Extended Snapchat Filter:

Using OpenCV Facial Recognition for fun

Max Hilgenberg

CPSC 4820 Section 3

Clemson University

mhilgen@g.clemson.edu

Preview

1. Introduction

When it came to choosing a topic for the Final Project, I was intrigued by the idea of using our Facial Recognition assignment. I wanted to place facial accessories like hats, glasses or even facial hair onto my face using the coordinates given by the Cascade methods from OpenCV. I knew one of the struggles I would have to figure out was how to place an image above another image without overwriting the original image. Another struggle would be resizing the accessories to fit onto my face without hiding my face too much.

2. Background Information

When going through the examples we were shown from past students, I thought that the Snapchat filter idea was actually pretty good, but I was a bit confused on how the student got his version to work. So, I made my own plan on how to work this out. I used the cascade xml files and the *detectMultiScale()* method to find the facial features on my face with the necessary coordinates. For example:

faces = faceCascade.detectMultiScale(gray, 1.05, 5, minSize=(w//6, h//6))

Then I used trigger buttons, which are basically just Booleans that are initially False, which when triggered enable the accessory requested to be placed on the face. This gave me the opportunity to make the program more interactive, in which the user can choose what type of accessory they would like on their face. This also enabled the chance to add many different accessories to your face without needing to stop the program.

3. Methodology:

So, I started to rework my Facial Recognition software, since it did not need to find all the facial features on the face. The 3 cascade xml files I used were the face, the nose and the left eye. The reason for that was that I wanted to use the nose mainly to put facial hair or glasses

on the face, while using the left eye for a monocle and other accessories. So far it was all working out fine, since it recognized the facial parts. There was just one main problem. I had no idea how to put these accessories onto the frames. For example, I could not put the glasses on my nose, and I was sure that I could not just draw the glasses on there with OpenCV. So, I started looking for some solutions. I went through a lot of stack overflow articles and other coding websites, but I ended up empty handed. Luckily enough I accidently clicked on the video section of the search engine, in which I found this great video. The creator of this video had found a way to put a mustache right below his lip. So, I read through his code and tried to make it work with mine as well. I did try to find another mustache but after looking for better images his mustache kept popping up as the ideal solution, so I stuck with it. I did add some more ideas to the overlay to ensure that it does exactly what I want it to do. I believe, I added around eight different accessories which you could add to your face: the mustache, the beard, sunglasses, a monocle, a disguise, a Manchu (Japanese mustache), a graduation hat and a fixed picture of a hat with a monocle and mustache attached. I needed of course to find images for each one of them that were transparent otherwise, the "white" image would be overlaid above the original image, which is just straight up ugly. When I had all the right pictures saved in my filter directory it was time to resize each picture and place them correctly with the suitable cascade method. For Example:

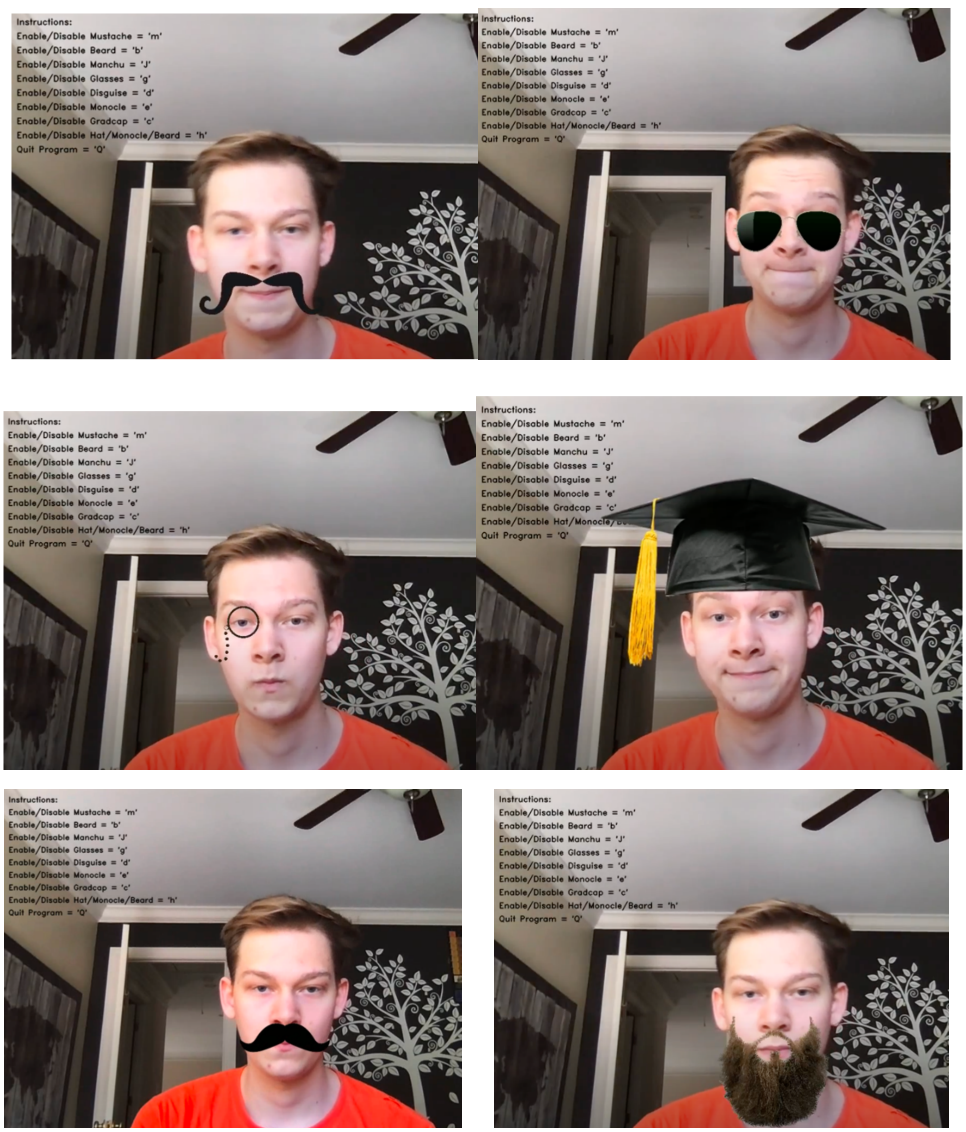
```
noses = noseCascade.detectMultiScale(grayroi, minSize=(w//6, h//6))
for nx, ny, nw, nh in noses:
    if manchu_button:
        manchu = cv2.imread('filters/manchu.png', -1)
        oh, ow = manchu.shape[:2]
        scale = .85 * w / ow
        layer = cv2.resize(manchu, (0, 0), fx=scale, fy=scale)
        oh2, ow2 = layer.shape[:2]
        xx = x2 + nx + (nw // 2) - (ow2 // 2)
```

yy = y2 + ny + (ny // 2) + (oh2 // 12)
frame = filter(frame, layer, xx, yy+10)

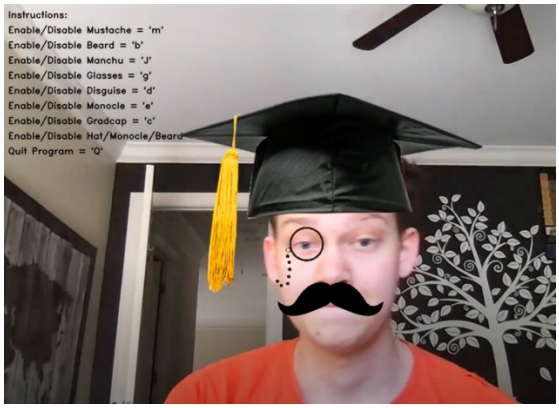
The easiest part at the end was just playing around with the coordinates in relation to the nose and the left eye to ensure that the accessories were not directly on the nose but perhaps a bit below or above it.

4. Results:

Surprisingly after tinkering more and more with the coordinates and sizes of the accessories in relation to my face I had a working, interactive Snapchat filter program. Here are some examples, that I screenshotted from the output mp4 file:



And a funny combination:



5. Conclusion:

This Final Project was a lot of fun to program and play around with since it was perhaps something more different from other programming classes that I have taken so far. I also think that I found my own way to write this program and had a different solution that the student with the initial idea. This program does not necessarily have any use, but it is than fun to just play around with it. I definitely was surprised how well it worked for most of the accessories I used, but I would have liked to improve the beard and the hat/mustache/monocle combo. I was really pleased with the monocle and the glasses, since my Facial Recognition program before had issues with the eyes which I apparently solved in this project.