

FACIAL RECOGNITION

Progress report

**This report is submitted in partial fulfilment of the requirements of the
Degree in Electronic and Communications Engineering (DT008) of the
Technological University Dublin**

Feb 09th, 2020

Supervisor: Dr. Kevin T

School of Electrical and Electronic Engineering

Talha Tallat – D18124645

D18124645@mytudublin.ie

Table of Contents

Table of Figures	1
DECLARATION	2
Project Report Check List	3
1. Introduction	4
2. Initial Research	5
2.1. Introduction to Anaconda	5
2.2. Anaconda Setup	5
2.3. Introduction to OpenCV	7
2.4. OpenCV setup	8
2.4.1. Installing an OpenCV manually for Python on Windows using Anaconda	8
2.4.2. Installing an OpenCV with Anaconda Navigation	9
3. Code Developed	10
4. Remaining Work	12
4.1. Project Planner	12
5. Conclusion	13
6. Reference	13

Table of Figures

Figure 1 - Patterns, facial textures and shapes on the face - By Simon Davies	4
Figure 2 - List of Anaconda's packages	5
Figure 3 - Anaconda's available versions	5
Figure 4 - Python Version is 3.7.4	6
Figure 5 - Anaconda version is 1.7.2	6
Figure 6 - Importing 'sys' library to check, if it is installed	6
Figure 7 - Cascade approach to detect & identify faces	7
Figure 8 - Creating a new environment in Anaconda Navigation	8
Figure 9 - Anaconda prompt command to install OpenCV	8
Figure 10 - OpenCV libraries available to install in Anaconda	9
Figure 11 - Importing 'OpenCV' library to check, if it is installed	9
Figure 12 - The code for detecting faces	10
Figure 13 - The Cascades stored in the folder, where the python code is located	10
Figure 14 - Photo window shows the selected images	11
Figure 15 - Box around the face shoes the Detected face	12

DECLARATION

I, the undersigned, declare that this report is entirely my own written work, except where otherwise accredited, and that it has not been submitted for a degree or other award to any other university or institution.

Signed: *Talha Tallat*

Date: **09/02/2020**

Project Report Check List

1. Include a title sheet as provided on Brightspace	✓
2. Include declaration page as provided on Brightspace	✓
3. Include an index with page numbers (Table of Contents)	✓
4. Include a list of acronyms	✓
5. When using acronyms spell them out in full first	✓
6. Include page numbering	✓
7. Keep the font and formatting style the same throughout	✓
8. Use consistent line spacing	✓
9. Do not abbreviate	✓
10. Write in the third person	✓
11. Generally, write in the past tense	✓
12. Run a spell check through the document	✓
13. Check for any grammatical errors	✓
14. Have the document proofread by someone else	✓
15. Number each section	✓
16. Refer to all figures and tables in the text before inclusion	✓
17. Label all figures and tables correctly and consistently	✓
18. Figure captions go below the figure	✓
19. Table captions go above the table	✓
20. Use figures to explain the theory	✓
21. Include an introduction with project objectives	✓
22. Do not quote the project brief verbatim as the introduction	✓
23. Identify the methods used	✓
24. Quantify and analyse results	✓
25. Include a conclusion summing up main findings	✓
26. Reference all cited source materials using the IEEE style guide	✓
27. Organise the appendices into sections with page numbers, titles, etc.	✓

Student signature: Talha Tallat

1. Introduction

A **Facial recognition** system is a technology capable of identifying or verifying a person from a digital image, video or a live video source from a camera. This technology detects people's faces and recognizes, who the person is. They work by reading the facial features of a given image and compare the faces with a face within a database. It is also known as Biometric Artificial Intelligence that can almost perfectly identify by analysing patterns based on the person's facial textures and shapes.

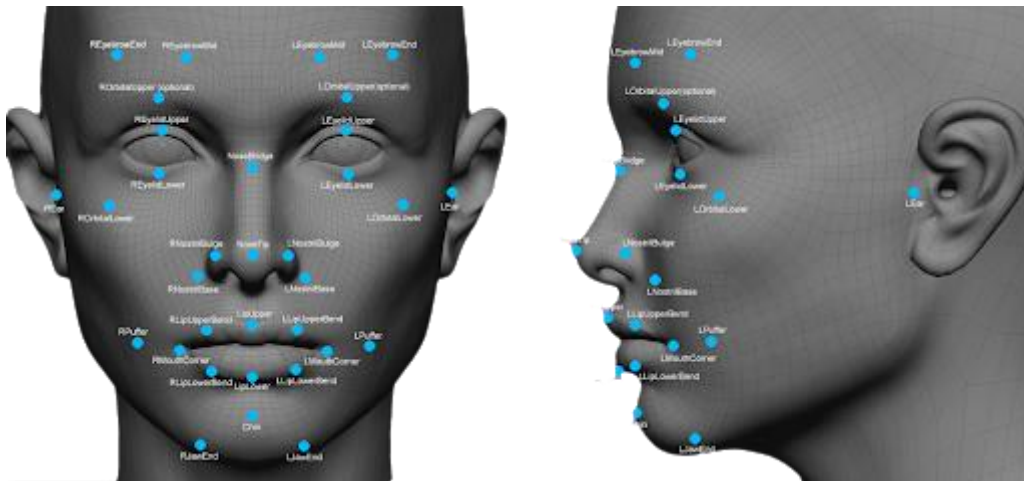


Figure 1 - Patterns, facial textures and shapes on the face - By Simon Davies

- The **face detection** process is an important step as it detects and discovers human faces in images and videos.
- The **face capture** process transforms analogue information into digital information data based on the person's facial features.
- The **face match** process compares two faces and verifies that both faces belong to each other.

These facial recognition technologies are used every day for things like phone security and for serious uses from important organizations, such as like, banks and governments.

Some banks are looking at implementing facial recognition technologies into their ATMs as a security measure for verifying transactions.

Government agencies, such as ICE and the FBI are using facial recognition to create the database from existing documents like drivers' licenses.

2. Initial Research

A simple way to start facial Recognition was to use Python programming language and the open-source library OpenCV to be able to process image. In order to run Python on Windows Operating system, a well-known “**Anaconda**” free and open-source distribution of them Python had to be installed.

2.1. Introduction to Anaconda

Anaconda is a free and open-source distribution of the Python and R programming language for scientific computing, such as data science, large scale data processing, machine learning applications etc. The goal is to provide simplified package management and deployment.

Anaconda is very popular because it provides short and simple setup and most importantly, it has many of the tools used in data science and machine learning with just only one installation. Anaconda Distribution contains Anaconda Navigator, Python and hundreds of scientific packages.

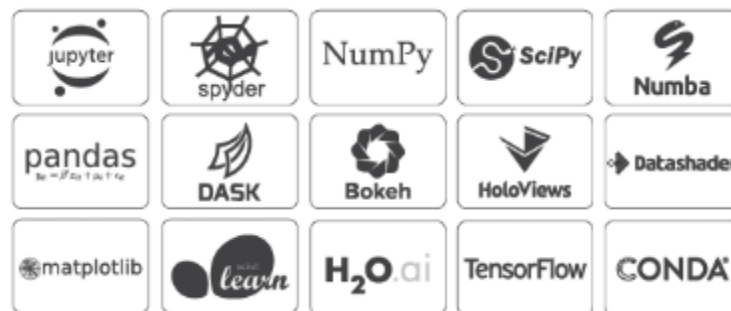


Figure 2 - List of Anaconda's packages

2.2. Anaconda Setup

Anaconda was installed on the windows operating system and the download link for the installation was available on Anaconda’s original website. The link for the installation as follows: <https://www.anaconda.com/distribution/>.

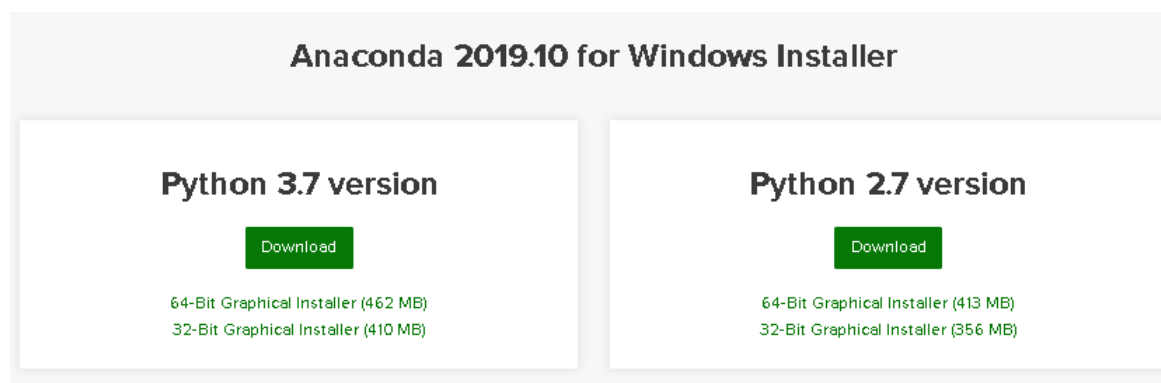
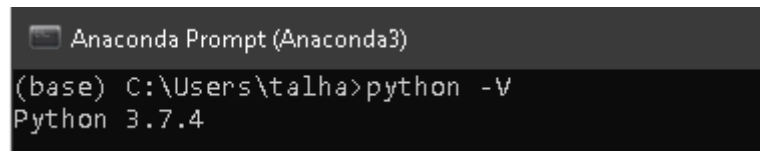


Figure 3 - Anaconda’s available versions

There are “Python 3.7” and Python 2.7” versions available on Anaconda ‘s official website for windows installer as shown in figure 3. The latest version Python 3.7 (64-Bit) was installed on the computer with a size of 462 MB.

To make sure the right version is installed, the Anaconda Prompt was opened to check the right versions are installed. The commands for checking the Python version and Anaconda version are “python -V” & “Anaconda -V”.

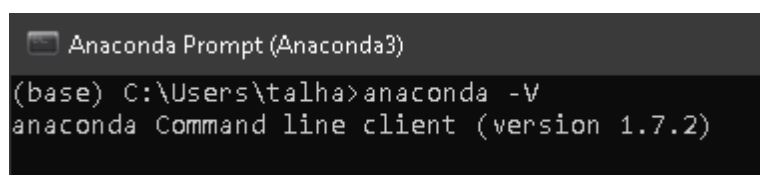


```

Anaconda Prompt (Anaconda3)
(base) C:\Users\talha>python -V
Python 3.7.4

```

Figure 4 - Python Version is 3.7.4



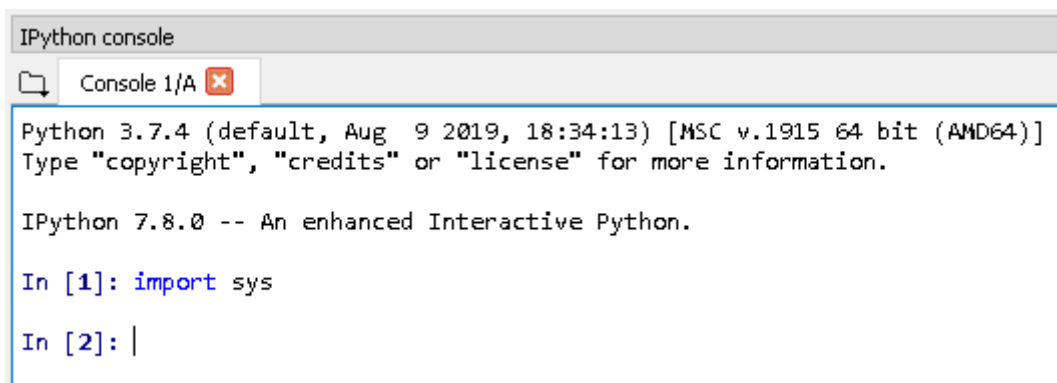
```

Anaconda Prompt (Anaconda3)
(base) C:\Users\talha>anaconda -V
anaconda Command line client (version 1.7.2)

```

Figure 5 - Anaconda version is 1.7.2

Most of the libraries are already installed in Anaconda because Anaconda provides most of the useful libraries. The “sys” library was imported in the Python console to test the library, whether it is installed or not. If it is not, an error will show up, meaning that the library is not installed otherwise there is no error.



```

IPython console
Console 1/A
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: import sys
In [2]: |

```

Figure 6 - Importing 'sys' library to check, if it is installed

The most important library that needs to be installed is “OpenCV2” and Anaconda provides that library, which was installed throw Anaconda.

2.3. Introduction to OpenCV

Open CV is an open-source computer vision library developed by the Intel. It mostly used for all the various face recognitions, object recognitions, and all other machine learning and a deep learning technique. Open cv is cross-platform itself but originally written in C/C++ and now it provides bindings for various programming languages, such as Python, Java, MATLAB, Interfaces, Windows and Linux.

OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are difficult task since it requires thousands of small patterns and features to be matched. These algorithms break the task of identifying the faces into small bite-sized tasks, making it easy to solve. These tasks are also known as classifiers.

The algorithms start at the top left of a picture and move down across small blocks of data, looking at each block, constantly asking, "Is this a face? ... Is this a face?" Since there are approximately 6,000 or more tests per block, that might have millions of calculations which will grind the computer to a halt.

To avoid this, OpenCV uses **cascades**, which breaks the problem of detecting faces into multiple stages. For each block, it performs a rough test, if the test passes then it performs a more detailed test and continues to do for the rest of them. These algorithms may have 30 to 50 of these stages or cascades, and it will only detect a face if all stages pass. The blocks are shown in figure 7.

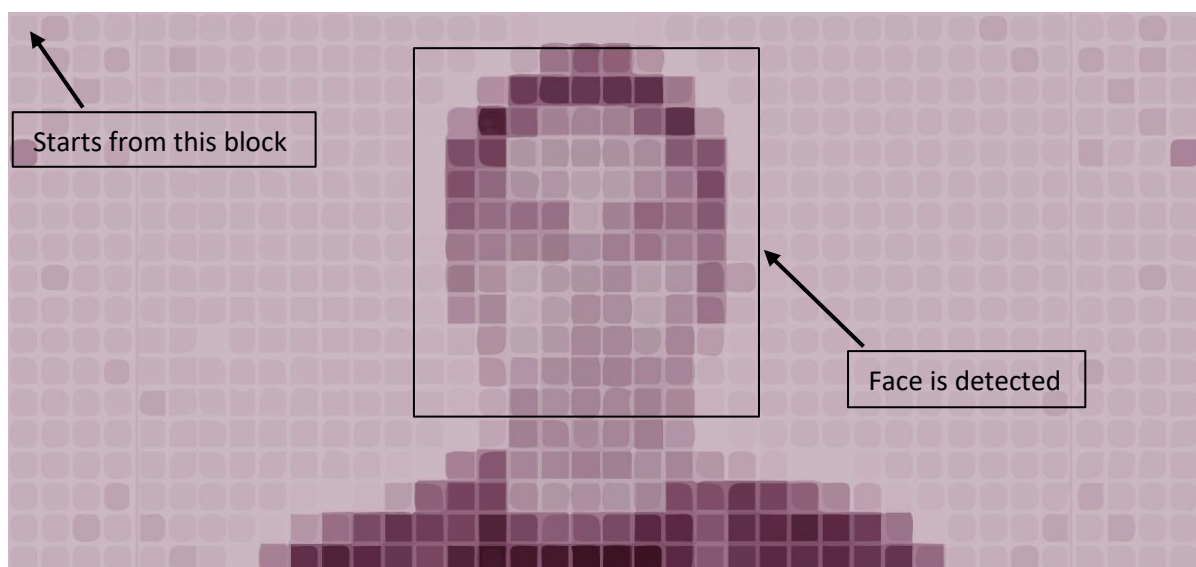


Figure 7 - Cascade approach to detect & identify faces

Cascades are just a bunch of XML files that contain OpenCV data used to detect objects. Cascade does all the work when initializing code with the selected cascade.

2.4. OpenCV setup

Open CV is the simplest way to recognize a face using Python language on Anaconda. To install open CV on the Anaconda, the Anaconda is installed first followed by OpenCV.

2.4.1. Installing an OpenCV manually for Python on Windows using Anaconda

OpenCV is a library for image processing and Computer Vision. Downloading the Anaconda graphical installer for windows platform from their original website to run the OpenCV library “<https://www.anaconda.com/distribution/>”, since I am using windows architecture on my computer.

A new environment had to be created in Anaconda Navigator to install OpenCV packages with an older version of Python 3.6. Since the latest 3.7 version was not supporting. Then OpenCV had to be activated in the Anaconda Prompt, which was creating a lot of issues.

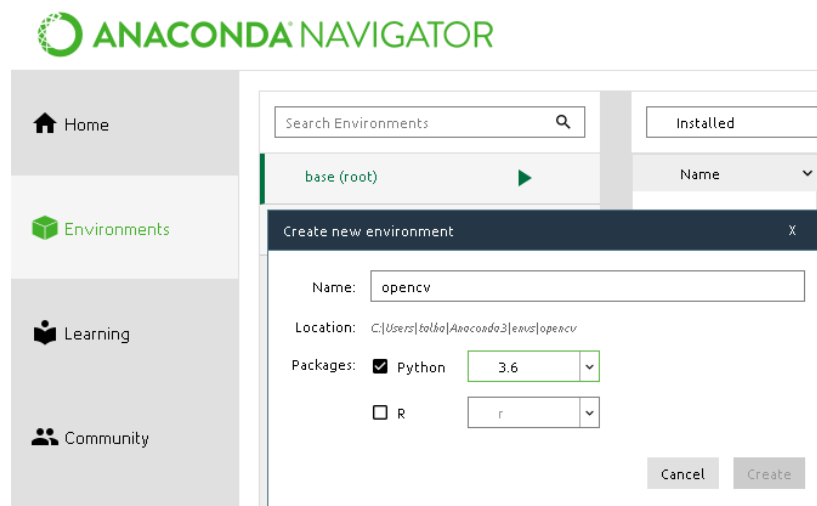


Figure 8 - Creating a new environment in Anaconda Navigation

The following command is typed in the Anaconda prompt to install the OpenCV.

```
Administrator: Anaconda Prompt (Anaconda3) - conda install -c conda-forge opencv
(base) C:\Windows\system32>conda install -c conda-forge opencv
```

Figure 9 - Anaconda prompt command to install OpenCV

This was taking too long to install those packages due to the internet speed and it was not resolving, because installing OpenCV manually was a difficult task. A simple way was to do through Anaconda, it provides with OpenCV option to install it.

2.4.2. Installing an OpenCV with Anaconda Navigation

A simple way to install OpenCV was with the Anaconda Navigation. OpenCV was typed in the search bar and all library option was selected and there were three main libraries available for OpenCV to install “libopencv”, “opencv” and “py-opencv” as shown in figure 10. It does not take much time as compared to manually installed.

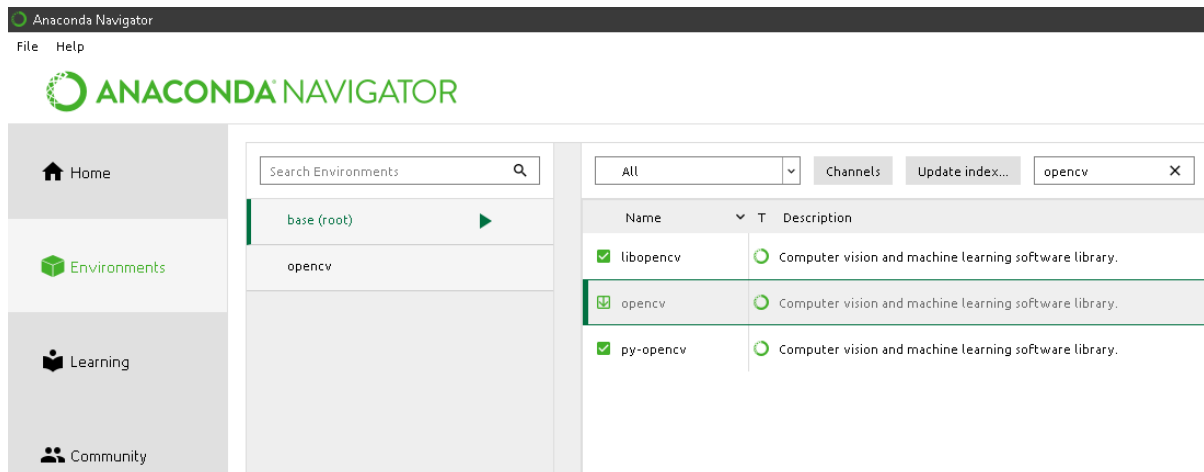


Figure 10 - OpenCV libraries available to install in Anaconda

The OpenCV was imported in the Python console in 'Spyder' to check whether the “OpenCV” library was installed properly or not. Figure 11 shows the OpenCV was installed properly without an error.

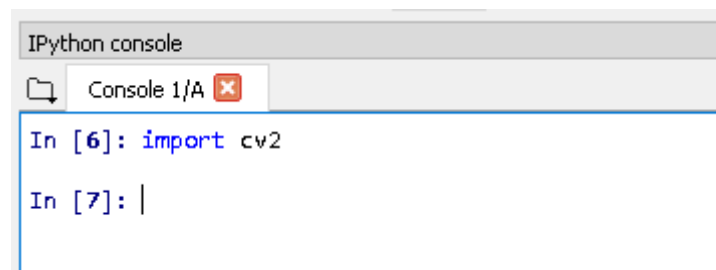
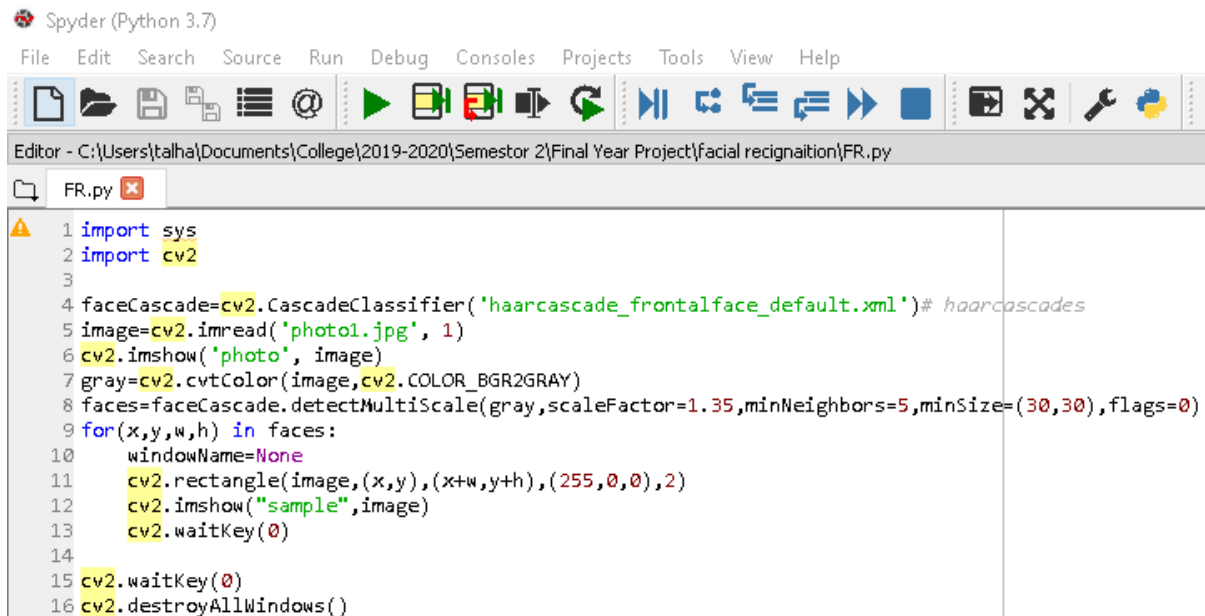


Figure 11 - Importing 'OpenCV' library to check, if it is installed

3. Code Developed

The progress on the code is devolved which detects faces only. The code is running on the Spyder with the latest version Python 3.7.



```

1 import sys
2 import cv2
3
4 faceCascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')# haarcascades
5 image=cv2.imread('photo1.jpg', 1)
6 cv2.imshow('photo', image)
7 gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
8 faces=faceCascade.detectMultiScale(gray,scaleFactor=1.35,minNeighbors=5,minSize=(30,30),flags=0)
9 for(x,y,w,h) in faces:
10     windowName=None
11     cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
12     cv2.imshow("sample",image)
13     cv2.waitKey(0)
14
15 cv2.waitKey(0)
16 cv2.destroyAllWindows()
  
```

Figure 12 - The code for detecting faces

The 1st line of the code in fig. 12 imports the 'sys' library, which provides information about the functions and methods of the python and 2nd line of code imports the OpenCV library.

The 4th line of code defines the face cascade equal to the loading of that cascade “cv2.CascadeClassifier ('haarcascade_frontalface_default.xml')” and the frontal face haar cascade is downloaded onto the folder where the python files are located. The cascade files were available on the following GitHub link.

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

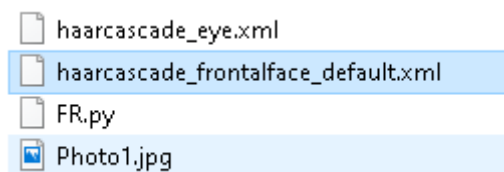


Figure 13 - The Cascades stored in the folder, where the python code is located

The 5th line of code has a variable called “image”, which hold the location of actual Photo1.jpg in the same directory as shown in figure 13 and the 1 indicates to the colour photograph.

The 6th line shows the photo that was read.

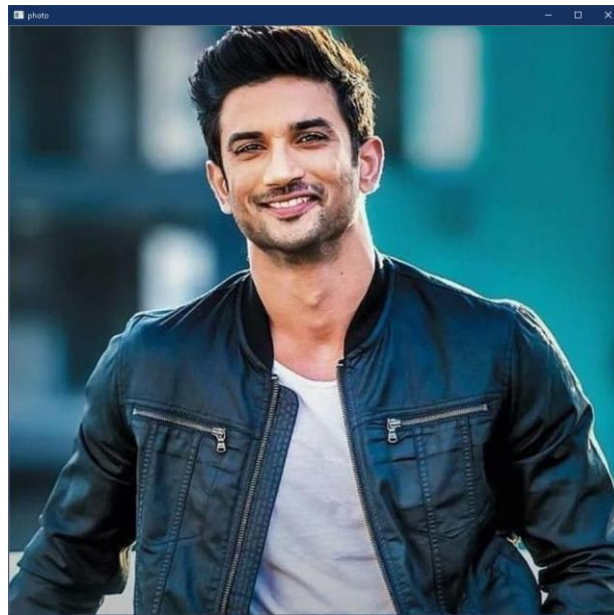


Figure 14 - Photo window shows the selected images

The 7th line converts image to grey image because classifier takes in a grey image, since it does not matter whether the image is coloured or black and white to detect a face, so distraction feature was removed.

The 8th line has a variable “faces” and this variable “faceCascade” is for loading the classifier from line 4. This function “detectMultiScale” returns a rectangle where the face is detected in the image. Then the grey image is passed with other parameters such as “scaleFactor” and “minNeighbors”.

scaleFactor = 1.32, because that turn out to be the functional value since 1.3 is like saying, scale the image and decrease its size by 32%. This classifier “haarcascade_frontalface_default.xml” was trained for some specific size of the image, so the images there are bigger in size are likely to be not detected by this classifier. That’s why it’s important to scale the size and rescale it so that it has more chances of getting detected.

minNeighbors = 5, because that is the optimal value. Since there are lots of false faces were detected by the image so it should have at least 5 neighbours for it to be detected as a true positive face.

The 9th and 11th line allowed drawing a rectangle around the detecting faces as shown in figure 15.

The 12th line shows the detected image as shown in figure 15.

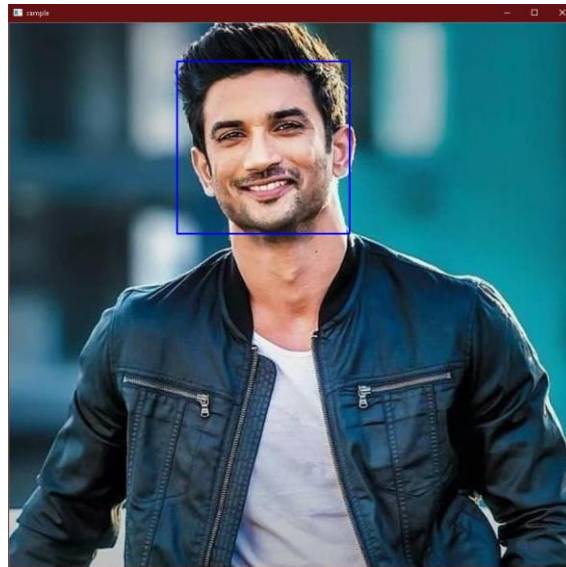


Figure 15 - Box around the face shows the Detected face

4. Remaining Work

There are a good few points need to cover to complete this project. The aim is to get these points to finish, which will complete facial recognition.

- Identify the faces and should be able to tell who it is
- Testing the accuracy and reliability
- Live video camera face detection and identification
- Create a directory that contains training images to recognize the face accurately

4.1. Project Planner

Starting Date: 27/01/2020

Ending Date: 30/03/2020

ACTIVITY	Weeks	PLAN DURATION Time	PERCENT COMPLETE
Install Anaconda	1	27/01 - 3/02	100%
Install OpenCV	2	3/02 - 10/02	100%
Install cascade	3	10/02 - 17/02	100%
Detect face	4	17/02 - 24/02	95%
Recognize face	5	24/02 - 2/03	5%

Create directory	6	02/03 - 09/03	2%
Training images	7	09/03 - 16/03	0%
Testing	8	23/03 - 30/03	15%

5. Conclusion

Face dedication works very well since it can recognize multiple faces simultaneously. It's not perfect yet, however, it can be perfect after tweaking scale Factor and min Neighbour. There are more testing needs to be done to get more precision for detection.

6. Reference

Davies, S. (2020). *privacysurgeon*. [online] Privacysurgeon.org. Available at: <http://www.privacysurgeon.org/blog/wp-content/uploads/2015/07/face-recognition-image-e1436186948904.png> [Accessed 9 Feb. 2020].

Python, R. (2020). *Face Recognition with Python, in Under 25 Lines of Code – Real Python*. [online] Realpython.com. Available at: <https://realpython.com/face-recognition-with-python/> [Accessed 9 Feb. 2020].

[1]"Anaconda Python/R Distribution - Free Download", *Anaconda*, 2020. [Online]. Available: <https://www.anaconda.com/distribution/>. [Accessed: 9- Feb- 2020].

Lynch, J. (2020). *Face Off: Law Enforcement Use of Face Recognition Technology*. [online] Electronic Frontier Foundation. Available at: <https://www.eff.org/wp/law-enforcement-use-face-recognition> [Accessed 9 Feb. 2020].

[2]"opencv/opencv", *GitHub*, 2020. [Online]. Available: <https://github.com/opencv/opencv/tree/master/data/haarcascades>. [Accessed: 9- Feb- 2020].