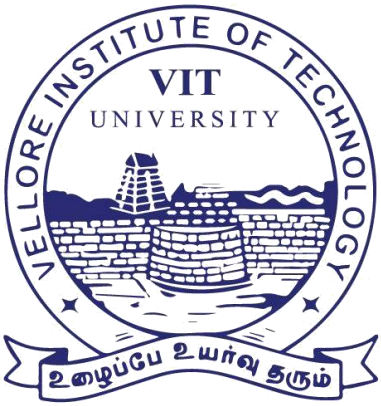


VIT®

UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

FACE DETECTION

CSE4019

Image Processing J components

Final Report

Siddharth Shailendra 16BCB0129

Vaasu Gupta 16BCB0062

Ashish Singh Kalakoti 16BCB0128

Abstract

The cascades themselves are just a bunch of XML files that contain OpenCV data used to detect objects

OpenCV comes with a number of built-in cascades for detecting everything from faces to eyes to hands and legs.There are even cascades for non-human things.

Use of Haar like features

Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region.The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object(the face in this Haar-like features are digital image features used in object recognition.They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector.[1]

Historically, working with only image intensities(i.e.,the RGB pixel values at each and every pixel of image) made the task of feature calculation computationally expensive . A publication by Papageorgiouetal discussed working with an alternate features based on Haar wavelets instead of the usual image intensities. ViolaandJones adapted the idea of using Haar wavelets and developed theso-called Haar-like features.A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window ,sums up the pixel intensities in each region and calculates the difference between these sums.This difference is then used to categorize subsections of an image.For example,let us say we havean image database with human faces.It is a common observation that among all faces the region of the eyes is darker than the region of the cheeks . There for a common case

How haar cascades work?

all thanks to the Viola-Jones algorithm for face detection, using Haar-based cascade classifiers.

It's basically a machine learning algorithm that uses a bunch of images of faces and non-faces to train a classifier that can later be used to detect faces in realtime.

The algorithm implemented in OpenCV can also be used to detect other things, as long as you have the right classifiers. This OpenCV distribution came with classifiers for eyes, upper body, hands, frontal face and profile face.

The way we did it was inspired by all of these tutorials, with some minor modifications and optimizations.

1. Negatives

This is where we gather about 1000 images of non-faces.

2. Positives

This is where we gather about 1000 images of faces.

2a. Pictures

This is where we take pictures of faces. We don't need 1000 of them. Somewhere between 15 and 20 should be enough.

2b. Process

This is where we use a Processing script to read the images and mark where the object is. Since we used high-contrast and a white background, it's pretty easy to get an initial guess by just keeping track of the min/max x- and y- positions of dark pixels. What is important here is to make sure that the aspect ratio of all of the marked objects is the same.

3. Train the Cascade

This is where we train a Haar Cascade Classifier using another OpenCV utility. Armed with about 1000 negative images and 2000 positive images, we can run this command to start training:

**opencv_traincascade -data outputDirectory **

**-vec cropped.vec **

**-bg negativeImageDirectory/negatives.txt **

**-numPos 1000 -numNeg 600 -numStages 20 \ -precalcValBufSize 1024 -precalcIdxBufSize 1024 \ -featureType HAAR **

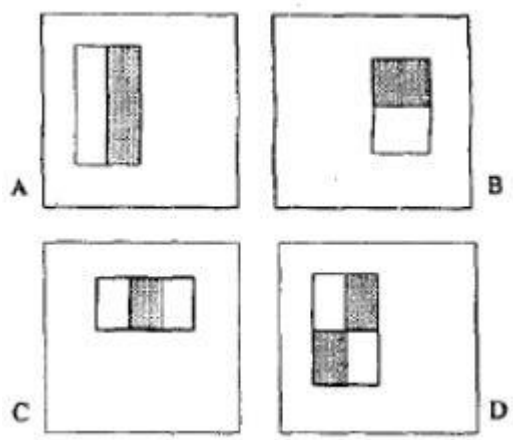
**-minHitRate 0.995 -maxFalseAlarmRate 0.5 **

-w 48 -h 48

If all goes well, a cascade.xml file should show up in the outputDirectory after a couple of hours (or days).

A script can be written that automates most of this process.

In general, three kinds of features are used in which the value of a two rectangular features is the difference sum of the pixels within two rectangular regions. These regions have same shape and size and are horizontally or vertically adjacent. Where as in the three rectangular features are computed by taking the sum of two outside rectangles and then subtracted with the sum in a center rectangle. Moreover, in the four rectangles feature computes the difference between diagonal pairs of rectangles (Viola P. & Jones M.; 2001).



Example rectangle features use in Haar-cascade. The sum of pixels in the white rectangles is subtracted for the sum of the pixels in the grey rectangles. Here A and B are two rectangle feature, and C and D are three and four rectangle feature (Viola P. & Jones M.; 2001).

Classification learning process requires a set of positive and negative images for training and a set of features are selected using Ada-boost for training the classifier. To increase the learning performance of the algorithm (which is sometime called as weak learner), the Ada-boost algorithm is used. Ad-boost provides guarantees in several procedures.

Code

Studying the code segment wise

1)

```
imagePath=sys.argv[1]
cascPath=sys.argv[2]
```

First pass in the image and cascade names as command-line arguments.We'll use the Abba image as we last he default cascade for detecting faces provided by OpenCV.

2)

```
faceCascade=cv2.CascadeClassifier(cascPath)
```

Now we create the cascade and initialize it with our face cascade. This loads the face cascade into memorys of it and it is ready for use.The cascade is just an XML file that contains the data to detect faces.

3)

```
faces=faceCascade.detectMultiScale(
   gray,
   scaleFactor=1.1,
   minNeighbors=5,
   minSize=(30,30),
   flags=cv2.cv.CV_HAAR_SCALE_IMAGE
)
```

Thisfunctiondetectstheactualface–andisthekeypartofourcode,solet'sgo overtheoptions.

1. ThedetectMultiScalefunctionisageneralfunctionthatdetectsobjects.Since wearecallingitonthefacecascade,that'swhatitdetects.Thefirstoptionis thegrayscaleimage.
2. These condisthescaleFactor.Sincesomefacesmaybeclosertothecamera, theywouldappearbiggerthanthosefacesintheback.Thescalefactor compensatesforthis.
3. Thedetectionalgorithmusesamovingwindowtodetect objects.minNeighborsdefineshowmanyobjectsaredetectednearthecurent onebeforeitdeclaresthefacefound.minSize,meanwhile,givesthesizeof eachwindow.

4)

```
print"Found{0}faces!".format(len(faces))

#Drawarectanglearoundthefaces
for(x,y,w,h)infaces:

    cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)
```

Thisfunctionreturns4values:thexandylocationoftherectangle,andthe rectangle's width and height(w,h).

5)

```
cv2.imshow("Facesfound",image)
    cv2.waitKey(0)
```

Intheend,wedisplaytheimage,andwaitfortheusertopressakey

Explaining The Python Code:

We are using "haarcascade_frontalface_default" and "haarcascade_eye" xml files.

Make different variables for each cascade like

face_cascade

eye_cascade

Including webcam through cap=cv2.VideoCapture(1)

Read from cam using ret,img=cap.read()

Convert to grayscale using

gray = cv2.cvtColor(img, cv2.COLOR_BG2GRAY)

faces= face_cascade.detectMultiScle(gray,1.3,5) //the values have been changed acc to efficiency

for(x,y,w,h) in faces: //w and h represent width and height respectively

cv2.rectangle(img , (x,y), (x+w, y+h), (255,0,0) , 2)

/*draw rectangle on "img" strating from (x,y) to (x+w,y+h) and the color is (255,0,0) i.e. blue and 2 is the line width*/

roi_gray = gray[y:y+h, x:x+w] /*Defines the region inside the face rectangle*/

/*Nobody wants eyes outside the face. But it could likely be eyeballs. Eyes have features like eyelashes, eyebrows, eyelids which cant be outside of face*/

roi_color = img[y:y+h, x:x+w]

eyes = eye_cascade.detectMultiScale(roi_gray)

/*detect eyes against roi_gray*/

for (ex,ey,ew,eh) in eyes:

cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh), (0,255,0), 2)

cv2.imshow('img',img)

k=cv2.waitKey(30) & 0xff

if k == 27:

break

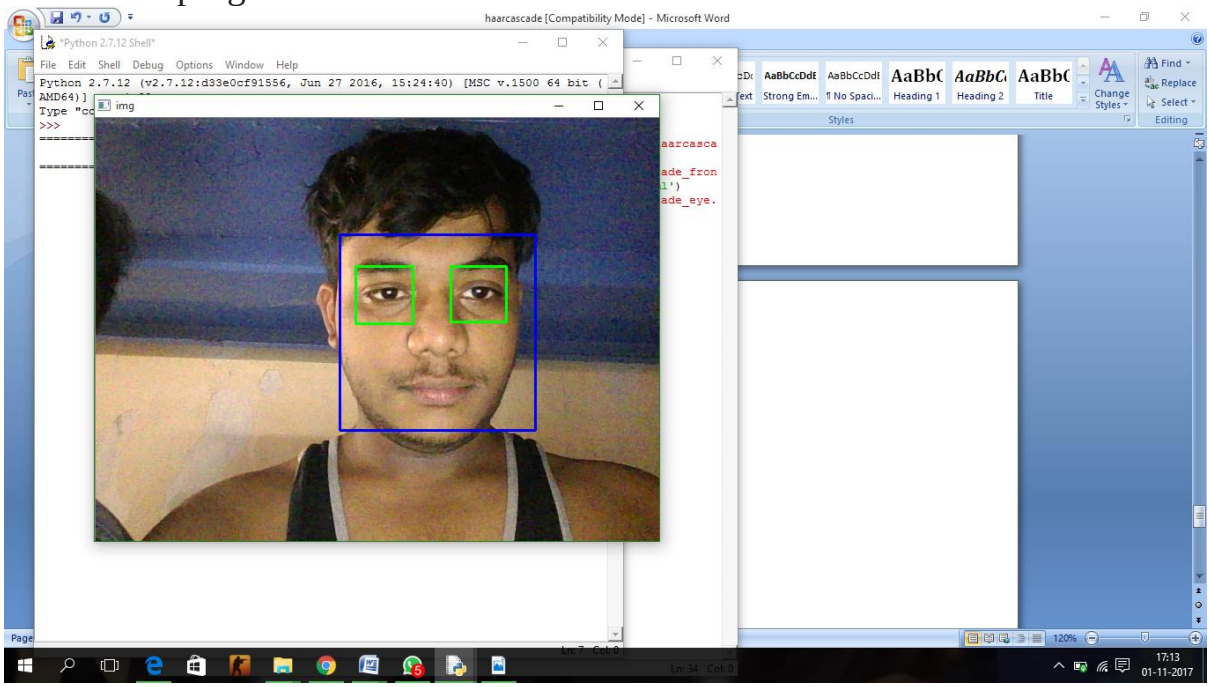
/*So we can press the escape key*/

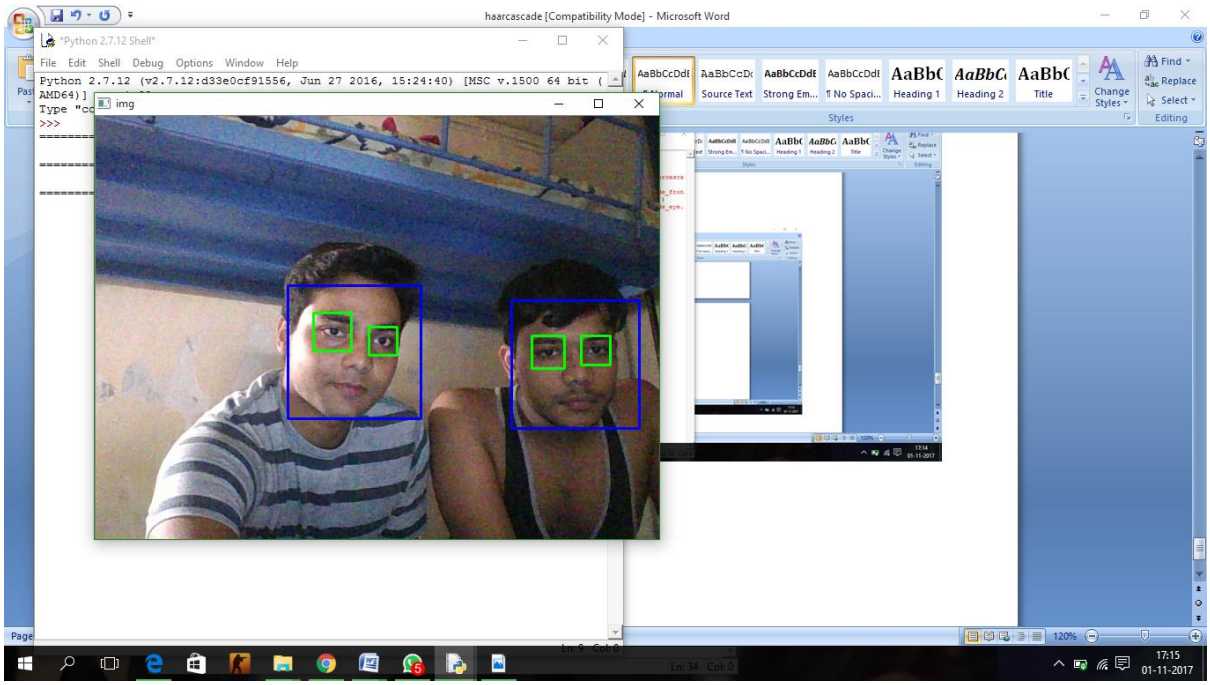
cap.release()

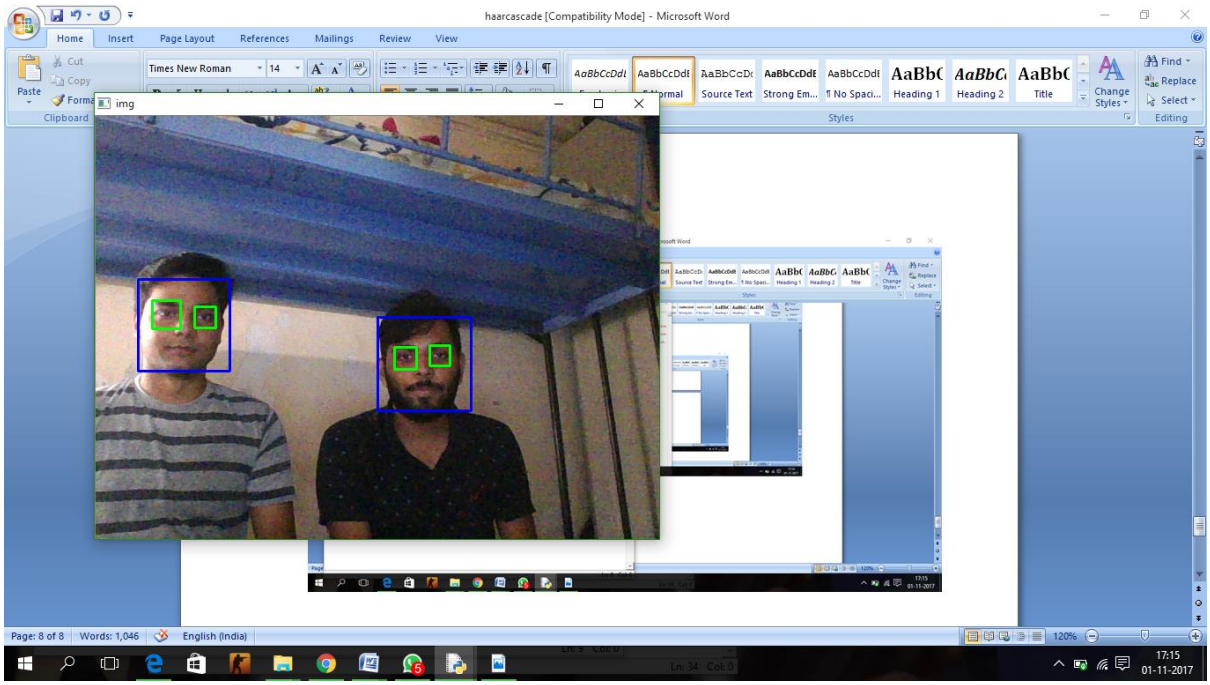
cv2.destroyAllWindows()

Conclusion

We ran the program and result are as follows







References:

[1] Kumar R. & Bindu A. (2006) An Efficient Skin Illumination Compensation Model for Efficient Face Detection *IEEE Industrial Electronics, IECON 2006 – 32nd Annual Conference* 3444 – 3449

[2] Viola P. & Jones M (2001) Rapid Object Detection using a Boosted Cascade of Simple Features *Computer Vision and Pattern Recognition Proceedings of the 2001 IEEE Computer Society Conference* vol. 1: I-511 – I-518

[3] Sialat M., Khlifat N., Bremond F. & Hamrouni K. (2009) People detection in complex scene using a cascade of Boosted classifiers based on Haar-like-features *Intelligent Vehicles Symposium, 2009 IEEE* 83 – 87

[4] Naotoshi Seo's- http://note.sonots.com/SciSoftware/haartraining.html#v6f077ba

[5] The Performance of the Haar Cascade Classifiers Applied to the Face and Eyes Detection

[6] International Journal of Computer Trends and Technology
(IJCTT)
A review on Face Detection and
study of Viola Jones method
Monali Chaudhari, Shanta sondur,Gauresh Vanjare
http://www.ijcttjournal.org/2015/Volume25/number-1/IJCTT-V25P110.pdf

[7] https://github.com/opencv/opencv/tree/master/data/haarcascades