CHAPTER 5
CODE AND RESULT

5.1 Code

5.1.1 Detect face from webcam : -

```
from sklearn.neighbors import KNeighborsClassifier
import cv2
import pickle
import numpy as np
import os

video=cv2.VideoCapture(0)
facedetect=cv2.CascadeClassifier('data/haarcascade_frontalface_default.xml')

with open('data/names.pkl', 'rb') as f:
    LABELS=pickle.load(f)
with open('data/faces_data.pkl', 'rb') as f:
    FACES=pickle.load(f)

knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(FACES, LABELS)

while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces=facedetect.detectMultiScale(gray, 1.3 ,5)
    for (x,y,w,h) in faces:
        crop_img=frame[y:y+h, x:x+w, :]
        resized_img=cv2.resize(crop_img, (50,50)).flatten().reshape(1,-1)
        output=knn.predict(resized_img)


        cv2.putText(frame, str(output[0]), (x,y-15),
cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255), 1)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)

    cv2.imshow("Frame",frame)
    k=cv2.waitKey(1)
    if k==ord('q'):
        break
video.release()
cv2.destroyAllWindows()
```

RESULTS

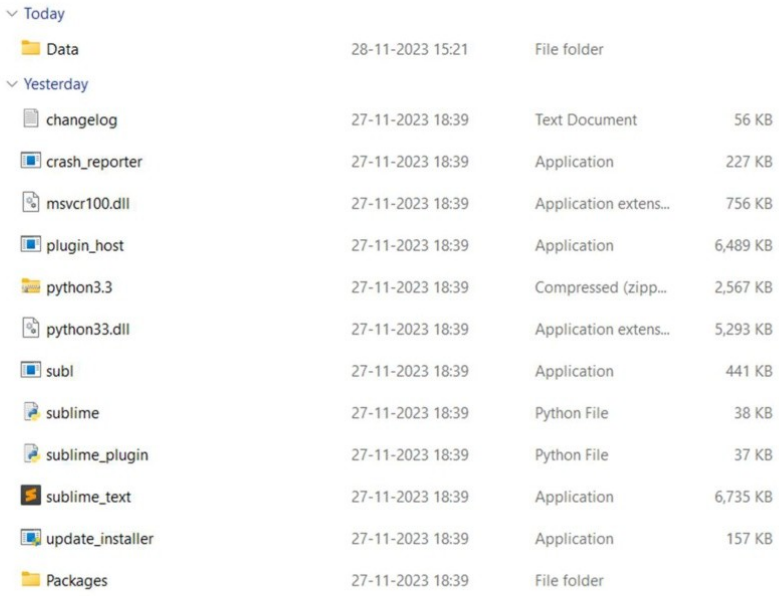

Fig 5.1 Open CV modules

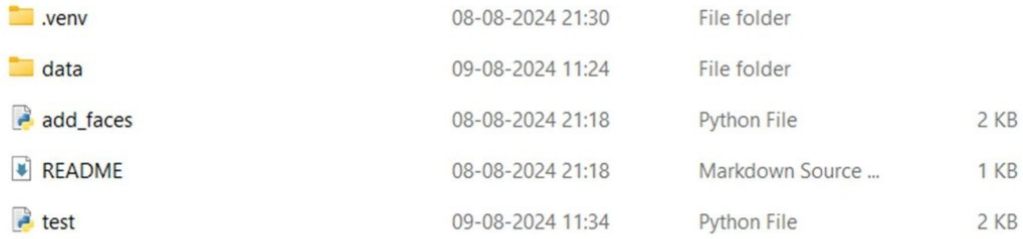

Fig 5.2 Face Detection and Recognition Files

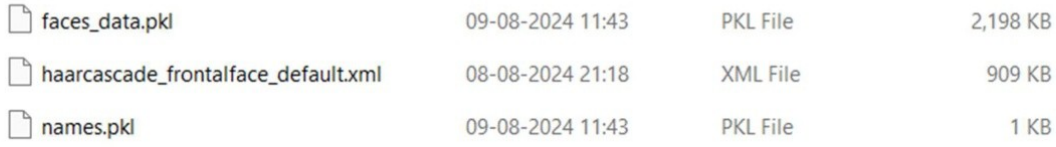

Fig 5.3 Trained Data Files

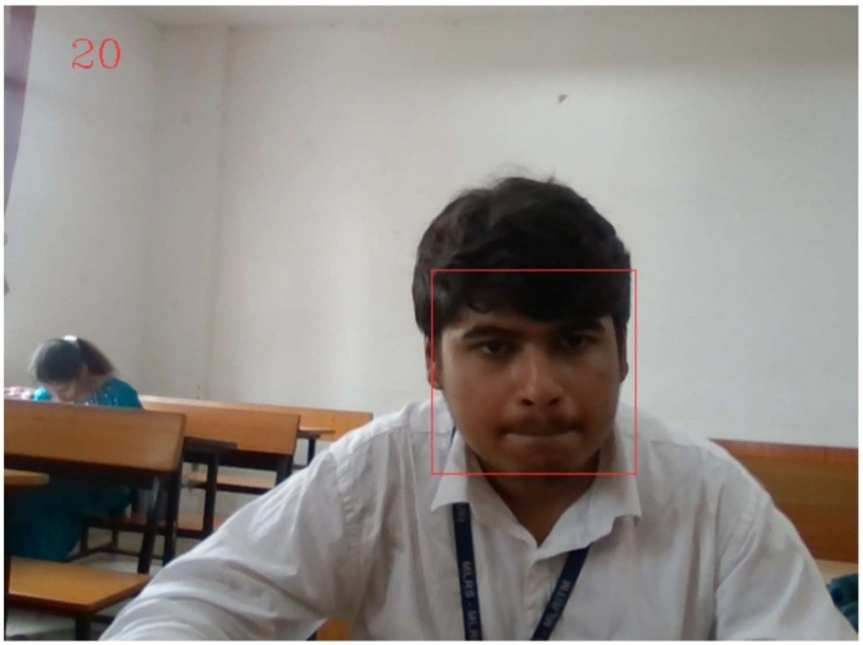

Fig 5.4 training the data using Web Cam

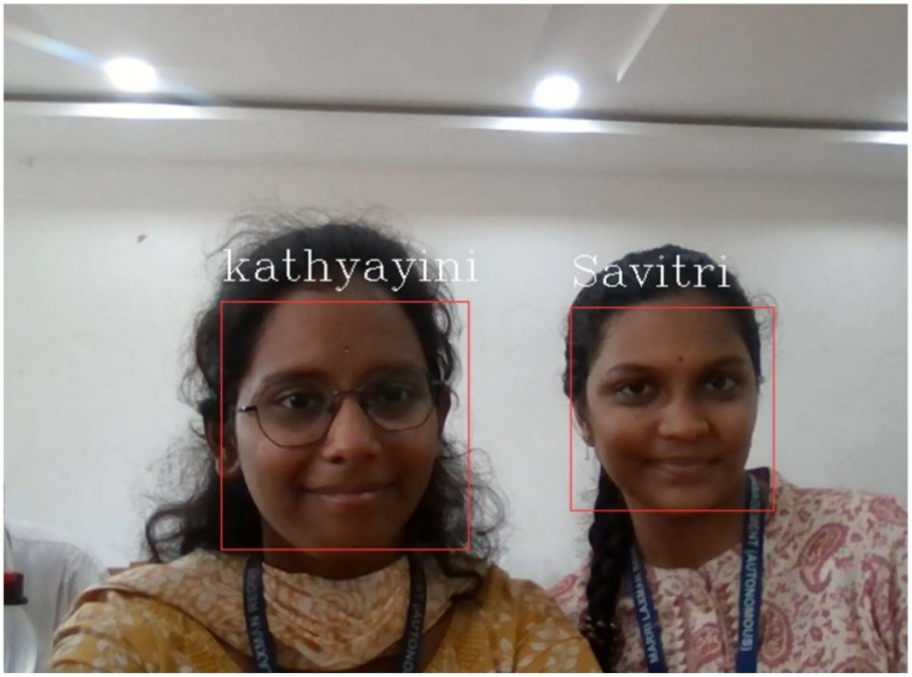

Fig 5.5 Face Recognition using Web Cam

CONCLUSION

face detection and face recognition using OpenCV in Python offer powerful tools for various applications in computer vision. Face detection involves identifying and locating human faces within images or video frames, typically using algorithms like Haar cascades or deep learning-based models such as DNN modules in OpenCV. This process is foundational for many subsequent tasks, including face recognition.

Face recognition builds on face detection, focusing on identifying and verifying individuals based on their facial features. Techniques like Eigenfaces, Fisherfaces, and Local Binary Patterns Histograms (LBPH) are some of the traditional methods, while more advanced approaches leverage deep learning models, such as convolutional neural networks (CNNs).

The combination of these techniques provides a robust framework for applications like security systems, biometric authentication, and human-computer interaction. Despite its effectiveness, challenges such as varying lighting conditions, facial expressions, and occlusions remain, highlighting the importance of ongoing research and development in this field.

Using Python and OpenCV, developers can implement these techniques with relative ease, benefiting from an extensive library of functions and a supportive community. As technology advances, the accuracy and efficiency of face detection and recognition systems will continue to improve, making them increasingly integral to various technological solutions.