Face Recognition

Face detection and recognition system

A facial recognition system is a technology potentially capable of recognizing a human face from a digital image or a video frame against a database of faces. Such a system is typically employed to authenticate users measuring facial features from a given image. This face recognition system can use in various fields such as security system ,attendance system.

Face Recognition

Face detection and recognition system

Facial recognition is a way of identifying or confirming an individual's identity using their face. Facial recognition systems can be used to identify people in photos, videos, or in real-time(web camera).

Facial recognition is a category of biometric security. Other forms of biometric software include voice recognition, fingerprint recognition, and eye retina or iris recognition. The technology is mostly used for security and law enforcement, though there is increasing interest in other areas of use.

Many people are familiar with face recognition technology through the Face-ID used to unlock iPhones. Beyond unlocking phones, facial recognition works by matching the faces of people walking past special cameras, to images of people on a watch list. The watch lists can contain pictures of anyone, including people who are not suspected of any wrongdoing, and the images can come from anywhere — even from our social media accounts

[Face Recognition]

Build on python

Required dependencies

-CMAKE

-DLIB

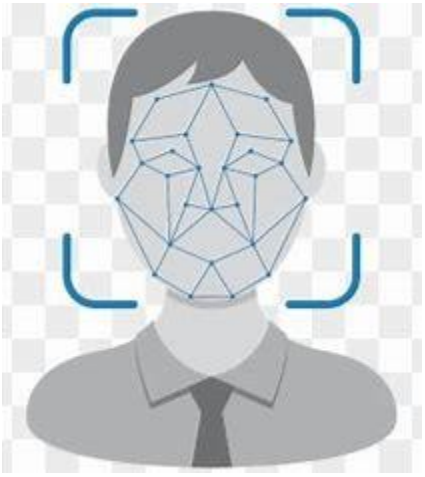
-FACE_RECOGNITION

#What is CMAKE?

CMAKE is an open source, cross-platform family of tools which is design to build, test and package software .CMake gives you control of software compilation process. In other words it is necessary for installing many softwares in our case we use cmake to install face_recognition.

#What is DLIB?

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments.



#What is face_recognition

Face_recognition Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images from the command line!

Face recognition

In [1]:
```
import face_recognition
import cv2
import numpy as np


# Get a reference to webcam #0 (the default one)
video_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.
saurav_image = face_recognition.load_image_file("D:\saurav.jpg")
saurav_face_encoding = face_recognition.face_encodings(saurav_image)[0]

# Load a second sample picture and learn how to recognize it.
bittu_image = face_recognition.load_image_file("D:\DSC_0005.JPG")
bittu_face_encoding = face_recognition.face_encodings(bittu_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings = [
    saurav_face_encoding,
    bittu_face_encoding
    ]
known_face_names = [
    "saurav",
    "bittu"
]

while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()

    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_re
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Find all the faces and face enqcodings in the frame of video
    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame,face_locations)
    #face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

    # Loop through each face in this frame of video
    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encoding
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)

        name = "unknown"

        # If a match was found in known_face_encodings, just use the first one.
        # if True in matches:
        #     first_match_index = matches.index(True)
        #     name = known_face_names[first_match_index]

        # Or instead, use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(known_face_encodings, face_enc
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
```

```
            name = known_face_names[best_match_index]

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FI
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1

    # Display the resulting image
    cv2.imshow('Video', frame)

    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
```