Understanding How Face Detection Works for Filter/Lens Creation

Blessing Ajibero
Louisiana State University
bajibe1@lsu.edu

Abstract

.

In recent times, researchers have gained interests in face recognition and detection systems due to its wide range of application. The reaction on human faces aids context based communication. Social network such as Snapchat and Instagram have recently adapted the use of lens and filter features to enhance social interactions online. This paper aims to show and explain one of many methods used in creating these features using OpenCv, Dlib and Python.

1. Introduction

The lenses and filter features on Social media applications such as snapchat and Instagram is one innovation that continues to stir interest among social media users. It has unlimited creative possibilities such as image manipulation, video effect enhancement, gaming and even show real time geo data. Filters add or reduce color from an image or video while lenses manipulates the image into something new or different through a camera.

Face detection is the foundation factor in constructing lens and filters. This technology dates far back as the 1980s but has in recent times has attracted a lot of research because of its wide range of application particularly in the area of computer vision. A lot of techniques have been developed and improved since its inception [1]. However we will be utilizing the Dlib library for face detection because it contains machine learning algorithms that accurately spots the facial region along with its features (i.e. eyes, mouth, nose and eyebrows). This paper explains and showcases a method used to create lenses and filters using Python, OpenCV and Dlib.

There are five main steps in which the program must undergo in order to achieve the desired results. They include (as shown in Figure 1): 1) Connect to a camera or webcam using OpenCV, an open source library that provides the tools needed to solve computer vision problems [6], 2) Image/video Pre-Processing 3) Use Dlib's landmark face detection to mark out facial features 4) Select the specific features needed to create the desired effect 5) Create Mask of feature that needs to be overlaid or duplicate (i.e. the creative process) and 6) Display the results.

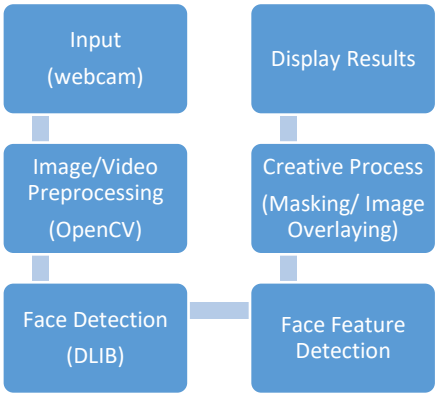

Figure 1: Process Overview

2. Related Work

There are a lot of techniques that have been developed solely for facial detection. First, Haar Cascade Classifier is based on Viola Jones detection algorithm, which achieves frontal face detection with some false positive rate [2] [10]. . It uses Haar- like features which uses the change in contrast values between adjacent rectangular groups of pixel rather than the intensity values of a single pixel [3]. The Cascade classifier requires training of the integral images before it can accurately detect human facial features such as the mouth, eyes, and nose. [4] mentions the Adaboost learning is able to select most effective features from a large feature pool to form a strong classifier and then use cascading techniques to detect a face or faces in an image.

In addition, Satyanadh Gundimada and Vijayan Asari [15] proposed a rotation and scale invariant face detection algorithm based on the texture of a human face. J. J. de Dios and N. Garcia [16] Proposed YCgCr, is described and applied for face detection.

Furthermore, [6]-[9] proposed HOG feature as a technique used for facial detection. It is explained as a feature descriptor that handles images by dividing it into cells and for each pixel within the cell, a histogram of gradients direction is assembled [5]. HOG features are

calculated by taking orientation histograms of edge intensity in a local region. Studies show that HOG combined with SVM, a model that finds a hyperplane that fits between two classes, can be used for face detection models [11] . It is represented by the equation below:

$$g(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0,$$

Where **w** is a matrix containing support vector and ω_0 represents the bias. [11] proposed an effective face detection method based on Two-Dimensional Basic Component Analysis (PCA) combined with Support Vector Machine (SVM). [12] put forward the proposition of a cascaded face detection method based on the histograms of oriented gradients (HOG), using different features and classifiers to step-out the non-faces.

The most commonly used techniques for face detection have been shown to be Haar Cascade and HOG feature (Histogram of Oriented Gradients) + SVM (Support Vector Machine) [13]. For my project, I implemented Dlib C++ library for face detection. The library is built on a combination of HOG and SVM. I chose this library mainly because HOG has a higher accuracy for face detection than Haar Cascade Classifier.

3. Approach

First, using the OpenCV library in Python we connect to a webcam. This webcam will serve as the input. Afterwards the input undergoes Image or video Pre-Processing. This aspect involves resizing the video and converting it from color to grayscale. This process is important in order for the system to run faster and much more efficiently. This process also helps in reduce space in memory and minimize the computational complexity that comes with colors. Converting RGB input to grayscale lessens the number of pixels that need to be processed thereby causing the system to perform better.

The next step is to detect a face by implementing some functions in the Dlib library. For testing purposes, I surrounded the face with a bounding rectangle (as shown in **figure 3.1**). I then utilized a pre-trained Dlib landmark or shape predictor to pin point specific facial features. These features include eyes, eyebrows, nose, lips and jawline. Dlib's landmark facial detector provides in estimation the location of 68 coordinates that map the facial features of a person's face as shown in **figure 3.2.** The range of numbers in figure 3.2 corresponds to the specific points in **figure 3.3**. For example the range of numbers between 36 and 41 represents the left eye and 0 to 16 represents the chin area. With this mapping, one can select a specific region on the face to work with for a filter.

After the face has been detected and mapped, the creative process begins. I chose all 68 coordinates to create a face animated effect for the lens inspired by the bullet dodging scene in the movie, the Matrix(see figure 3.4 for results). I stored the coordinates of my face as it moves in a list and placed my duplicated mask on the coordinates stored in the list. Also, for the filter I chose the ranges for both eyes and created an overlay of a cartoon eye masked on the eye region. The result off the program is then displayed once it compiles. (See figure 3.4 and 3.5 for results)

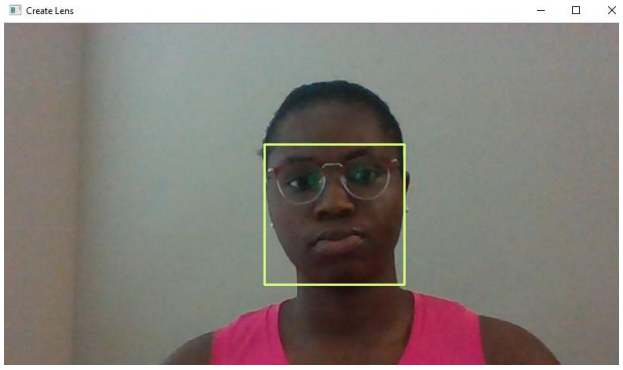

Figure 3.1: Result of Dlib's front face detector with bounding rectangles.

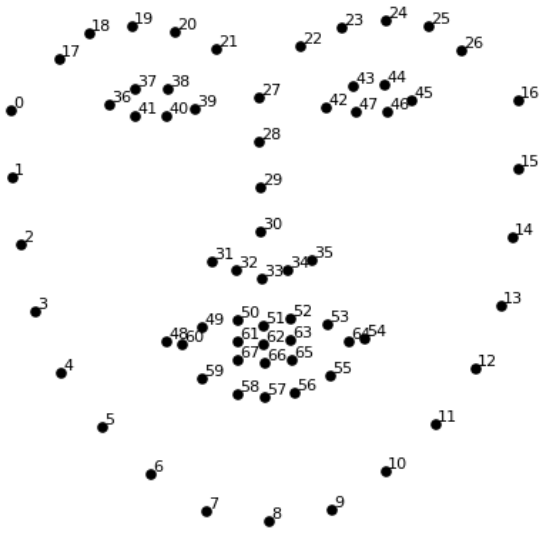

Figure 3.2: Dlib Shape Predictor Model

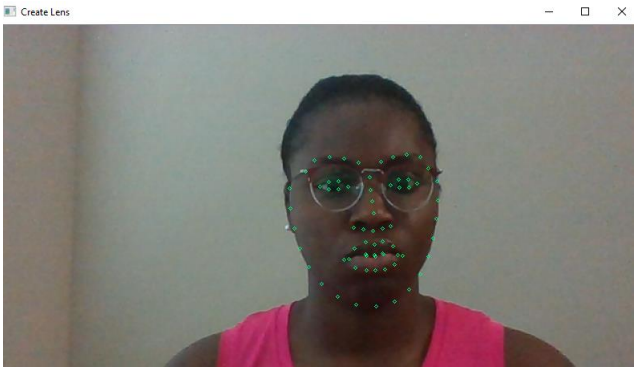

Figure 3.3: Dlib's Face 68 coordinates Landmark Detector compiled using OpenCv and Python. Note the similarities and mapping between Figure 3.2

4. Drawbacks

The model applied does not detect the forehead due to the 68 coordinate's limitation. I was unable to apply the Dlib predictor for my proposed third filter because I found it difficult to target the forehead region using an automated process.

Furthermore the library has little to no accuracy in detecting a face from the side view or up view. The subject has to be totally facing the camera.

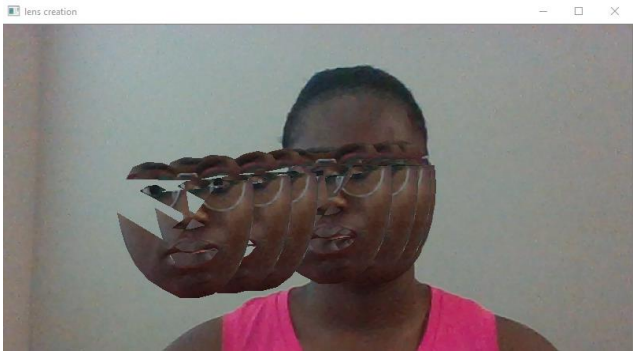

Figure 3.4: Results from masking previous head movement on a webcam input

Figure 3.5: Results of the filter overlay on a webcam input

6. Conclusion

Creating lens or filters using a combination of OpenCv and Dlib through Python implementation is attainable. Making use of Dlib's pre-trained model certainly helps speed up the computing process. The methods executed provides an idea of how social media companies like Snapchat and Instagram create cool filters and lenses.

5. Future Works

Creating a diagnosis supporting system for eye diseases by using the model for eye detection coupled with coloration. A system that will help doctors confirm if their diagnosis of a particular eye disease is accurate based on a learned machine learning algorithm. Also, considering a case study that compares the HOV based DLIB implementation with the CNN (Convolution Neural Network) based. More improvement on the methods and algorithms put in place.

References

[1] M. V. Choudhari, M. S. Devi, and P. Bajaj. 2011. Face and facial feature detection. *In Proceedings of the International Conference & Workshop on Emerging Trends in Technology (ICWET '11)*. Association for Computing Machinery, New York, NY, USA, 686–689.

[2] Viola, P. and Jones, *M.* Rapid object detection using boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[3] Phillip Ian Wilson and John Fernandez. *2006.* Facial feature detection using Haar classifiers. *J. Comput. Sci. Coll. 21, 4 (April 2006), 127–133.*

[4] Zhen Lei, Stan Z. Li, Face Recognition Models: Computational Approaches, Editor(s): James D. Wright, International Encyclopedia of the Social & Behavioral Sciences (Second Edition), Elsevier, 2015, Pages 658-662, ISBN 9780080970875

[5] Batoul Husain Bani Hashem and Tomoko Ozeki. 2015. Pedestrian Detection by Using FAST-HOG Features. *In Proceedings of the 3rd International Conference on Human-Agent Interaction (HAI '15).* Association for Computing Machinery, New York, NY, USA, 277–278.

[6] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. *2012. Realtime Computer Vision with OpenCV. Queue10,4 (April2012),40–56.*

[7] X. Xu, C. Quan and F. Ren, "Facial expression recognition based on Gabor Wavelet transform and Histogram of Oriented Gradients," *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, Beijing, 2015, pp. 2117-2122

[8] Yanwei Pang, Yuan Yuan, Xuelong Li, and Jing Pan. 2011. Efficient HOG human detection. *Signal Process. 91, 4 (April 2011), 773–781.*

[9] Akanksha Das, Ravi Kant Kumar, and Dakshina Ranjan Kisku. 2016. Heterogeneous Face Detection. *In Proceedings of the International Conference on Internet of things and Cloud Computing (ICC '16).* Association for Computing Machinery, New York, NY, USA, Article 32, 1–6.

[10] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, 2002 vol. 57, no. 2, pp. 137- 154.

[11] Zhang, X, Pu, J., Huang X., "Face Detection Based on Two-Dimensional Principal Component Analysis and Support Vector Machine", *Mechatronics and Automation Proceedings of the 2006 IEEE International Conference on IEEE*, 1488- 1492, (2006)

[12] Yang H.C, Xu A.W., "Cascade Face Detection Based on Histograms of Gradients and Support Vector Machine", *Parallel, Grid, sCloud, and Internet Computing (3PGCIC)*, 2015 10th International Conference on IEEE, 2015. 17

[13] Emine Cengil, Ahmet Cinar. "Comparison of HOG (Histogram of Oriented Gradients) and Haar Cascade Algorithms with A Convolutional Neural Network Based Face Detection Approach." *International Journal of Advance Research, Ideas and Innovations in Technology 3.5* (2017).

[14] Chauhan, M., & Sakle, M. (2014). Study & analysis of different face detection techniques. International Journal of Computer Science and Information Technologies, 5(2), 1615-1618.

[15] Satyanadh Gundimada and Vijayan Asari, "Face detection technique based on rotation invariant wavelet features," International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004., Las Vegas, NV, USA, 2004, pp. 157-158 Vol.2.

[16] J. J. de Dios and N. Garcia, "Face detection based on a new color space YCgCr," Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429), Barcelona, Spain, 2003, pp. III-909.