Real-Time Face Detection using OpenCV and Python

A Project report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR'S OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

by

Rao Suraj Rao
MIS-112015108
Semester-IV

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Indian Institute of Information Technology Pune

Near Bopdev Ghat, Kondhwa Annexe, Yewalewadi, Pune, Maharashtra 411048

APRIL 2022

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"Real-time Face Detection using OpenCV and Python"** *submitted by* **Rao Suraj Rao** *bearing the* **MIS No: 112015108***, in completion of his project work under the guidance of* **Dr.Chandrakant Guled** *is accepted for the project report submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering, Indian Institute of Information Technology, Pune, during the academic year 2022-2023.*

Dr.Chandrakant Guled
Project Guide
Assistant Professor
Department of CSE
IIIT Pune

Dr.Ritu Tiwari
Head of department
Professor
Department of CSE
IIIT Pune

Project Viva-voice held on 29-04-2022

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

This project would not have been possible without the help and cooperation of many. I would like to thank the people who helped me directly and indirectly in the completion of this project work.

First and foremost, I would like to express my gratitude to our beloved director, ***Dr.Anupam Shukla****, for providing his kind support in various aspects.*

I would like to express my gratitude to my project guide ***Dr.Chandrakant Guled****, Professor, Department of CSE, for providing excellent guidance, encouragement, inspiration, and constant and timely support throughout this B.Tech project.*

I would like to express my gratitude to the head of the department ***Dr.Ritu Tiwari****, Professor, Department of CSE, for providing his kind support in various aspects.*

I would also like to thank all the faculty members in the Dept. of CSE and my classmates for their steadfast and strong support and engagement with this project.

Abstract

This project gives an ideal way of detecting and recognizing human faces using OpenCV, and python which is part of Machine learning. It contains the ways in which Machine learning, an important part of the computer science field, can be used to determine the face using several libraries in OpenCV along with python. This System will contain a proposed system that will help in detecting the human face in real-time. This implementation can be used on various platforms in machines and smartphones, and several software applications.

Keywords - Python, OpenCV, Machine Learning, Face detection

Table of Contents

Chapter 1. Introduction

Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing

Chapter 2. Motivation

This Face Detection Project is developed for our Security CCTV Police, our Indian army and even Common Use of people to detect Faces in single and groups of people in Images. Today in the world, Security is advancing to detect terrorists and criminals easily in crowds. This Project will easily detect faces even old/blur Images with very less time for execution. This Face Detection Code can be deployed and used in any application camera, security app, or website to find faces easily.

Chapter 3. Project Requirements

- Requirements for this project are: Python 2.7 or 3.7, OpenCV, Numpy, Haarcascade classifier
- OpenCV-Python supports all the leading platforms like Mac, OS, Linux, and Windows.
- It can be installed in either of the following ways:

  Packages for standard desktop environments (Windows, macOS, almost any GNU/Linux distribution)

  ➢ run pip install opencv-python if you need only main modules
  ➢ run pip install opencv-contrib-python if you need both main and contrib modules

- We need to download trained classifier XML file (haarcascade_frontalface_default.xml)

Chapter 4. Technology Overview

4.1 Python:

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. It is used for:

- web development (server-side)
- software development
- Mathematics
- system scripting
- Python can be used on a server to create web applications.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.
- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this project Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

4.2 OpenCV:

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSDlicensed library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposed to the Cbased OpenCV 1.x API (C API is deprecated and not tested with "C" compiler since OpenCV 2.4 releases).
OpenCV has a modular structure, which means that the package includes several shared or static libraries.
The following modules are available:

- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.
- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.

Chapter 5. Literature Review

Limitations of the Existing System

The existing or traditional face recognition system has some limitations which can be overcome by adopting new methods of face recognition:

- The existing system cannot tolerate variations in the new face image. It requires the new image to be almost exactly matching with one of the images in the database which will otherwise result in denial of access for the individual.
- The performance level of the existing system is not appreciable.

The proposed system and its advantages:-

The proposed system here is face detection through haar cascade classifier using opencv.

- It can Detect faces easily with less time of execution.
- It can Detect Faces in any quality of Image even blur.
- It can Detect single, double, or even multiple faces in any Image.
- It can tell the number of People/faces in the Image.
- It is a fully automatic face recognition system
- It is one of the best in today's world of Innovation as per short amount of code used in the system.
- It uses Haar feature-based Algorithm to detect which makes it very accurate and fast processing of face detection

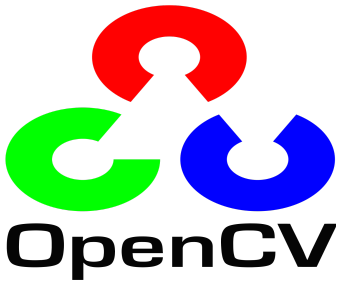
The directory structure of the project is as follows:-

1. **OpenCV-Python**
    - Overview
    - Installation
2. **Images and OpenCV**
    - Importing images in OpenCV
3. **Face Detection**
    - Overview
    - Face detection classifiers
    - Haar cascade classifier
4. **Steps to implement human face detection with Python & OpenCV**

Chapter 6. Methodology

5.1 OpenCV-Python

5.1.1 Overview:-



OpenCV was started at Intel in the year 1999 by **Gary Bradsky**. The first release came a little later in the year 2000. OpenCV essentially stands for **Open Source Computer Vision Library**. Although it is written in optimized C/C++, it has interfaces for Python and Java along with C++. OpenCV boasts of an active user base all over the world with its use increasing day by day due to the surge in computer vision applications.

OpenCV-Python is the Python API for OpenCV. You can think of it as a python wrapper around the C++ implementation of OpenCV. OpenCV-Python is not only fast (since the background consists of code written in C/C++) but is also easy to code and deploy(due to the Python wrapper in the foreground). This makes it a great choice to perform computationally intensive programs

5.1.2 Installation:-

OpenCV-Python supports all the leading platforms like Mac OS, Linux, and Windows. It can be installed in either of the following ways:

1. From pre-built binaries and source :

2. Unofficial pre-built OpenCV packages for Python.

Packages for standard desktop environments (Windows, macOS, almost any GNU/Linux distribution)

➢ run pip install opencv-python if you need only main modules
➢ run pip install opencv-contrib-python if you need both main and contrib modules (check extra modules listing from OpenCV documentation)

You can either use Jupyter notebooks or any Python IDE of your choice for writing the scripts.

5.2 OpenCV and Images

Before we jump into the process of face detection, let us learn some basics about working with OpenCV. In this section, we will learn how to import images into OpenCV.

5.2.1 Importing Images in OpenCV :-

➢ **Using Jupyter notebooks :-**

- Import the necessary libraries

  ```
  import numpy as np

  import cv2

  import matplotlib.pyplot as plt

  %matplotlib inline
  ```

- Read in the image using the **imread function**. We will be using the colored 'mandrill' image for demonstration purpose.

  ```
  img_raw = cv2.imread('image.jpg')
  ```

- **The type and shape of the array**.

  type(img_raw)

  numpy.ndarray

  img_raw.shape

  (1300, 1950, 3)

  Thus, the .png image gets transformed into a numpy array with a shape of 1300x1950 and has 3 channels.

- **Viewing the image**

  plt.imshow(img_raw)

➢ **Using Python Scripts**:-

Jupyter Notebooks are great for learning, but when dealing with complex images and videos, we need to display them in their own separate windows. In this section, we will be executing the code as a .py file. You can use Pycharm, Sublime or any IDE of your choice to run the script below

```
import cv2

img = cv2.imread('image.jpg')

while True:

    cv2.imshow('mandrill',img)

    if cv2.waitKey(1) & 0xFF == 27:

        break

cv2.destroyAllWindows()
```

In this code, we have a condition, and the image will only be shown if the condition is true. Also, to break the loop, we will have two conditions to fulfill:

- The cv2.waitKey() is a keyboard binding function. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues
- The second condition pertains to the pressing of the Escape key on the keyboard. Thus, if 1 millisecond has passed and the escape key is pressed, the loop will break and program stops
- cv2.destroyAllWindows() simply destroys all the windows we created. If you want to destroy any specific window, use the function cv2.destroyWindow() where you pass the exact window name as the argument

5.3 Face detection

5.3.1 Overview:-

Face detection is a technique that identifies or locates human faces in digital images. A typical example of face detection occurs when we take photographs through our smartphones, and it instantly detects faces in the picture. Face detection is different from Face recognition. Face detection detects merely the presence of faces in an image while facial recognition involves identifying whose face it is. In this article, we shall only be dealing with the former.

5.3.2 Face Detection Classifiers:-

Face detection is performed by using classifiers. A classifier is essentially an algorithm that decides whether a given image is positive (face) or negative(not a face). A classifier needs to be trained on thousands of images with and without faces. Fortunately, OpenCV already has two pre-trained face detection classifiers, which can readily be used in a program. The two classifiers are:

➢ Haar Classifier
➢ Local Binary Pattern (LBP) classifier.

In this project, however, we will only discuss the Haar Classifier

5.3.3 Haar feature-based cascade classifiers:-

- 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. Paul Viola and Michael Jones in their paper titled "Rapid Object Detection using a Boosted Cascade of Simple Features" used the idea of Haar-feature classifier based on the Haar wavelets.
- This classifier is widely used for tasks like face detection in the computer vision industry.
- Haar feature-based cascade classifiers is an effectual machine learning-based approach, in which a cascade function is trained using a sample that contains a lot of positive and negative images. The outcome of AdaBoost classifier is that the strong classifiers are divided into stages to form cascade classifiers

Haar cascade files:-

OpenCV comes with a lot of pre-trained classifiers. For instance, there are classifiers for smiles, eyes, faces, etc. These come in the form of xml files and are located in the opencv/data/haarcascades/ folder. However, to make things simple, you can also access them from here. Download the xml files and place them in the data folder in the same working directory as the jupyter notebook.

You can find the necessary XML files at:-

/home/<username>/.local/lib/<python-version>/site-packages/cv2/data/

Loading the classifier for frontal face:-

haar_cascade_face=cv2.CascadeClassifier('data/haarcascade/haarcascade_frontalface_default.xml')

5.4 Steps to implement human face detection with Python & OpenCV:-

1. **Imports**

```
import cv2
```

2. **Initialise the classifier**

```
# Create the haar cascade
faceCascade = cv2.CascadeClassifier("Resources/haarcascade_frontalface_default.xml")
```

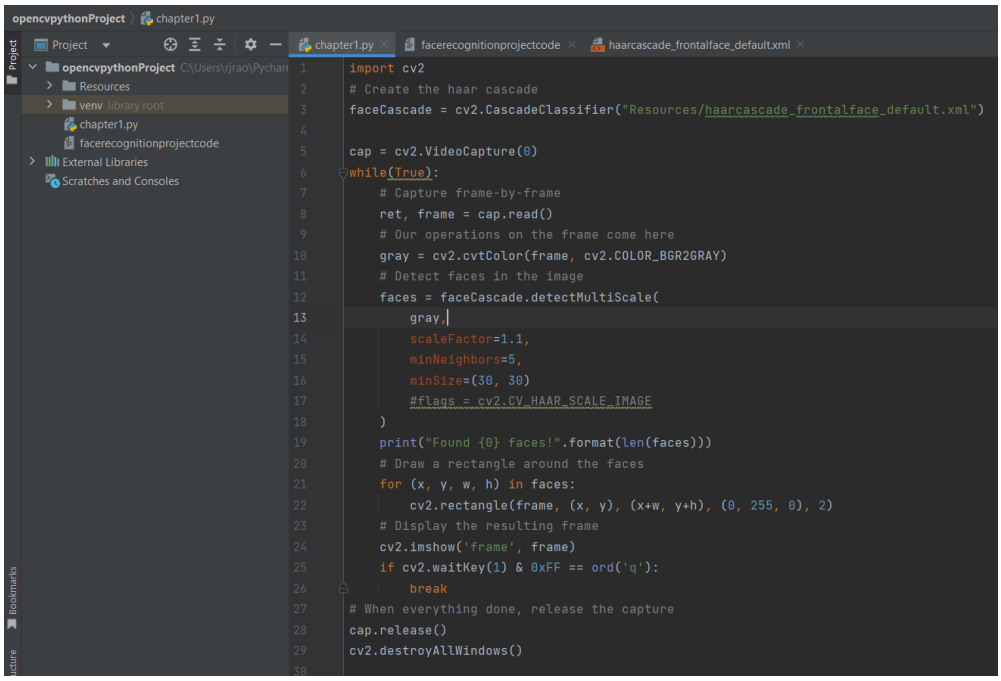
3. **Apply face cascade on webcam frames:**

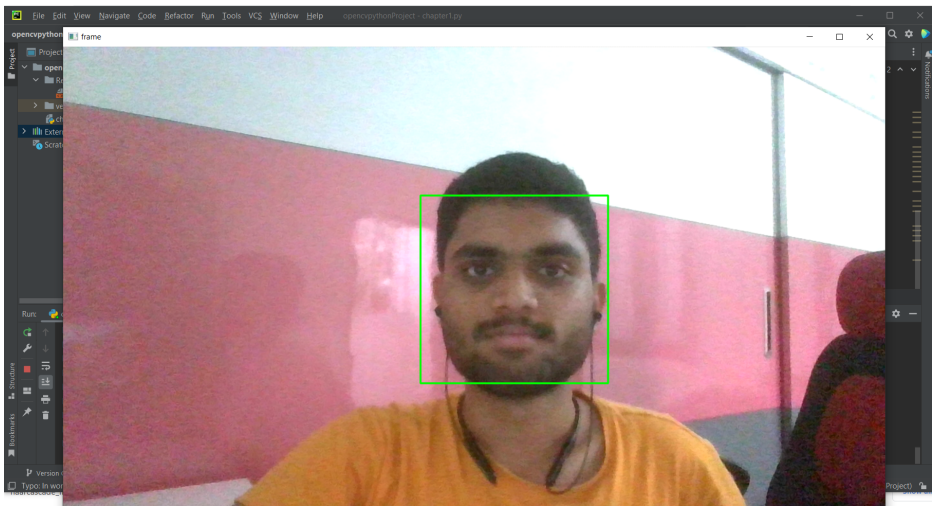
```
cap = cv2.VideoCapture(0)
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Detect faces in the image
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30)
        #flags = cv2.CV_HAAR_SCALE_IMAGE
    )
    print("Found {0} faces!".format(len(faces)))
    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    # Display the resulting frame
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

4. **Release the capture frames:**

```
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Screenshots of code and face detection:-





Chapter 7. Results

6.1 Source Code for detecting faces in images:-

```
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Read the input image
img = cv2.imread('test.jpg')
# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

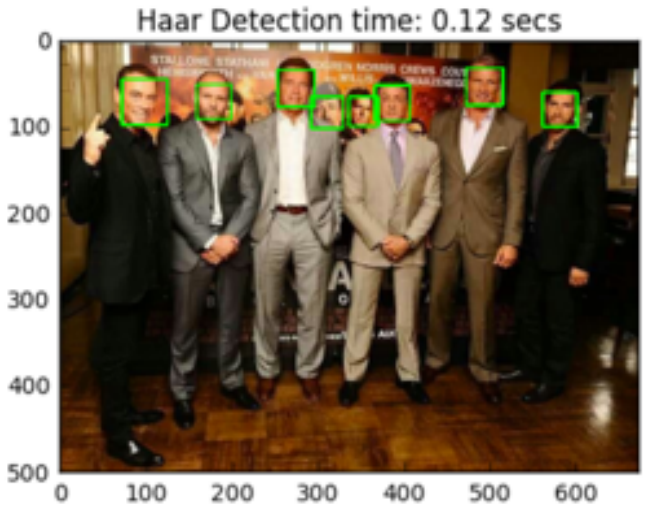


Image Face detection

6.2 Source code for detecting faces in videos:-

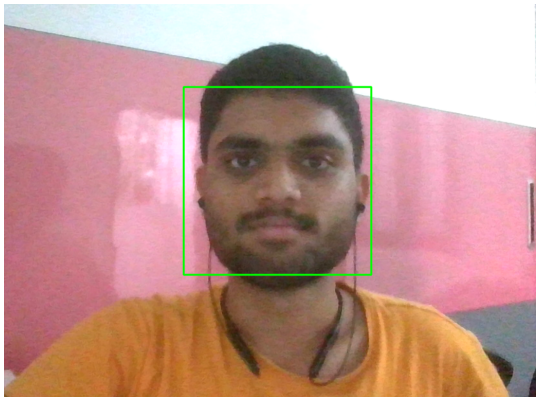
Videos are basically made up of frames, which are still images. So we perform the face detection for each frame in a video.

```
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# To capture video from webcam.
cap = cv2.VideoCapture(0)
# To use a video file as input
# cap = cv2.VideoCapture('filename.mp4')

while True:
    # Read the frame
    _, img = cap.read()
    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Detect the faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
    # Draw the rectangle around each face
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
    # Display
    cv2.imshow('img', img)
    # Stop if escape key is pressed
    k = cv2.waitKey(30) & 0xff
    if k==27:
        break
# Release the VideoCapture object
cap.release()
```



Webcam face detection

Chapter 8. Scope of the Project

- To identify and verify terrorists at airports, railway stations and malls the face recognition technology will be the best choice in India as compared with other biometric technologies since other technologies cannot be helpful in crowded places.
- This technology can also be used effectively in various important examinations such as SSC, HSC, Medical, Engineering, MCA, MBA, B- Pharmacy, Nursing courses, etc. The examinee can be identified and verified using Face Recognition Technique.
- It can also be deployed in the police station to identify and verify the criminals.
- To Identify the number of students in the classroom. It can collect all the number of faces in less than 1 sec. Easy to take attendance.

Chapter 9. Conclusion

- Face recognition technologies have been associated generally with very costly top secure applications. Today the core technologies have evolved and the cost of equipment is going down dramatically due to the integration and the increasing processing power. Certain applications of face recognition technology are now cost-effective, reliable, and highly accurate. As a result, there are no technological or financial barriers to stepping from the pilot project to widespread deployment.
- Government and NGOs should concentrate on and promote applications of facial recognition systems in India in various fields by giving economical support and appreciation