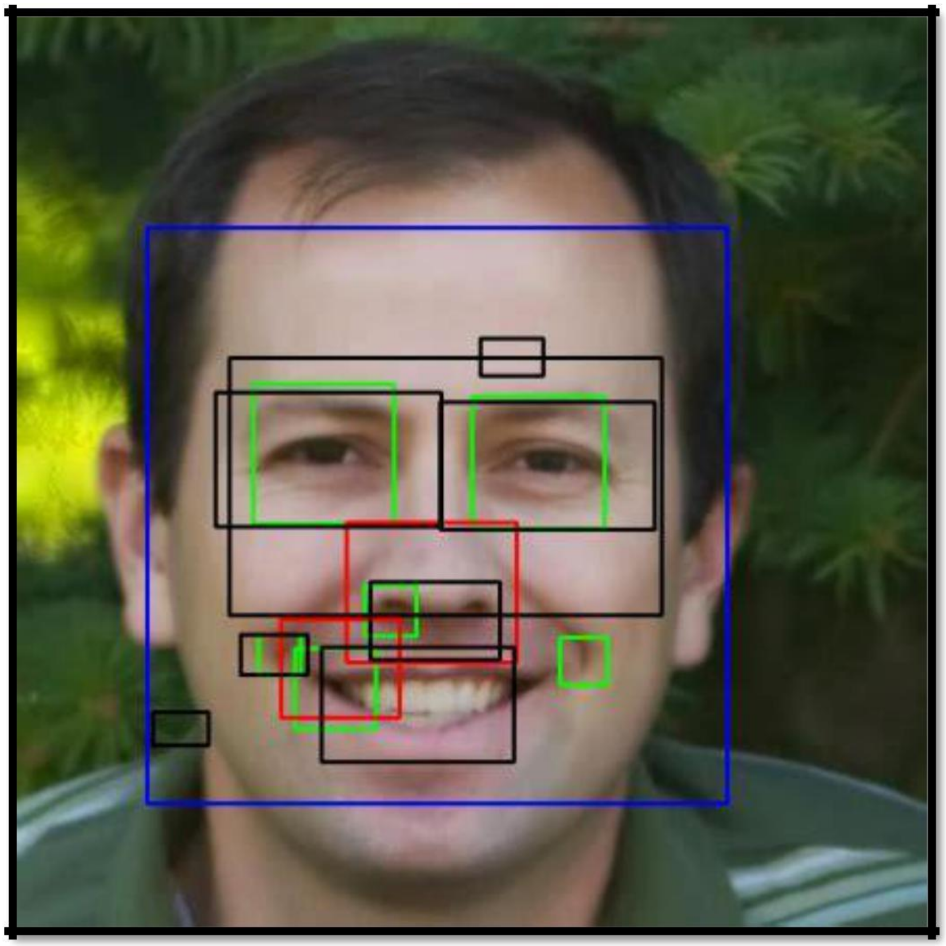
FINAL PROJECT REPORT



Smile Detection

Gokul Balagopal – PSID 1590661

Artificial Intelligence COSC 6368 – Spring 2018

Professor – Dr. Ricardo Vilalta May 7, 2018

PART 1- Survey

1. Motivation

In today's day and age, cellphones are ubiquitous. It is a commonplace to find tech-savvy youth indulging in taking self-portraits and uploading on social media. This as a phenomenon is one of the many reasons fueling smartphone giants like Samsung, Apple and others to compete in their tech for photo capture. Every year it is not uncommon to find eye catching updates in their products particularly to accommodate photo capture. This coupled with artificial intelligence has given a new dimension in range of products pertaining to smartphone industry. For ex. FaceID in Apple's iPhone X is powered by an on-device AI processor along with infrared face detecting schemes. Another chic feature is the Animoji which basically animates emoji with face expression. This combination of face detection and AI is not limited to smartphone industry.

High level security systems are increasingly using this feature for efficient security. Iris scan a common example of biometric security. Motion capture technology which uses dots on an actors' face to capture the facial expression and render a digital format is also hugely dependent on face detection. Gender classification for image database systems, CC TV systems, human lie detectors, digital cosmetics are a few examples which are dependent on face detecting algorithms. In 2016, MasterCard launched a selfie pay app which enables customers to confirm payments with their camera. Google's image search option or Facebooks automatic tagging of people feature is something we take for granted without realizing its dependence on face recognizing algorithms.

If face detection is authorizing payments and unlocking phones, are chances for it to be compromised? One story from 2017 regarding failure of Apple's FaceID to distinguish between two Chinese users in unlocking phones made quite a story and eventually accusing it of racism. Stakes are usually high with face detecting systems which requires them to robust. This is quite challenging and complex as the facial profile, facial appearance, camera limitations and lighting may vary. In addition to this, the feature space representation of captured content is highly nonlinear and complex which may be a drawback.

Now that we have defined the scope of field of interest, we decided to work on smile detection a subset of face detection as a final project for this course.

Overview:

This project implements smile detection using OpenCV library. It faithfully detects features like face (blue box), nose (green box), eyes (green box) and mouth (blue box). OpenCV has functions which can filter out features in an image. Initially the color image is downscaled to grey scale where the detection of features takes place. Each detected feature is encompassed by a rectangle. This grey scale image with encompassed features are again upscaled by same proportion to get encompassed color image. The filters in OpenCV are pretrained.

The fundamental algorithm used in face detection is described briefly in the next section. The section after describes in detail the working of code.

PART 2- Development

2. Algorithm

2.1 Viola-Jones Algorithm

To understand how the smile detection works, we must understand first how face detection works. The Viola-Jones algorithm is used to detect face. For it to detect properly, a full front upright face is needed. This algorithm, in general, has high true-positive rate, works well in real time and does detection but not recognition. The algorithm has the following stages:

1. Haar Feature selection
2. Creating an integral image
3. Adaboost training
4. Cascading classifiers

2.2 Haar-Like Features

First, the training images are converted into grayscale for simplicity because there is less data to process. This algorithm uses Haar-Like features to compare and extract the features of a face from an image. For example, eyes, nose, the area between eyebrow and forehead, lips are a few features unique to a face which are Haar-Like features. Figure 1 shows different Haar-Like features used in this algorithm.

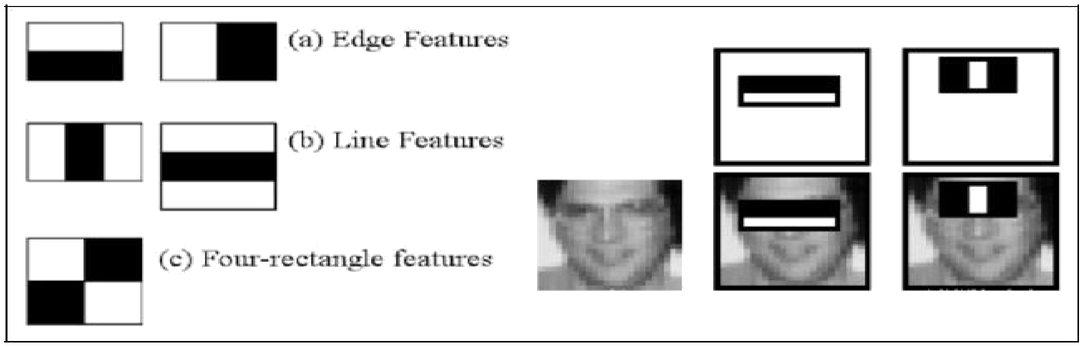


Figure 1: Haar-Like Features

2.3 Integral Image

The Haar-Like features are in rectangular form. These features are obtained by subtracting light area from dark areas in the image. These areas are the summation of all gray values of the pixels within the rectangle. The integral images have summed area table instead of direct gray value of the pixels, as shown in Figure 2 following.

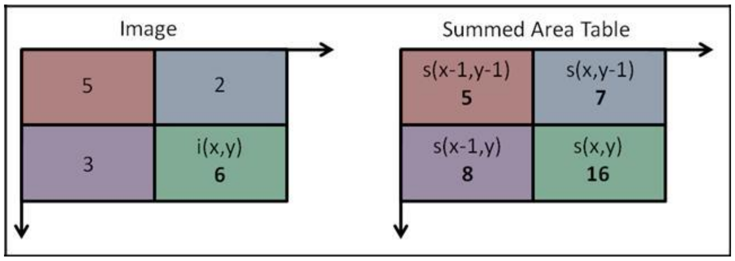


Figure 2: Integral Image

The integral image reduces the number of calculations in order to detect a feature with 4-point calculation.

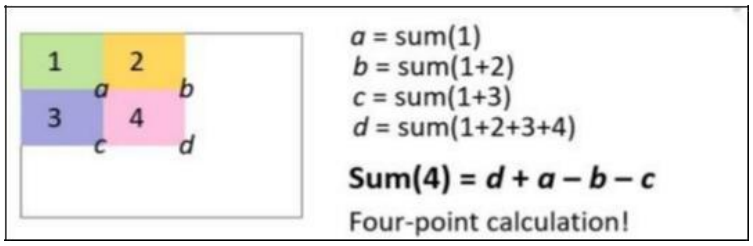


Figure 3: 4-Point Calculation

2.4 Training Classifiers

In the training, the image is shrunk to a 24 x 24 pixels size so that it is faster to process. Then the features are identified in an image and threshold is calculated to determine whether the feature fits the criteria of a face or not. In the actual detection, original sized images are used, and the Haar-Like features are scaled up to the equivalent image sizes.

Two types of images are provided –

1. Face Images
2. Non-Face Images

The algorithm is trained by using images labeled as face images. Here, algorithm learns which features are common in any face, and it keeps those features. After that, the algorithm is trained using non-face images, where it learns about features in non-face images, compares them with previously learnt features of face images, and discards common features which give false positives.

2.5 Adaptive Boosting

A feature on its own can be weak and have less accuracy of detecting a face. But, if many of such weak classifiers are combined, the accuracy of the face detection increases.

The Figure 4 shows how using a cascade of features gives a boost to accuracy of face detection.

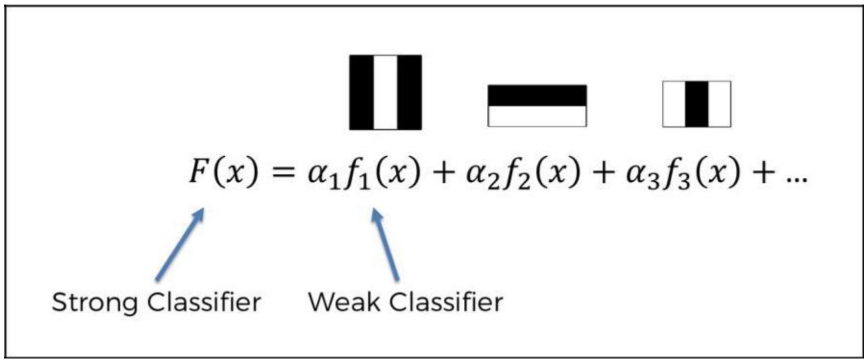

$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$
Strong Classifier
Weak Classifier

Figure 4: Adaptive Boosting & Cascade of Features

2.6 Smile Detection

For smile detection, we must extend the use of the Viola-Jones algorithm for face detection and train it to detect smile by adding a new set of features. Once, a face is identified in an image, next part is mouth detection.

Different smile-images and non-smile images must be provided to the algorithm for it to learn to detect smile.

3. Implementation

3.1 Loading the cascades:

Cascades for each of the features which are stored in XML format are loaded . First we classify an object with a pretrained data set of Haar-like facial features using OpenCV. *Cascade* classifier classifies the frontal face, eyes, nose, and smile.

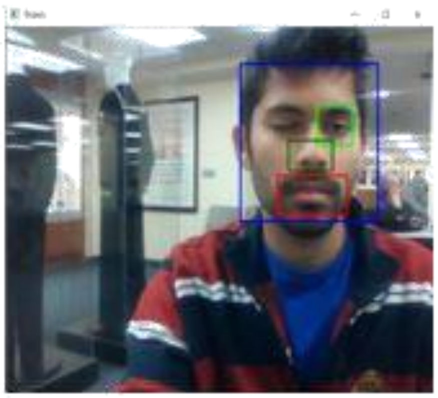
3.2 Facial and Feature Detection

- Video recorded by the webcam will be divided into frames. The input to the *detect* function will be the gray scale of the selected frame and the selected frame (frames are used for detecting objects /elements in a video and not the entire video).
- Initially we use frontal face cascade from Haar-cascade for loading the classes that are used for detecting the face.
- We use *detect multiscale* function to find the facial features by taking the grayscale of the actual frame. The *detect multiscale* method takes scale Factor, minimum-neighbors as the input. A scale factor of 1.1 corresponds to a decrease of 10% in the input (by how much the size of the image is going to be reduced).
- Minimum-neighbors means that for a zone of pixels to be accepted we need to have a certain number of its neighbor zones accepted. Coordinates (x,y) are the top left coordinates of the face threshold box. w and h are the width and height, which is used for defining the threshold window used for detecting the face.

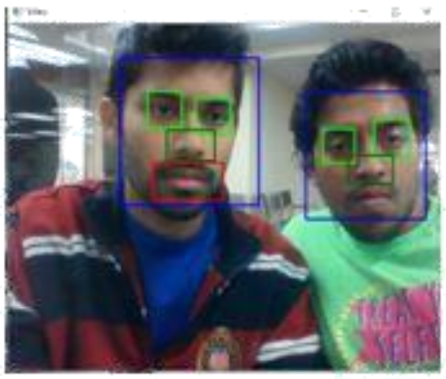
- The frame (selected color frame, and not the gray one) where the threshold box is to be drawn, the coordinates of the top left and the bottom right corners of threshold box, color of threshold box and the thickness of the threshold box are given to the *rectangle* class of OpenCV. The top left corners coordinates are(x,y) and the bottom right corners coordinates are(x+w, y+h).
- The portion of the frame where the face is detected is used as the region of interest for finding the features like eyes, nose and smile. The gray scale image of that portion of the frame is also used.
- The above steps are repeated for detecting the eyes, nose and smile. They use the region of interest and not the entire frame as the reference.
- Here the parameters for the *detect multiscale* function are adjusted by trial and error for different features.
- The color of the frames is adjusted based on our interest.
- After the providing windows for all the features, the output after detecting all the features will be saved as a jpg file using the *imwrite* class of OpenCV.

3.3 Activating the webcam

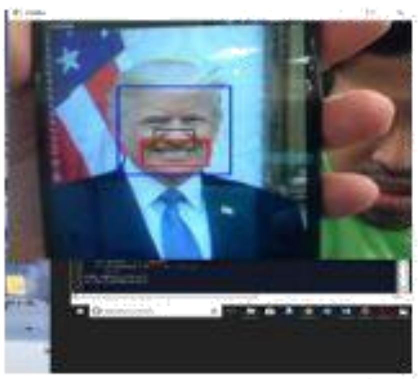
- Webcam is used to input images for detection. The webcam is turned on by the *Video Capture* class. If we are using an inbuilt webcam (laptop) we must provide '0' as the input, otherwise when using an externally connected webcam we provide '1' as the input.
- For running the video continuously, we use an infinite loop.
- The frames are read using the *capture read* function and the gray scale image is obtained by using the *cvtColor* class. The captured frame as well as its gray scale image is given to the *detect* function defined earlier.
- For quitting the video, we need to press 'q'.



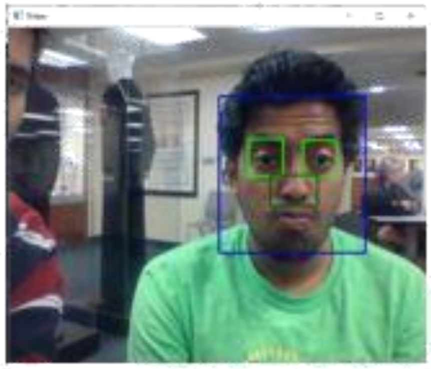
1. Winking



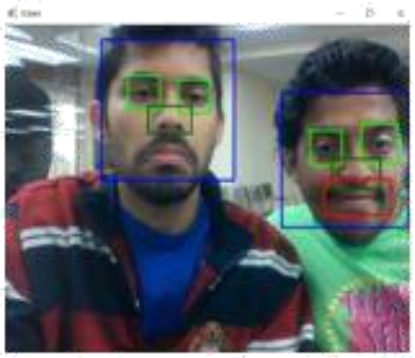
2. Wrongly detects smile



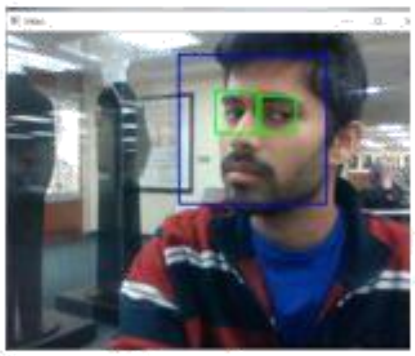
3. Eyes not detected (phone)



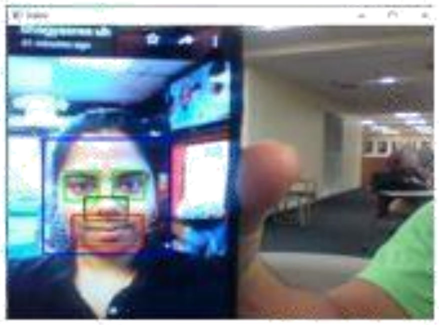
4. no smile, no detection



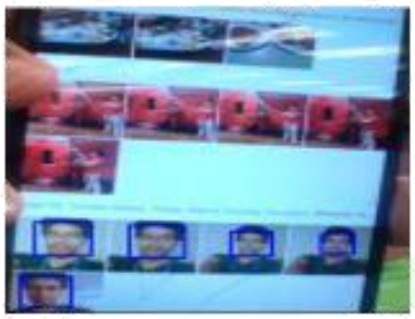
5. Smile correctly detected



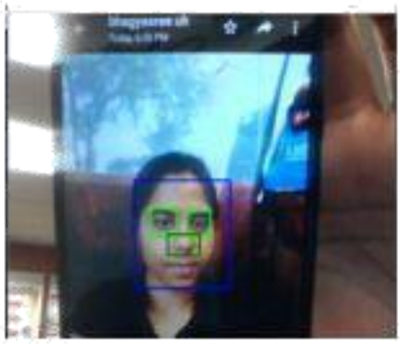
6. Face Tilted, smile not detected



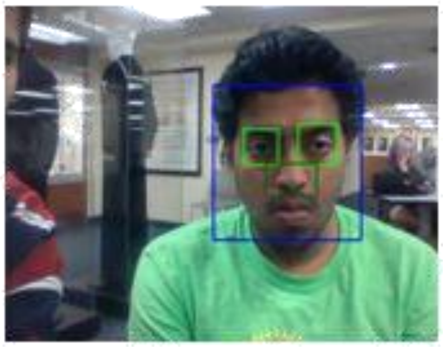
7. Smile detected



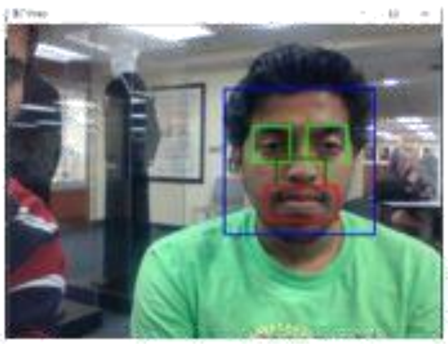
8. Multiple face detection(phone)



9. No smile, no detection



10. no smile, no detection



11. Smile detected

Conclusion:

To summarize, we implemented a facial feature detector using OpenCV tools to detect face, smile, nose and eyes. Voila-Jones algorithm was used along with other cascades to implement this. The model provides an acceptable detection of features but is not short of drawbacks.

For example, in fig 2 smile is not detected properly possibly because of the beard. In fig 3 eyes are not detected. Model fails to detect nose and smile because the head is tilted, and the algorithm mandates an upright frontal face as in shown in fig 6. In most of the cases, smile is detected acceptably and does a decent job for images shown via smartphone. Model can also predict features on multiple individuals as shown in fig 5.

Given the performance of the model there is scope for improvements. Training for different orientation of face can remove the upright front face condition. Model can be tuned to perform for different facial expressions and lighting conditions.

This face detection framework can be improved to perform higher functions. The output of this framework can be used in conjunction with a neural network for identification which can be used as a security feature. Higher level abstraction of features in images can help the model perform better even in uneven conditions like lighting, camera defects etc. Future developments in this face detecting framework can be extended to medical diagnosis, instantaneous ID of people, cosmetic surgeries, real time face detection, social network services and many more.

References

[1] computersciencesource.wordpress.com

[2] www.cs.northwestern.edu

[3] web.stanford.edu

[4] M. Pantic, L.J.M. Rothkrantz
Automatic analysis of facial expressions: the state of the art
IEEE Trans. Pattern Anal. Mach. Intell., 22 (12) (2000), pp. 1424-1445

[5] C. Shan
Smile detection by boosting pixel differences
IEEE Trans. Image Process., 21 (1) (2012), pp. 431-436

[6] J. Whitehill, G. Littlewort, I. Fasel, M. Bartlett, J. Movellan
Toward practical smile detection
IEEE Trans. Pattern Anal. Mach. Intell., 31 (11) (2009), pp. 2106-2111

[7] www.youtube.com