FACIAL KEYPOINTS DETECTION

NAME: G REVANTH REDDY

FACIAL KEYPOINTS DETECTION

ABSTRACT

Biometrics are a way to measure a person's physical characteristics to verify their identity. Face detection is used to detect the faces in the images or live video. It is a part of Object Detection. It is used to detect the faces in real time for surveillance and tracking of a person or objects. In the past few years face recognition, appreciated as one of the most promising applications in the field of image analysis .Facial Keypoints Detection detects the location of keypoints on the face video or Images. The keypoints are ranging from 0 to 68.The keypoints on face starts with 0 i.e... Face outline and end at 68 i.e. Mouth. Facial Keypoints Detection is a computer vision topic and it deals with the problem of detecting distinctive features in human faces automatically.

The primary aim of this project is to determine whether there is any face in an image or not and detect the location of keypoints on the face through a webcam. Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images. Facebook is also using face detection algorithm to detect faces in the images and recognise them. Snapchat and Instagram are using facial keypoints for their face filters to attract people.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	1 - 2
2	LITERATURE SURVEY	3
3	SYSTEM REQUIREMENTS	4
4	PROJECT REQUIREMENTS	5
5	DESIGN	6
6	IMPLEMENTATION	7
7	TESTING	8
8	OUTPUT SCREENS	9
9	CONCLUSION	10
10	BIBLIOGRAPHY	11

LIST OF FIGURES

TOPIC NO.	TITLE	PAGE NO.
2.1	Keypoints Range	3
5.1	HighLevel – UseCase Daigram	6
5.2	Low Level – UseCase Diagram	
6.1	Code Snippet	7
7.1	Test_Output	8
7.2	Final_Output	
8.1	Face and Facial Keypoints Detection	9

1. INTRODUCTION

With the fast development in computer vision area, more and more research works and industry applications are focused on facial keypoints detection. Facial keypoints are the vital areas in the face from which a person's facial expressions — and, therefore, emotions — can be evaluated. They play an important role in developing real-time applications for detecting drowsiness, analysing biometrics and reading a person's emotions. dlib is a toolkit for making real world machine learning and data analysis applications in C++.

4W's and 1'H

Who:

It is helpful for the detecting the faces in real-time for tracking of a person through surveillance or Webcam.

What:

The primary aim of face detection is to detect the location of the keypoints on face Images.

When:

- Facial keypoints detection can be used in various applications like tracking the faces in video or images, analyzing facial expressions, biometrics/facial recognition.

Where:

- It is being used by Snapchat, Instagram to attract people with their face filters.
- It can be used for tacking people who had done any uncertain things

How:

Face Detection works as to detect multiple faces in an image.

Steps that show how face detection and facial Keypoints Detection works:

1. We have to check whether the face is detecting through the webcam or not.
2. The picture in the video is transformed from RGB to Grayscale because it is easy to detect faces in grayscale.
3. Image Segmentation which is used for counter detection or segment the multiple objects in a single image so that the model used can easily or quickly detect the faces in the video.
4. We use dlib.get_frontal_face_detector () to detect whether the face is in the webcam video or not . dlib.shape_predictor() is a tool that takes in an image region containing some object and outputs a set of point locations that define the pose of the object. Here we use the

shape_predictor_68_face_landmarks.dat model to create the predictor object. We then pass the frame and detect the rectangular dimensions.

5. We then can detect the facial keypoints using the landmark points which range from 0 to 68 :
    1. 0 – 26 - Face Outline from Eyebrows
    2. 27 – 35 – Nose
    3. 36 – 41 – Right Eye
    4. 42 – 47 – Left Eye
    5. 48 – 68 – Mouth
6. The next Step is to give the coordinates of x1, x2, y1, y2 which makes a circle in the video to show the keypoints of the face.
7. Finally, If there is a face in the video opened via webcam then the rectangular box which detects whether the face is present or not and the circular keypoints around the face.

2. LITERATURE SURVEY

In field of computer vision research, one of the most important branch is Face recognition. It's target is to verifying the identity of an individual using their face. For the purpose of developing an advanced face recognition algorithm, Detection of facial key points is the basic and very important task, basically it is about finding out the location of specific key points on facial images. These key points can be mouth, noses, left eyes, right eyes and so on. Recognizing facial key points is a difficult to solve. Facial characteristics differ significantly from one person to the next, and even within a single person, there is a lot of variances owing to 3D posture, size, location, viewing angle, and lighting circumstances.

Facial key points can be used in a variety of machine learning applications from face and emotion recognition to commercial applications like the image filters popularized by Snapchat. Facial keypoints (also called facial landmarks) are the small dots shown on each of the faces in the Web camera. While testing the image through a webcam, there will be a 68 keypoints, with coordinates (x, y), for that face. These keypoints mark important areas of the face: the eyes, corners of the mouth, the nose, etc.

In the below image, these keypoints are numbered, and you can see that specific ranges of points match different portions of the face.

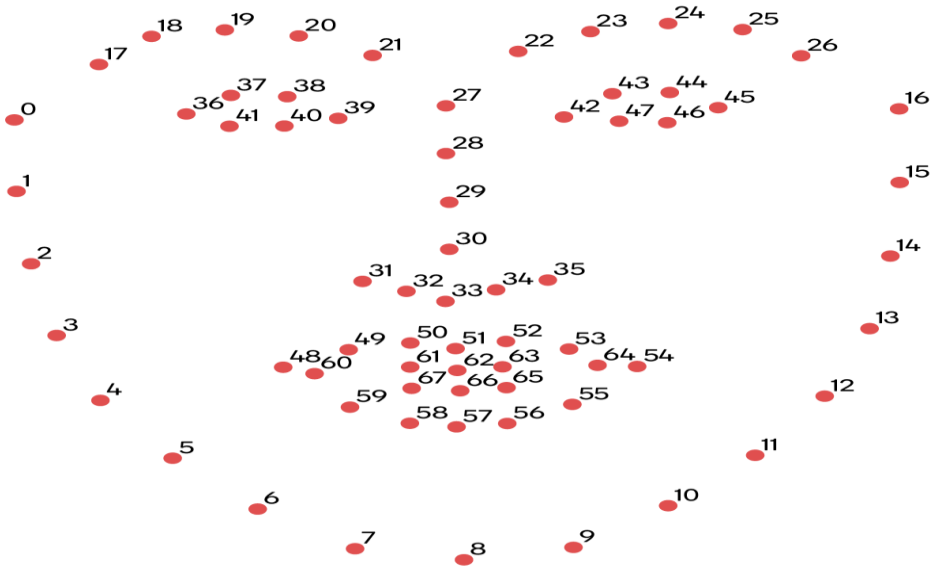

Figure 2.1 : Keypoints Range

3. SYSTEM REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

Operating System **:** Windows 10 and below, Linux. Mac OS X, Unix

Programming Language **:** Python 3.8 & below

Python

Python is high-level programming language like a Perl, Ruby etc. Which are used as a scripting language. It was conceived by Guido van Rossum in 1989. Free, Python is product of open source. People allows to use it in business or commercial without any charge.

1. Easy to read, Syntax in Python is clear and readable. Beginner can be easily to read and handle Python's coding very well.
2. Reusability, Python is easily reused modules and packages. Peoples can be developed their own library and reused it later project.
3. Object-Oriented Programming. Unlike scripting language, Python is designed to be object-oriented. OO programming means you can implement using idea of inheritance and polymorphism.

OpenCV

OpenCV is a synonym of Open Computer Vision Library, which has at least 500 algorithms, documentation and sample code for real time computer vision. OpenCV is originally developed by Intel and launched in 1999.

3.2 HARDWARE REQUIREMENTS

Hardware **:** Pentium Based System with a minimum of P4

RAM **:** 1GB(minimum)

Pentium 4

Pentium 4 is the intel processor that was released in the November 2000.The P4 processor has a viable clock speed that now exceeds 2GHz – as compared to the 1GHz of the Pentium3.

RAM

Random Access Memory is a form of computer storage that stores data and machine code currently being used. A RAM device allows data item to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory

4. PROJECT REQUIREMENTS

4.1 Functional Requirements

Sequence of Operations

1. We have to check whether the face is detecting through the webcam or not.
2. The picture in the video is transformed from RGB to Grayscale because it is easy to detect faces in grayscale.
3. Image Segmentation which is used for counter detection or segment the multiple objects in a single image so that the model used can easily or quickly detect the faces in the video.
4. We use dlib.get_frontal_face_detector () to detect whether the face is in the webcam video or not . dlib.shape_predictor() is a tool that takes in an image region containing some object and outputs a set of point locations that define the pose of the object. Here we use the shape_predictor_68_face_landmarks.dat model to create the predictor object. We then pass the frame and detect the rectangular dimensions.
5. We then can detect the facial keypoints using the landmark points which range from 0 to 68 :
    - 0 – 26 - Face Outline from Eyebrows
    - 27 – 35 – Nose
    - 36 – 41 – Right Eye
    - 42 – 47 – Left Eye
    - 48 – 68 – Mouth
6. The next Step is to give the coordinates of x1, x2, y1, y2 which makes a circle in the video to show the keypoints of the face.
7. Finally, If there is a face in the video opened via webcam then the rectangular box which detects whether the face is present or not and the circular keypoints around the face.

4.2 Non Functional Requirements

Performance Requirements

The performance is independent on the Video Input. Performance depends solely on the accuracy of the classifier which does the prediction.

5. DESIGN

5.1 High Level Requirements

Webcam will be opened from that a face will be detected and location of keypoints will be detected. If there is a face it displays “a rectangular box(Face Detection)” and “Circular points(Facial Keypoints Detection)” around the face.

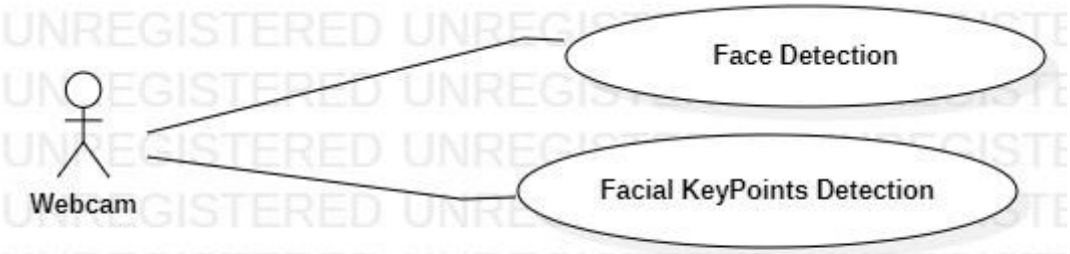

Figure 5.1: Usecase Diagram

5.2 Low Level Requirements

Webcam will be opened from that a face will be detected and location of keypoints will be detected. If there is a face it displays “a rectangular box(Face Detection)” and “Circular points(Facial Keypoints Detection)” around the face. For facial keypoints , it detects eyes,nose,mouth and face outline from eyebrows.The keypoints range from 0 to 68.

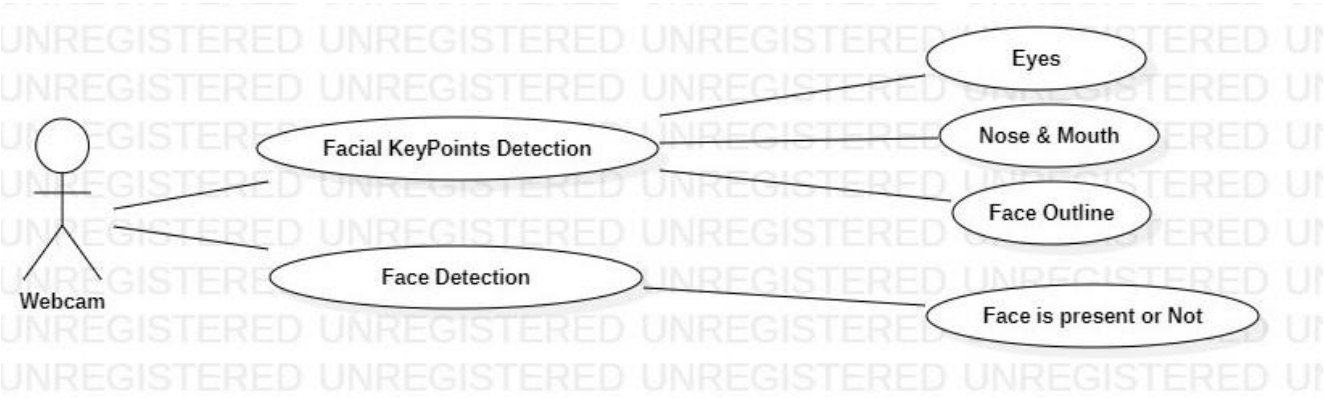

Figure 5.2: Usecase Diagram

6. IMPLEMENTATION

6.1 CODE SNIPPET

```
import cv2
import numpy as np
import dlib
class FKD:
    def FacialKeypoint(self):
        webcam = cv2.VideoCapture(0)
        detector = dlib.get_frontal_face_detector()
        predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
        while True:
            _, frame = webcam.read()
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = detector(gray)
            for face in faces:
                x1 = face.left()
                y1 = face.top()
                x2 = face.right()
                y2 = face.bottom()
                cv2.rectangle(frame, (x1, y1), (x2, y2), (255,255,255), 1)
                landmarks = predictor(gray, face)
                # print(landmarks)
                for n in range(0, 68):
                    x = landmarks.part(n).x
                    y = landmarks.part(n).y
                    # print(x,y)
                    cv2.circle(frame, (x, y), 2, (0, 225, 0), -1)
            cv2.imshow("Frame", frame)
            if cv2.waitKey(20) & 0xFF == ord('e'):
                break
        webcam.release()
        cv2.destroyAllWindows()
m = FKD()
m.FacialKeypoint()
```

Figure 6.1: Code Snippet

7. TESTING

Test Case	Description	Output(Passed/Not Passed
TC_01	Face Detection & Facial Keypoints Detection	Passed

Testing Output:

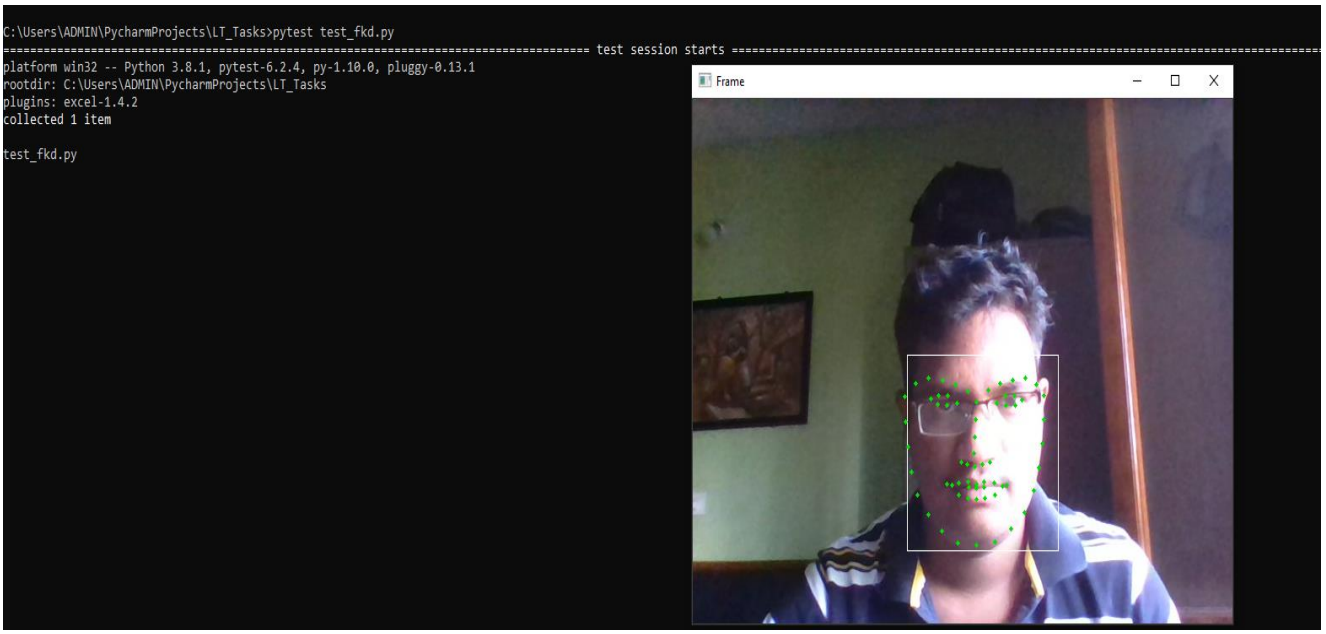

Figure 7.1 : Test_output

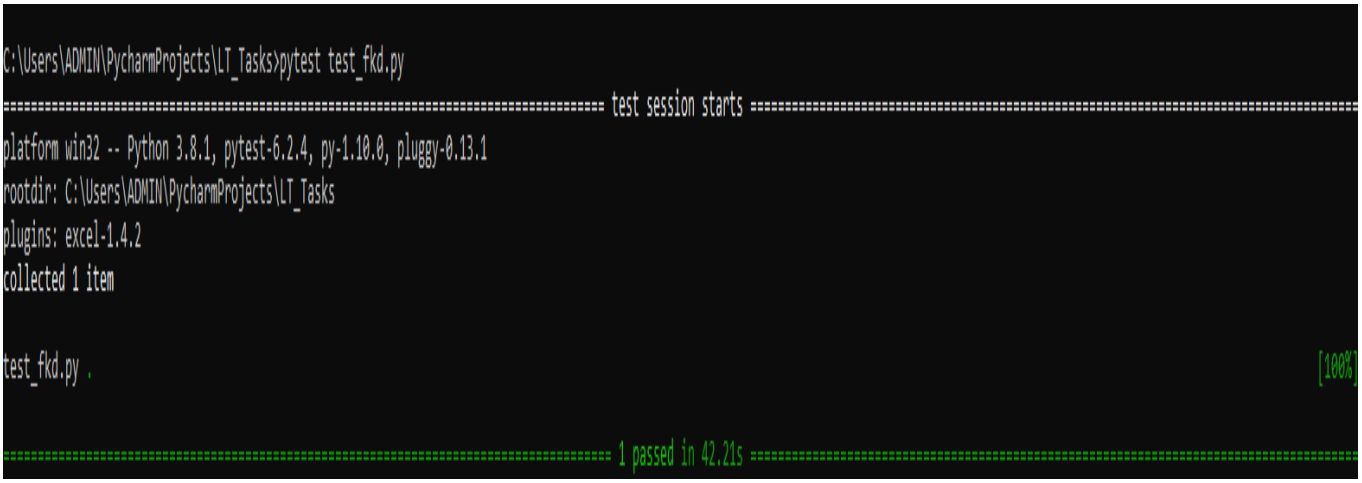

Figure 7.2: Final_output

8. OUTPUT SCREENS

Face Detection & Facial Keypoints Detection :

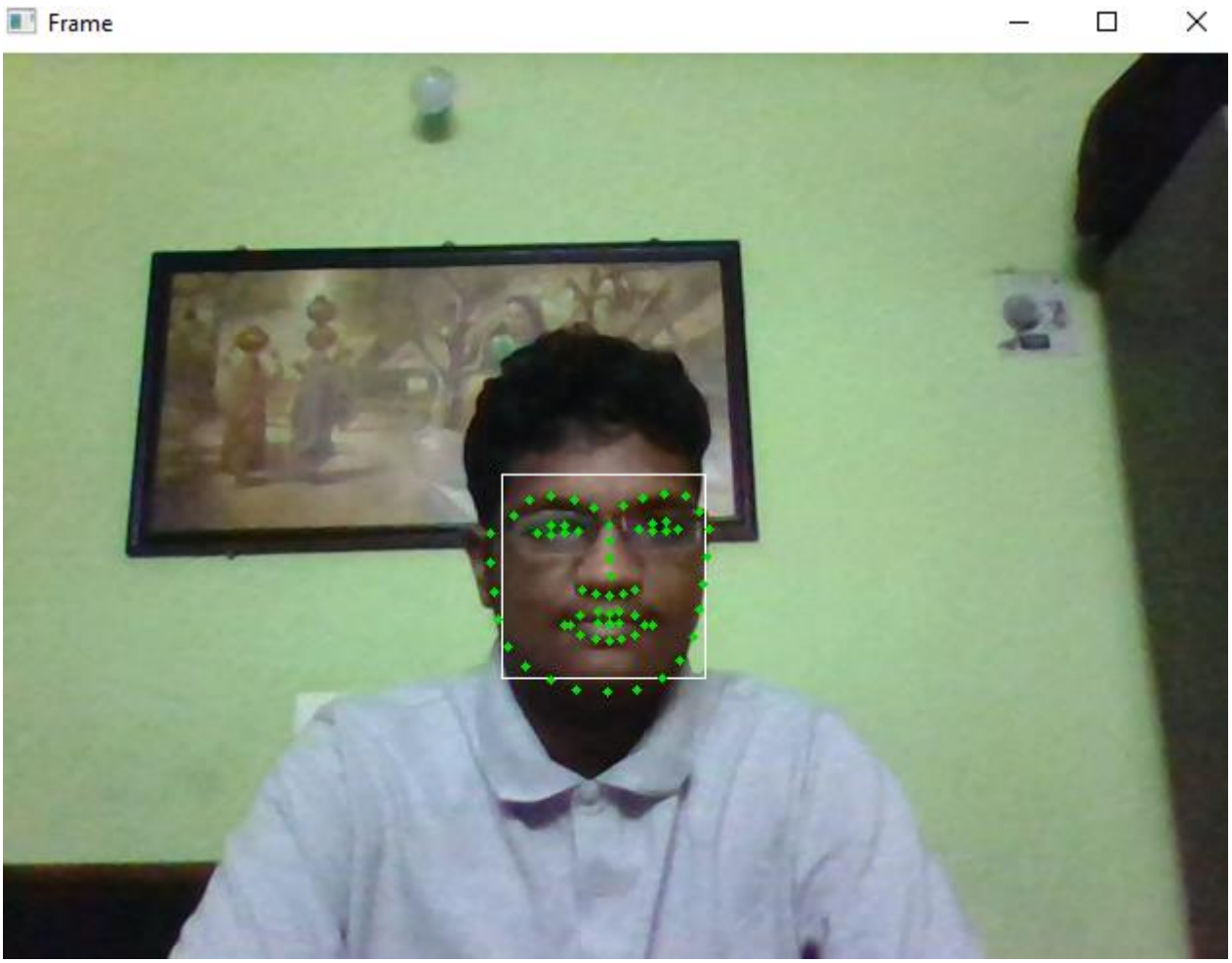

Figure 8.1 Face and Facial Keypoints Detection

Note: Press “e” to exit from the opened webcam window.

9. CONCLUSION

Facial keypoints can be used to align facial images to a mean face shape, so that after alignment the location of facial keypoints in all images is approximately the same. Detection of facial key points is the basic and very important task, basically it is about finding out the location of specific key points on facial images. These key points can be mouth, noses, left eyes, right eyes and so on.

Facial key points can be used in a variety of machine learning applications from face and emotion recognition to commercial applications like the image filters popularized by Snapchat. dlib is a toolkit for making real world machine learning and data analysis applications in C++. While the library is originally written in C++, it has good, easy to use Python bindings. dlib is generally used for face detection and facial landmark detection.

Some applications for facial keypoints detection are:

1. Head Pose Estimation.
2. Face Morphing.
3. Virtual Makeover.
4. Face Replacement.

10. BIBLIOGRAPHY

1. Face Detection For Beginners - https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9
2. http://www.peter-lo.com/Teaching/U08096/200912-B1
3. Facial Landmark Detection - Facial Landmark Detection - Convolutional Neural Networks for Image and Video Processing - TUM Wiki
4. Reference & model(shape_predictor_68_face_landmarks.dat) download from - https://github.com/davisking/dlib-models