**<https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177>**

[**http://somideolaoye.com/blog/opencv-haar-cascade-face-detection/**](http://somideolaoye.com/blog/opencv-haar-cascade-face-detection/)

[**https://pythonprogramming.net/haar-cascade-face-eye-detection-python-opencv-tutorial/**](https://pythonprogramming.net/haar-cascade-face-eye-detection-python-opencv-tutorial/)

[**https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/**](https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/)

[**http://alereimondo.no-ip.org/OpenCV/34**](http://alereimondo.no-ip.org/OpenCV/34) **(OTHER HAARCASCADES)**

**Face & Eye** **detection using Haar Cascade method.**

Face detection is a computer technology that determines the locations and sizes of human faces in images. It detects facial features and ignores anything else, such as buildings, trees, and bodies. These can be in real time from a video camera or from photographs.

An example where this technology is used are in airport security systems. In order to recognize a face, the camera software must first detect it and identify the features before making an identification. Likewise, when Facebook makes tagging suggestions to identify people in photos it must first locate the face. On social media apps like Snapchat, face detection is required to augment reality which allows users to virtually wear dog face masks using fancy filters. Another use of face detection is in smartphone face ID security.

Before we continue, we must differentiate between face recognition and face detection. They are not the same, but one depends on the other. In this case face recognition needs face detection for making an identification to “recognize” a face. Here we will only cover face detection.

Face detection uses ***classifiers***, which are algorithms that detects what is either a face(1) or not a face(0) in an image. Classifiers have been trained to detect faces using thousands to millions of images in order to get more accuracy. OpenCV uses two types of classifiers, LBP (Local Binary Pattern) and Haar Cascades. I will be using the latter classifier.

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

**Understanding Haar Cascades**

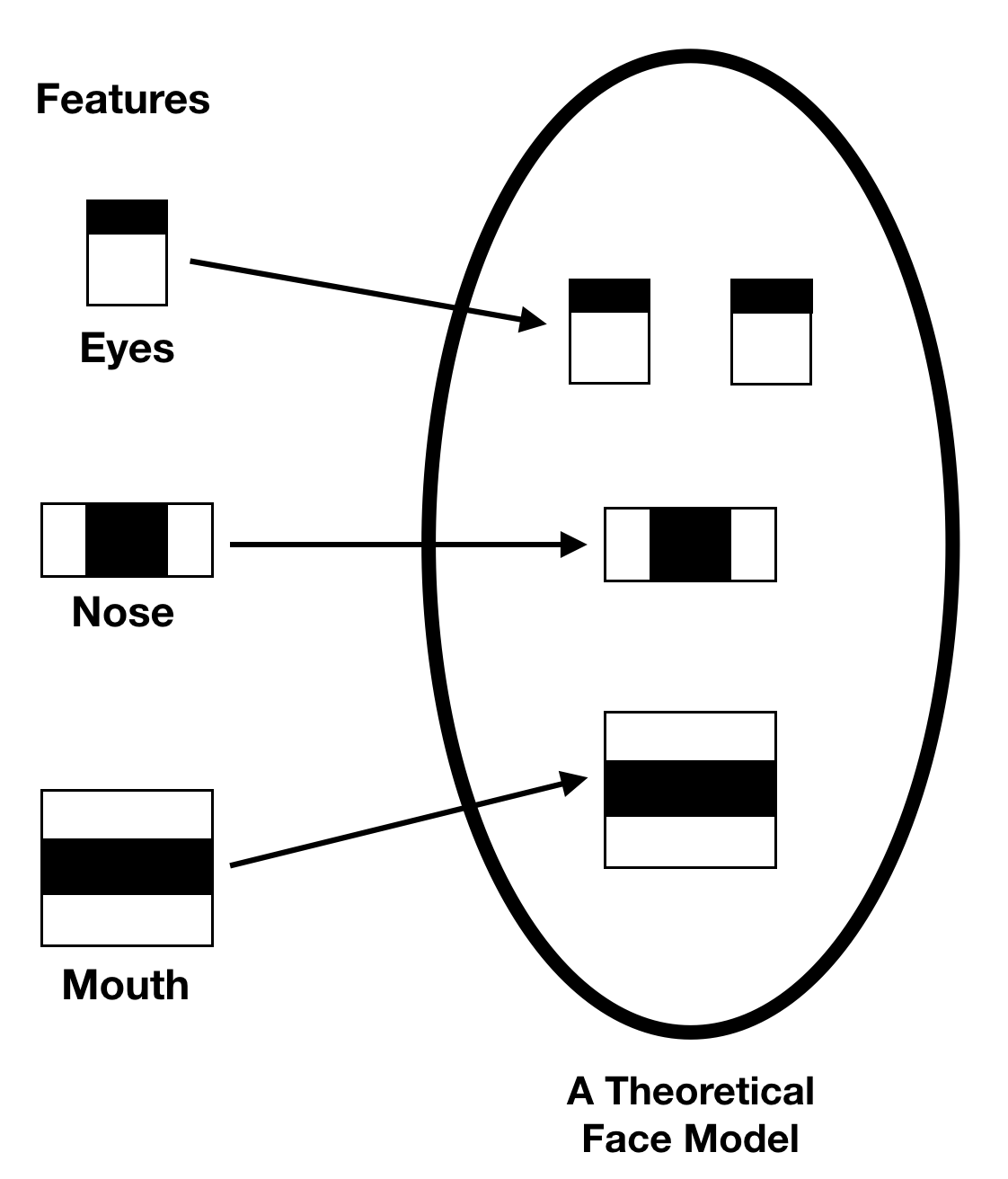
A *Haar Cascade*is based on “Haar Wavelets” which Wikipedia defines as:

**A sequence of rescaled “square-shaped” functions which together form a wavelet family or basis.**

It is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses machine learning techniques to get a high degree of accuracy from what is called “training data”. This uses “integral image” concepts to compute the “features” detected. Haar Cascades use the **Adaboost** learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers.

**Feature Extraction**

As I mentioned earlier, Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images. This process in the algorithm is feature extraction.



In feature extraction, the algorithm uses training data to best identify features that it can consider a face.

Our goal for this exercise is to be able to detect the faces in an image. OpenCV comes with many pre-trained classifiers for face, eyes, smiles, etc. Those classifiers are stored in an XML file format and can be found in

<https://github.com/opencv/opencv/tree/master/data/haarcascades>

Or click here to download

<https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_frontalface_default.xml>

#### Step 1: Import required Python modules.

First, we need to load the required Python modules into our work environment. To ensure that opencv module is loaded correctly, its version was printed below.

|  |
| --- |
| # import python modules  import cv2  print ("Version of OpenCV: {}".format(cv2.\_\_version\_\_)) |

**Step 2: Load and display image of faces to be used.**

For this exercise, we will be using a sample image of different faces. We will load this image into our work environment using the OpenCV “imread” function. Once the image has been loaded, we will use OpenCV “imshow” function to visualize the image.

|  |
| --- |
| # load image using opencv image = cv2.imread("images/faces2.jpg") gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  # display image using opencv  cv2.imshow(“Input Image”,image)  cv2.imshow(“Gray Image”,gray) |

**Step 3: Detect faces in the image using a classifier library.**

At this stage, we will use the default classifier library that came with opencv to create a face cascade that will be applied on our sample image.

|  |
| --- |
| # load the required XML classifiers  face\_cascade = cv2.CascadeClassifier ('haarcascade\_frontalface\_default.xml')  eye\_cascade =cv2.CascadeClassifier('haarcascade\_eye.xml') |

Now we find the faces in the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face and apply eye detection on this ROI (since eyes are always on the face !!! ).

|  |
| --- |
| # match the classifier with the faces image  faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.07, minNeighbors=5, minSize=(40,40))  # output number of faces found  print (“Found {0} faces!”.format(len(faces))) |

|  |
| --- |
| for (x,y,w,h) in faces:  cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)  roi\_gray = gray[y:y+h, x:x+w]  roi\_color = img[y:y+h, x:x+w]  eyes = eye\_cascade.detectMultiScale(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(‘Faces Detection’,image)  cv2.waitKey(0)  cv2.destroyAllWindows() |

**Detect faces in Real Time:**

|  |
| --- |
| import cv2  face\_cascade = cv2.CascadeClassifier ('haarcascade\_frontalface\_default.xml')  eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml')  cap = cv2.VideoCapture(0)  while 1:  ret, img = cap.read()  gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  faces = face\_cascade.detectMultiScale(gray, 1.3, 5)  for (x,y,w,h) in faces:  cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)  roi\_gray = gray[y:y+h, x:x+w]  roi\_color = img[y:y+h, x:x+w]    eyes = eye\_cascade.detectMultiScale(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  cap.release()  cv2.destroyAllWindows() |