso quindi di optare per le librerie di dlib, che offrono maggiore stabilità e precisione nella rilevazione dei keypoint facciali (utilizzati per posizionare le immagini che compongono i filtri).

Per ovviare al problema dello sviluppo su mobile, abbiamo optato per un applicativo desktop implementato grazie all'utilizzo di Tkinter, una libreria python per la creazione di GUI

2.1 Python e OpenCV

Per la realizzazione dell'applicazione abbiamo utilizzato principalmente le funzioni fornite da OpenCV, che è una libreria software multipiattaforma nell'ambito della visione artificiale in tempo reale. Per interfacciarci con questa libreria abbiamo scelto come linguaggio di programmazione Python 3.7

Alcuni esempi di funzioni OpenCV utilizzate sono:

- cv2.imread( ) : legge un'immagine dal path passato come argomento
- cv2.cvtColor( ) : converte l'immagine da uno spazio colore ad un altro
- cv2.videoCapture( ) : fa partire lo streaming video dalla webcam passata come parametro
- cv2.imshow( ) : mostra l'immagine/video nella finestra indicata

Queste funzioni ci permettono non solo di caricare immagini e convertire colori, ma soprattutto ci forniscono le basi per andare poi a lavorare sui filtri stessi. Con gli ultimi due comandi infatti riusciamo a far partire lo stream video dalla webcam ed a vederne la registrazione dal vivo in una semplice finestra.

2.2 Python e dlib

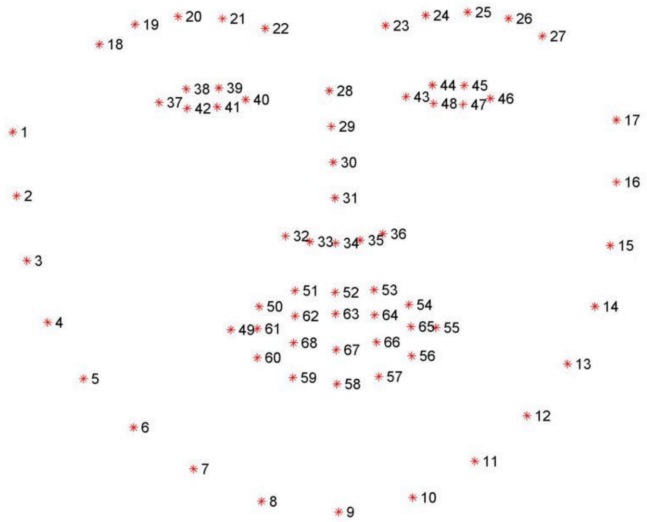
Il passo successivo è stato quello di trovare una libreria che permettesse di rilevare dei volti nella sequenza video. Inizialmente siamo partiti utilizzando gli "haarcascades" ma risultavano molto "laggy" e non molto precisi. Effettuando altre ricerche abbiamo trovato dlib che da citazione: "Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems."

La funzionalità che interessa a noi di questa libreria è l' "Image Processing", che contiene al suo interno:

- Tools for detecting objects in images including frontal face detection and object pose estimation.
- High quality face recognition

Inizialmente abbiamo utilizzato la funzione "dlib.get_frontal_face_detector()" che ritorna il face detector di default.

Successivamente grazie alla funzione "dlib.shape_predictor()" ed al file "shape_predictor_68_face_landmarks.dat" siamo riusciti ad identificare sui volti rilevati dal face-detector le coordinate di 68 punti che mappano i punti facciali chiave di ogni persona.



Una volta fatta partire la videocamera ed individuato il volto a questo punto applichiamo le immagini png alla sequenza video, facendo coincidere le immagini alle coordinate dei punti ritornati dai predictor.

2.3 Python e Tkinter

Per l'implementazione dell'interfaccia grafica abbiamo scelto di utilizzare Tkinter che è un pacchetto standard Python per la creazione di interfacce grafiche. L'idea iniziale di sviluppare un'applicazione Android non è andata a buon fine in quanto ci siamo trovati molto in difficoltà nel momento in cui abbiamo dovuto unire i filtri con l'interfaccia grafica.

Cambiando dispositivo quindi è cambiata anche l'idea del come mostrare all'utente un'interfaccia semplice e funzionale. Abbiamo pensato quindi ad un applicazione desktop divisa in:

- Menù principale: permette di selezionare i filtri che si desiderano applicare. Qui vengono visualizzati anche il nome dell'applicazione e le istruzioni d'uso
- Schermata videocamera: finestra che compare una volta selezionato il filtro, nella quale appare il video registrato dalla webcam con il filtro applicato
- Schermata salvataggio: una volta effettuato lo screenshot della webcam compare questa finestra che mostra l'immagine scattata e chiede se si desidera salvarla o eliminarla

Gli oggetti base di Tkinter sono finestre, bottoni, label, ecc.

Qui di seguito sono rappresentate delle illustrazioni del wireframe riguardante la struttura dell'app, la quale è stata poi effettivamente sviluppata usando Tkinter

3 Descrizione filtri

Per la creazione dei filtri applichiamo delle png sopra ai frame. Le immagini sono rappresentate inizialmente attraverso matrici a tre dimensioni (RGB), poi attraverso l'utilizzo della funzione cvtColor di openCV aggiungiamo una quarta matrice che rappresenta l'alpha channel.

Dopo che il programma rileva un volto ed a sua volta i landmark relativi ad esso applichiamo delle funzioni di resizing che permettono di far variare la dimensione dei png in base alla grandezza del viso (dipendente dalla distanza dalla videocamera).

Abbiamo dovuto testare e modificare il numero di volte che questa funzione viene applicata, in quanto abbiamo notato fosse un parametro critico per al fluidità del programma.

3.1 Filtro Tattoo

Il filtro Tattoo applica due png colorate, che rappresentano 2 tatuaggi , sulla guancia destra e sulla fronte.

Per quanto riguarda gli effetti statici, essi non sono stati modificati in questo filtro, quindi l'unica alterazione rispetto al video ripreso è l'applicazione delle png.

3.2 Filtro BD-Loove

Il filtro BD-Loove si compone di un paio di occhialetti da lettura posizionati sulla parte centrale del naso e da un paio di baffi molto pronunciati. Per dare il tocco retrò al filtro abbiamo pensato di applicare un filtro statico in scala di grigi, rendendo tutto un po' più anni 30.

3.3 Filtro Magnum P.I.

Questo filtro è dedicato alla serie televisiva “Magnum P.I.”.

La nostra intenzione era quella di rappresentare nel modo più verosimile possibile il protagonista, applicando così i suoi inconfondibili baffi e degli occhiali stile Rayban.

Se al filtro precedente abbiamo assegnato la categoria “retrò” a questo potremmo assegnargli la categoria “police”

3.4 Filtro JoJo

L’idea di questo filtro era di fare qualcosa di contemporaneo ed artistico, quindi abbiamo optato per il tema “cartoon”.

Abbiamo applicato un filtro statico (crisp) che dà un effetto “cartonesco” allo scatto. Questo filtro accentua i colori e delinea in modo marcato i bordi degli oggetti

Come immagini invece abbiamo applicato due png a lato del viso tipiche del famoso cartone JoJo.

3.5 Filtro Ouch

Abbiamo applicato due png, una sull’occhio destro ed una sulla guancia sinistra.

L’ispirazione per questo filtro ci è venuta durante la pausa pranzo guardando una puntata di Spongebob.

4 Risultati testing sugli utenti

Per quanto riguarda la fase di testing abbiamo fatto provare l'applicazione ad alcuni nostri conoscenti, per avere dei feedback per quanto riguarda il funzionamento e la grafica. Da questa fase di testing abbiamo ricavato le seguenti informazioni positive e negative:

- semplicità nell'utilizzo del software
- filtri simpatici e misti (categorie e tipi diversi)
- facilità nel salvataggio ed eliminazione delle foto scattate
- frame rate piuttosto basso su Mac, più ottimizzato per Windows
- manca la funzionalità che permette di registrare sequenze video
- alcuni filtri non funzionano quando rilevano più di un volto

5 Conclusioni

Valutando i risultati della fase di testing abbiamo constatato che possibili sviluppi futuri dell'applicazione prevedono:

- la diminuzione del lag / aumento degli FPS nella schermata della videocamera
- implementazione di una funzione per registrare sequenze video
- correzione bug di alcuni filtri che non funzionano con più volti

Sviluppando questa applicazione abbiamo imparato ad utilizzare dei software per l'Image Processing applicati a fatti reali (filtri nei social network), quali il riconoscimento facciale e l'applicazione di filtri dinamici.