Glasses Frame Animation with Python and OpenCV

Calvin Cash
Clemson University
Clemson, SC

Figure 1: Glasses animation with Python and OpenCV

ACM Reference Format:
Calvin Cash, Clemson University, Clemson, SC. 2022. Glasses Frame Animation with Python and OpenCV. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

In recent years, technological innovations have given way to a rise of online services and markets that previously existed only in physical stores and locations. Online shopping has begun to take over the retail industry with the rise of giants like Amazon, Ebay, and Alibaba. Even some services like patient screenings for health clinics and college classrooms have moved to virtual spaces. Recent advances in virtual reality and augmented reality have allowed customers to virtually tour a home or use their cellphone camera to view what a new appliance would look like in their kitchen. However, one area of consumer interest that I have found lacking is the ability for people to virtually try on clothes or accessories, such as a pair of glasses. This was the motivation for this project, to allow a user to virtually try on a pair of glasses in real time using their laptop or cellphone camera. Using Python, OpenCV, and the Dlib library's 68 point facial detection model, I was able to create a program that, given an image of a pair of glasses, could detect a person's face and draw the glasses on an outputted image. The program was moderately successful in placing the glasses correctly centered around the user's eyes and rotate with facial rotations about the x-axis of the user's face.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 INITIAL FACE DETECTION

In order to animate glasses onto a user's face, I first had to write a program that would recognize a human's face and facial features, mainly the eyes. During the semester, I wrote a program for this purpose for an assignment by implementing different Haar cascade filters trained to detect a face or facial features. The Haar cascades return pairs of Cartesian coordinates representing the top left vertex of a rectangle encompassing the feature they are trained to detect, as well as the length and width of the rectangle.

Calvin Cash
Clemson University
Clemson, SC

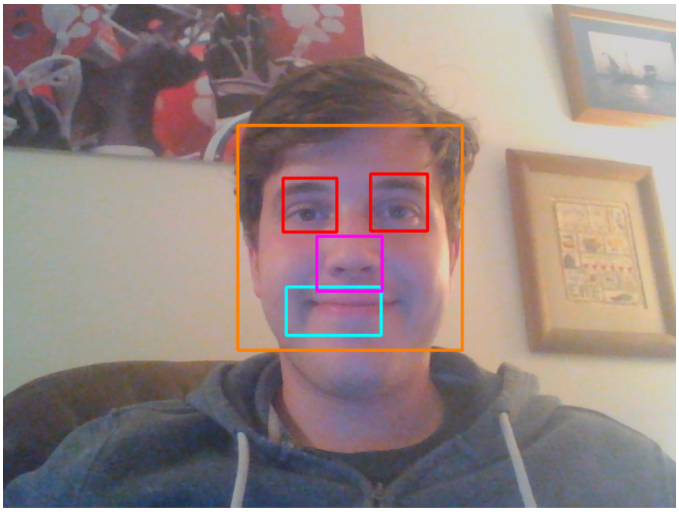

Figure 2: Features detected by Haar cascade filters

Figure 3: Exaple image of glasses frames

So, using the coordinates for the right and left eyes and removing false positives, I could construct a rectangle representing the area of the face containing the eyes where a pair of glasses would rest. Overlaying an image of a pair of glasses into that rectangle was a simple matter.

However, the Haar cascades do not deal well with facial rotations and always return coordinates for rectangles aligned vertically and not with the face. So, to animate glasses to rotate and tilt as the user's face tilts, I needed a more sophisticated facial recognition method that could estimate facial posture.

3 ADVANCED DETECTION AND POSE ESTIMATION

The problem of pose estimation is one I solved in an assignment with the use of ArUco markers. The heart of the problem is understanding the rotation of a surface about all 3 axes so that 3d points can be properly projected onto a 2d image. In order to solve the pose estimation problem, you must have a theoretical 3d model of a 2d image produced by a camera. OpenCV has a built in function, *projectPoints* to solve this problem, which works rather well for markers that have very precise, defined shapes on a flat surface. A person's face is more difficult to model as no two persons have the same exact same facial structure, but I used a common model that works decently well [2]. I then found a more advanced facial recognition model in the Dlib library that contains a model to estimate 68 points on a person's face [1].

Using the 3d model and the Dlib facial recognition package, I was able to use OpenCV's *projectPoints* function to fairly accurately project a frame around a user's eyes that rotated as the user's face rotated. Using the same method as before for overlaying the glasses, I was able to produce images and videos that animated glasses frames on people's faces with a fair degree of accuracy.

Figure 4: Haar detection and glasses draw with no facial rotations

Figure 5: Haar detection and glasses drawn with facial pitch

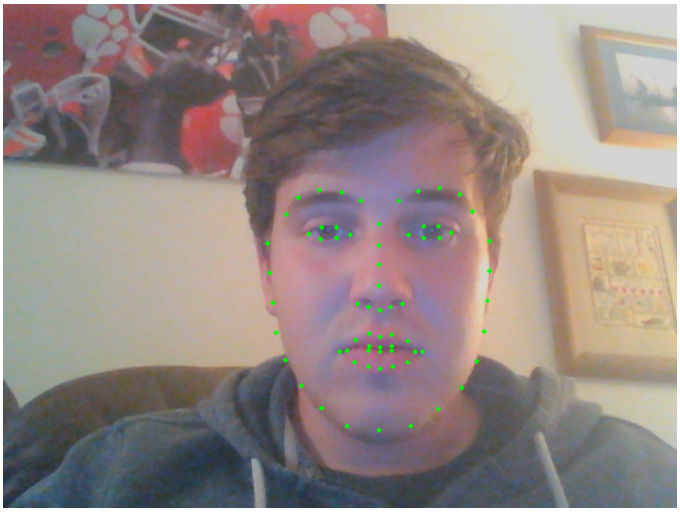

Figure 6: Facial recognition performed by Dlib library model

Figure 7: Dlib detection and glasses drawn with no facial rotation

4 CONCLUSIONS

Using a combination of facial pose estimation and facial detection models, I was able to implement a program that drew glasses on a user's face. The translation and rotation vectors found using the pose estimation allowed me to have the glasses rotate as the user's face rotated around the x-axis. An improvement would be to have the glasses also rotate around the y and z axes, better known as yaw and roll. This would require implementing something similar to image rectification if I were to use the same image of glasses. Another possibility would be to use a different method of drawing the glasses, such as animating a more realistic 3D model using OpenGL. Worked pretty well Animation was not great. could do better Future work using this method would use inverse rectification perhaps

Figure 8: Dlib detection and glasses drawn with facial pitch

Could try to use rvecs and tvecs to properly roatate the glasses image.

REFERENCES

[1] Italo José. 2021. Facial mapping (landmarks) with dlib + python. *Medium* (Jun 2021). https://towardsdatascience.com/facial-mapping-landmarks-with-dlib-python-160abcf7d672

[2] Satya Mallick. 2021. Head pose estimation using opencv and Dlib: Learnopencv. *LearnOpenCV* (May 2021). https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/