*Face Analyzer  
Calculating Eyes Opening Ratio using OpenCV*

Table of Contents

[Scope 2](#_Toc80357385)

[Pre-requisites 2](#_Toc80357386)

[Setup 2](#_Toc80357387)

[Final Words 10](#_Toc80357388)

Scope

The target of this tutorial is to develop a lightweight command line based utility, through Python based modules to automatically detect faces in a static image and to calculate the opening ratio of the eyes of the spotted faces.  
  
The code of this tutorial was placed in the following GitHub repository: “<https://github.com/bassemmarji/FaceAnalyzer/> ”.

# Pre-requisites

The following components come into play:

* **OpenCV**: is an open-source library for computer vision, machine learning and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java and it is used for all sorts of image and video analysis like facial detection and recognition, photo editing, optical character recognition and a whole heap more.
* **Dlib**: is principally a C++ library however you can use a number of its functions from Python applications.
* **filetype**: is a small and dependency free Python package to infer file and MIME types.
* **imutils**: encompasses a series of functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges.
* **SciPy**: is an open-source library used for solving mathematical, scientific, engineering and technical problems. It allows users to manipulate and to visualize the data using a wide range of high-level Python commands. SciPy is built on the Python Numpy extention.

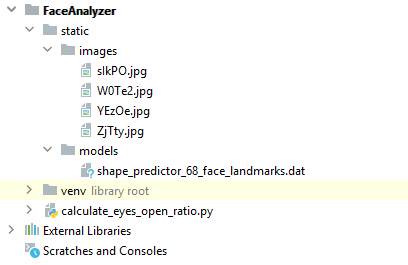
# Setup

To setup the environment, you need python3 installed on your system. It is highly recommended to setup a virtual environment which will host the needed libraries.

1. Create a virtual environment and activate it.
2. Create a file named requirements.txt and add the following lines to it.

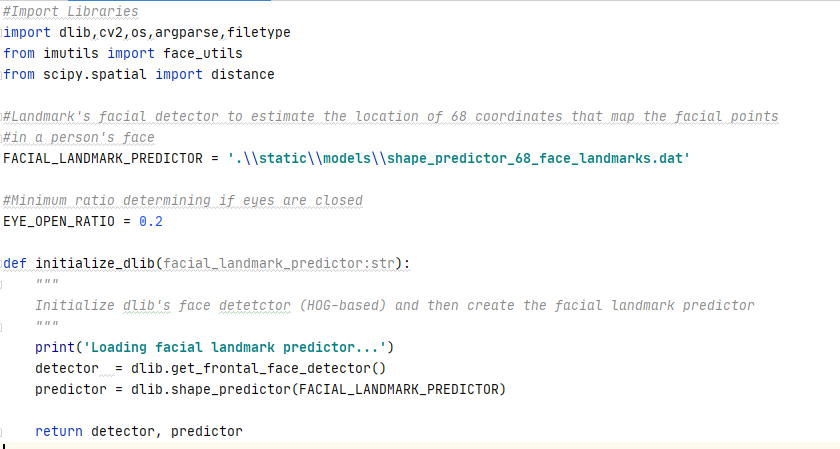
|  |
| --- |
| requirements.txt |
| opencv-python==4.4.0.46 dlib==19.17.0 imutils== 0.5.3  filetype ==1.0.7  scipy==1.5.4 |

1. Now, let’s install the required libraries to the project.  
   pip install –r requirements.txt
2. Create a folder for our project called “FaceAnalyzer”.

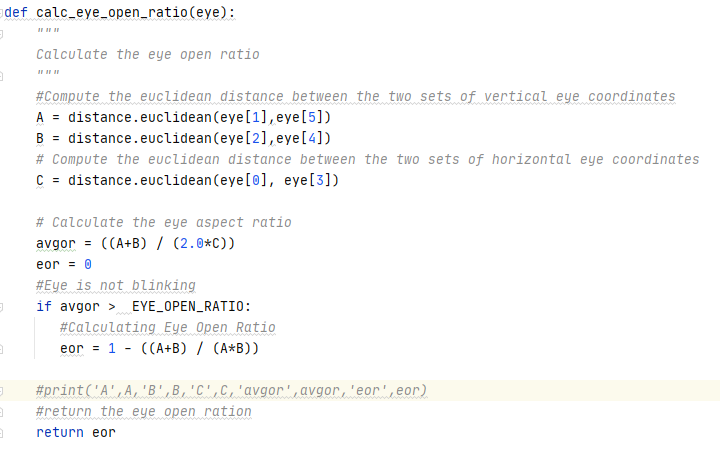
At the end, our folder structure will look like the following:  
  


**NB:**   
  
For the purpose of this article, we will use the “shape\_predictor\_68\_face\_landmarks.dat”;   
A dlib pre-trained model to estimate the location of 68 coordinates (x,y) that map the facial points on a person’s face. The model details are available in the below link   
<https://github.com/davisking/dlib-models>.

Let’s move into coding:  
 *#calculate\_eyes\_open\_ratio.py*



* This function initializes the dlib using the pre-trained model and returns:
  + detector: used for detecting the face in an image.
  + predictor: shape or landmark predictor used to predict the coordinates of a given shape. The facial landmark predictor is used to localize individual facial structures.



* Each eye is represented by 6 coordinates starting from the left-corner to the right-corner (in a clock-wise mode) as shown below:

A

B

C

P4

P3

P2

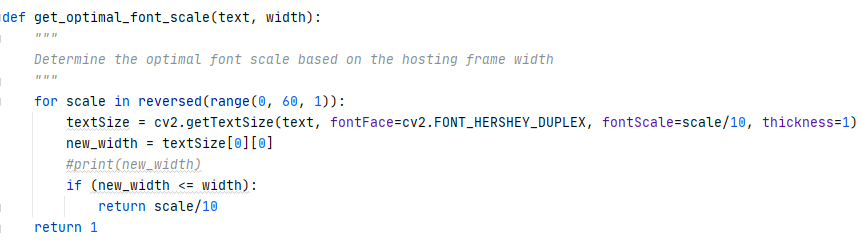
P1

P5

P6

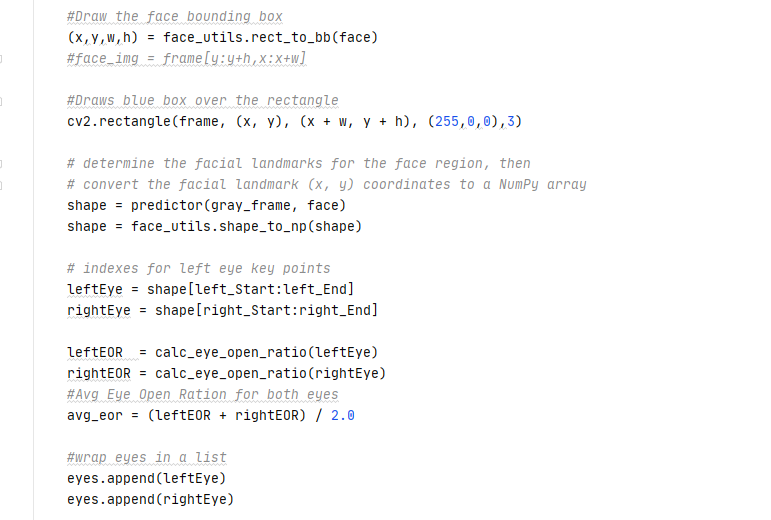
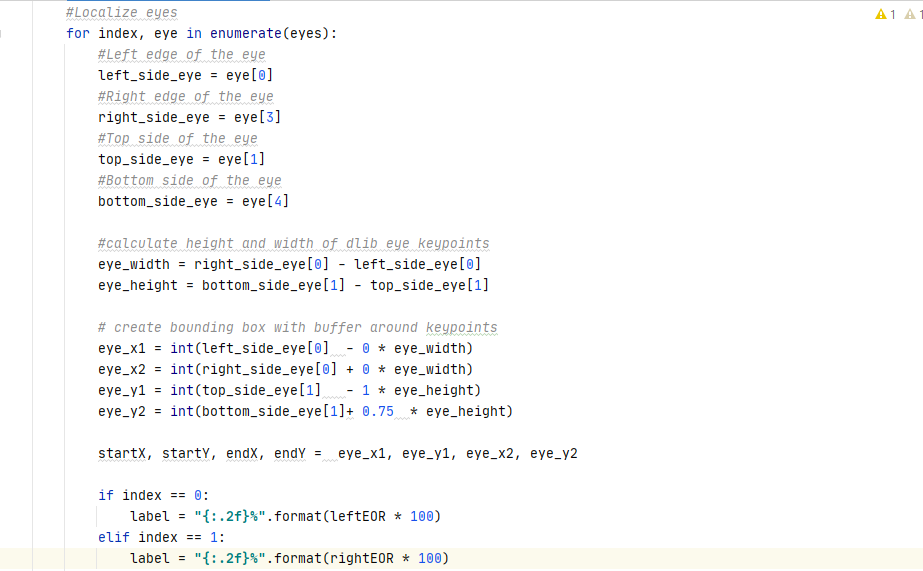
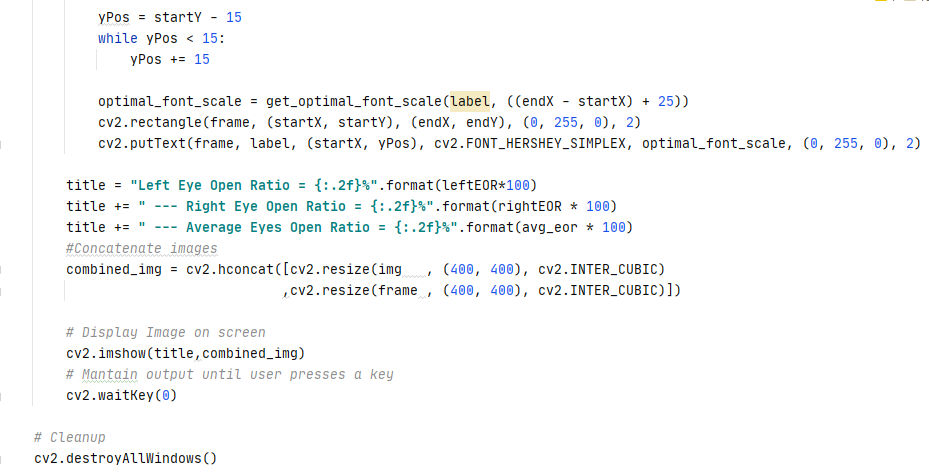
This function performs the following:

1. Calculates the distance between the vertical eye landmarks ( (P2 & P6) and (P3 & P5)).
2. Calculates the distance between the horizontal eye landmarks ( (P1 & P4) )
3. Calculates the eye aspect ratio.
4. If the eye aspect ratio is greater than the minimum eye closed ratio then it computes the eye opening ratio.

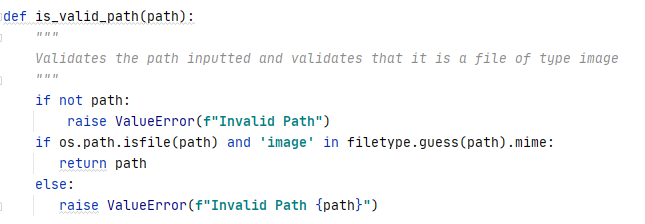


* This function calculates an optimal font scale based on the width of a hosting frame.

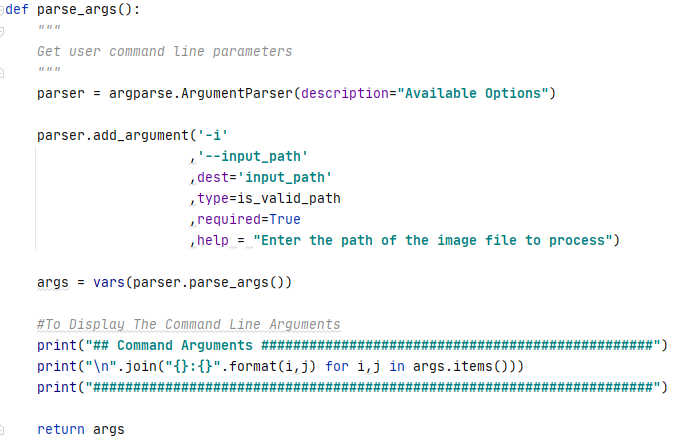


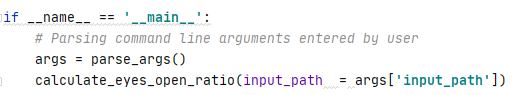
* This function constitutes the core of our program and performs the following:
  + Initialize the dlib face detector.
  + Read the input image.
  + Preserve a copy of the original image.
  + Convert the image into gray scale.
  + Detect faces in the grayed image.
  + Determine the landmarks of the left and right eyes.
  + Iterate throughout the faces detected and displays the positions of the faces detected.
  + Draw a bounding box around the faces detected.
  + Determine the facial landmarks for the face region then convert its coordinates to a Numpy array.
  + Calculate the opening ratio of the left and right eyes and calculate the average open ratio.
  + Localize the eyes and draw a bounding box around the eyes considered as the regions of interest (ROI) then add a label for each frame exhibiting the opening ratio.
  + Display a window containing the original image, and the processed image.



* This function validates a path inputted as a parameter and ensures that it is a file path also it ascertains that the type of the file chosen is an image.



* This function defines and sets the appropriate constraints for the command line arguments to be specified by the user when running this utility.
  + input\_path: A required parameter containing the path of the image file to process associated with the predefined function “is\_valid\_path”.

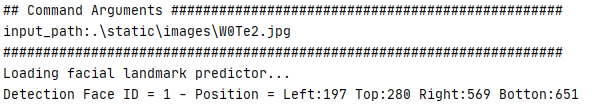


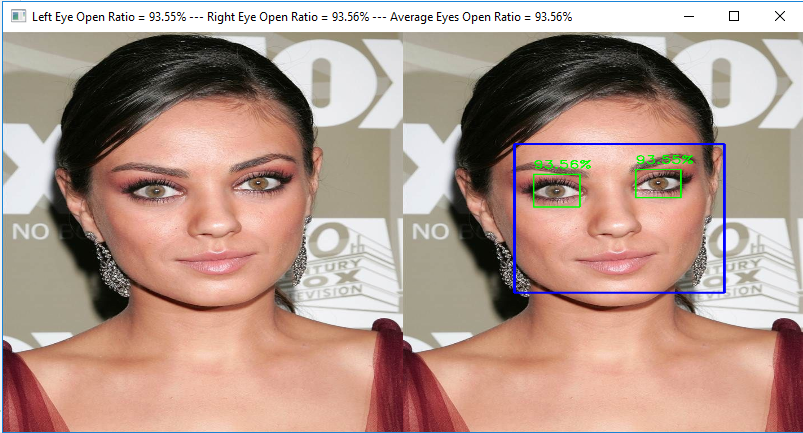
* The above represents the main function of our program.

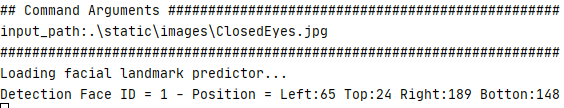
Let’s test our program:  
  
Kindly proceed as per the following steps:

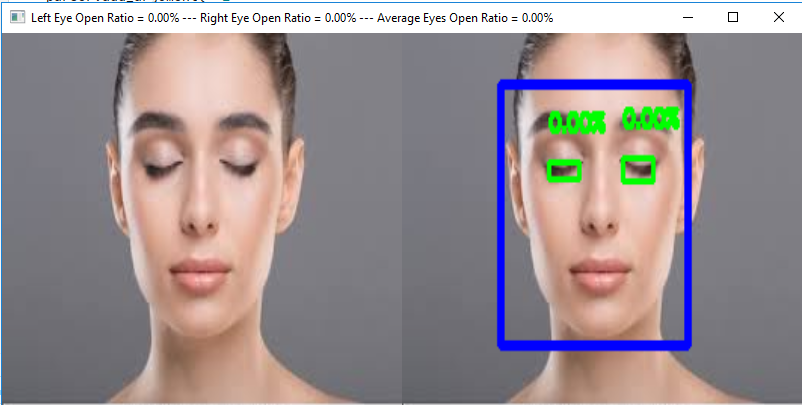
***NB: Please select images with high resolution while testing this utility.***

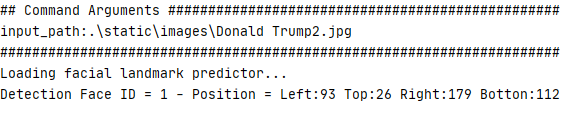
Open up a terminal window and type the following in it:

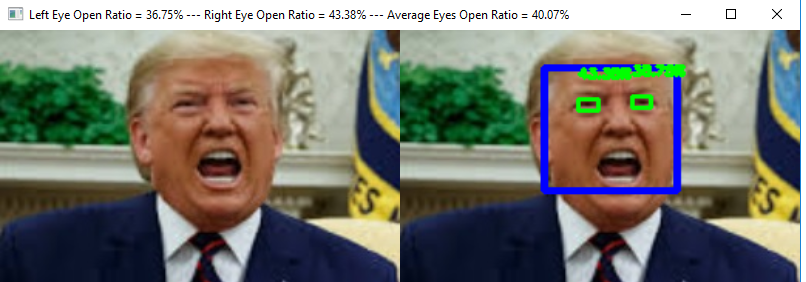
**calculate\_eyes\_open\_ratio –i ".\static\images\W0Te2.jpg"**  
The following summary will be displayed in the terminal:  
  


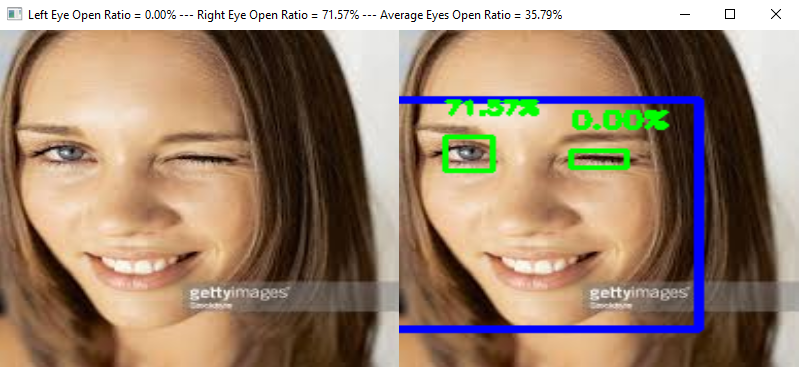


**calculate\_eyes\_open\_ratio –i** ".\static\images\ClosedEyes.jpg"  
  




calculate\_eyes\_open\_ratio **-i** ".\static\images\Donald Trump2.jpg"  
  




calculate\_eyes\_open\_ratio -i ".\static\images\OneEye.jpg”  
  


Final Words  
  
Hope you enjoyed this article.

|  |  |
| --- | --- |
|  | ***Bassem Marji*** *is a project implementation manager at BLOM Bank with a proven track record of success.  He managed the implementation of over 50 projects and propelled the digital transformation of mission critical applications. He spends his free time discovering the latest technology trends in the IT field.* |