In the name of God

Real-Time Face Recognition: An End-to-End Project

Amirhosein Rezazade

Arian Ahmadifard

Dr.Reza Javanmard

Computer Vision 982

June 2020

Abstract

The key challenge of face recognition is to develop effective feature representations for reducing intra-personal variations while enlarging inter-personal differences. In this paper, we show that it can be well solved with deep learning and using both face identification and verification signals as supervision. The Deep Identification-verification features are learned with carefully designed deep convolutional networks. The face identification task increases the inter-personal variations by drawing features extracted from different identities apart, while the face verification task reduces the intra-personal variations by pulling features extracted from the same identity together, both of which are essential to face recognition. The learned features can be well generalized to new identities unseen in the training data.

Project description

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!

Things used in this project

Hardware components :

1- ultrasonic sensor
2- raspberry pi 3
3- raspberry pi 3 camera module
4- Speaker

Software apps and online services :

1- Python 3
2- Linux Debian
3- face recognition library
4- Windows form application using C#
5- Microsoft Access Database

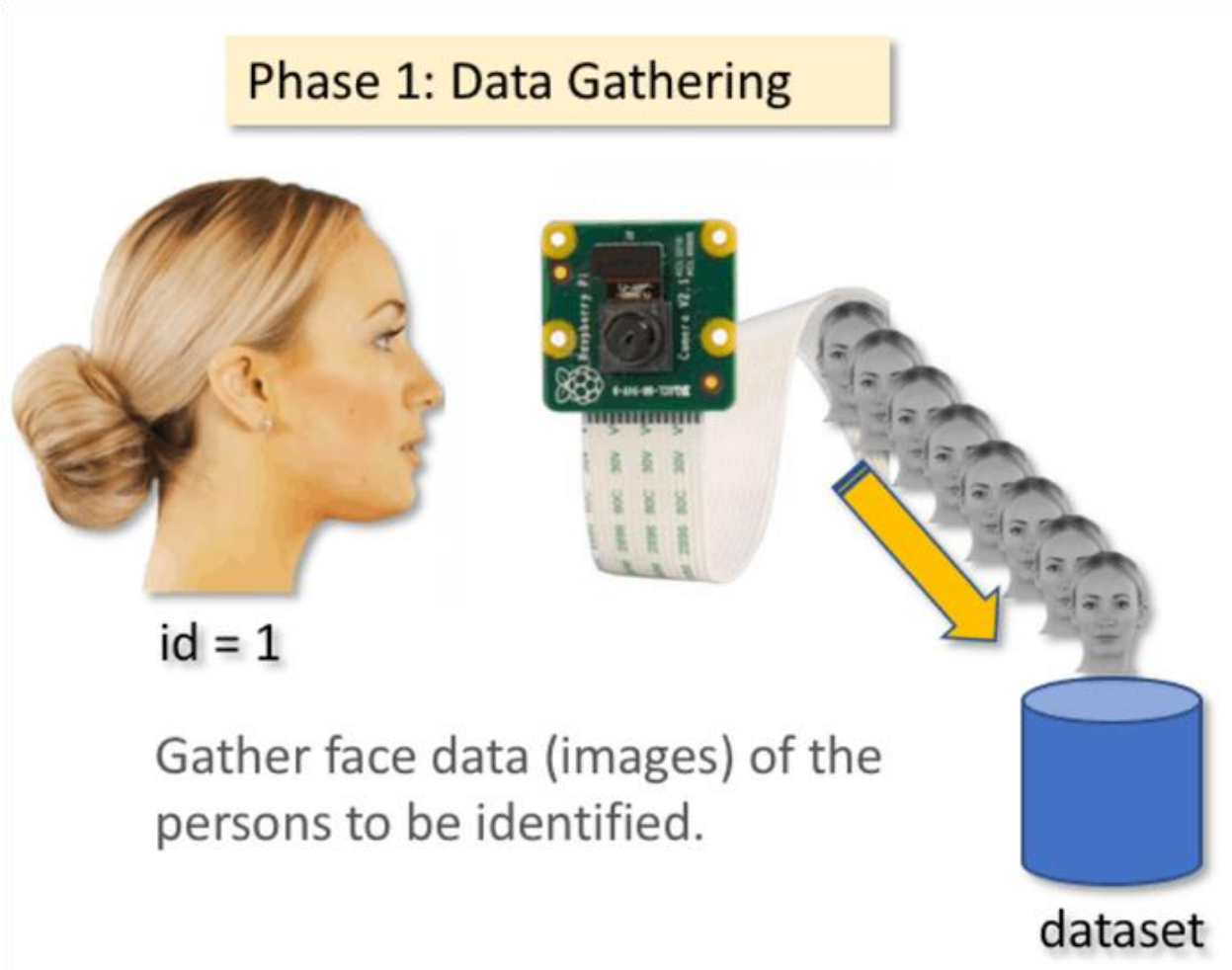
Project Components Overview :

Story

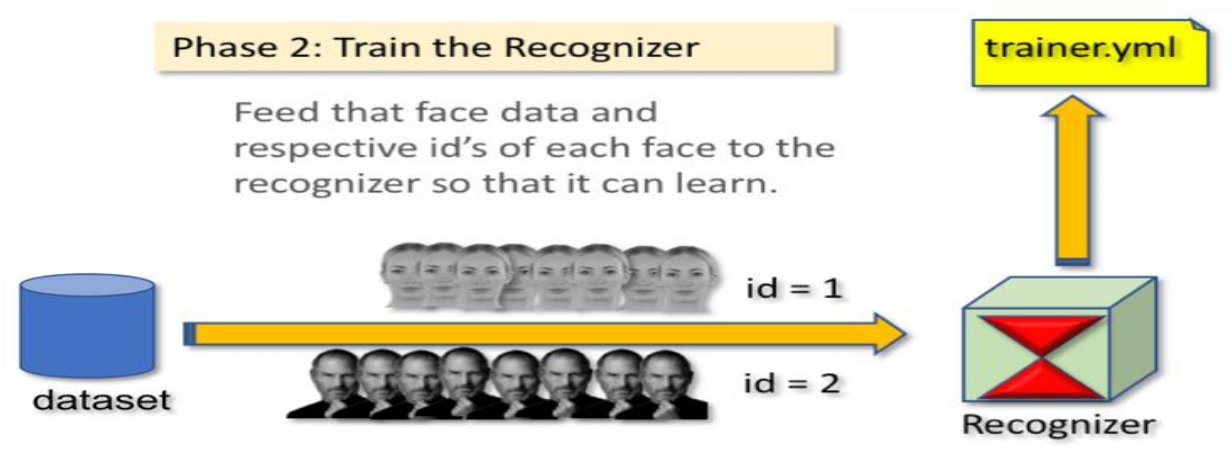
To create a complete project on Face Recognition, we must work on 3 very distinct phases:

1- Face Detection and Data Gathering
2- Train the Recognizer
3- Face Recognition

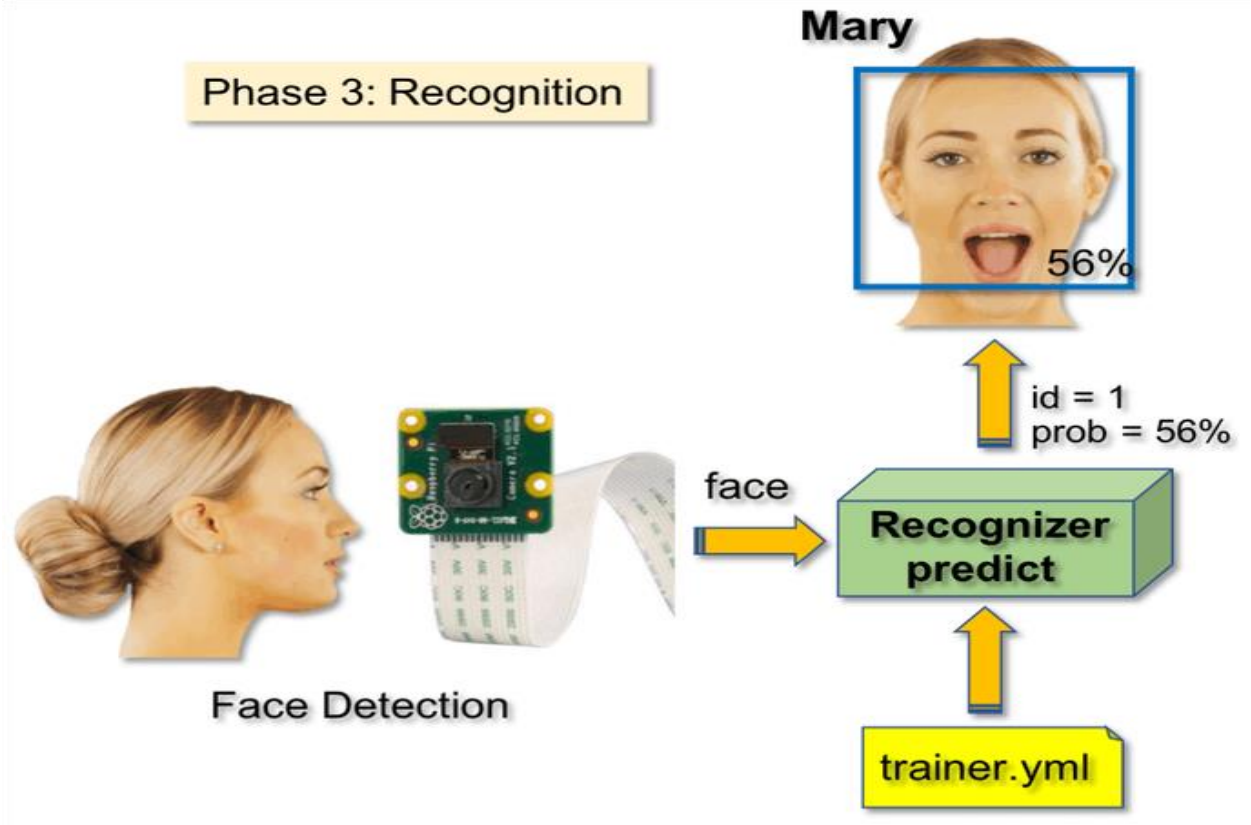
Let's start the first phase of our project. What we will do here, is starting from last step (Face Detecting), we will simply create a dataset, where we will store for each id, a group of photos in gray with the portion that was used for face detecting.



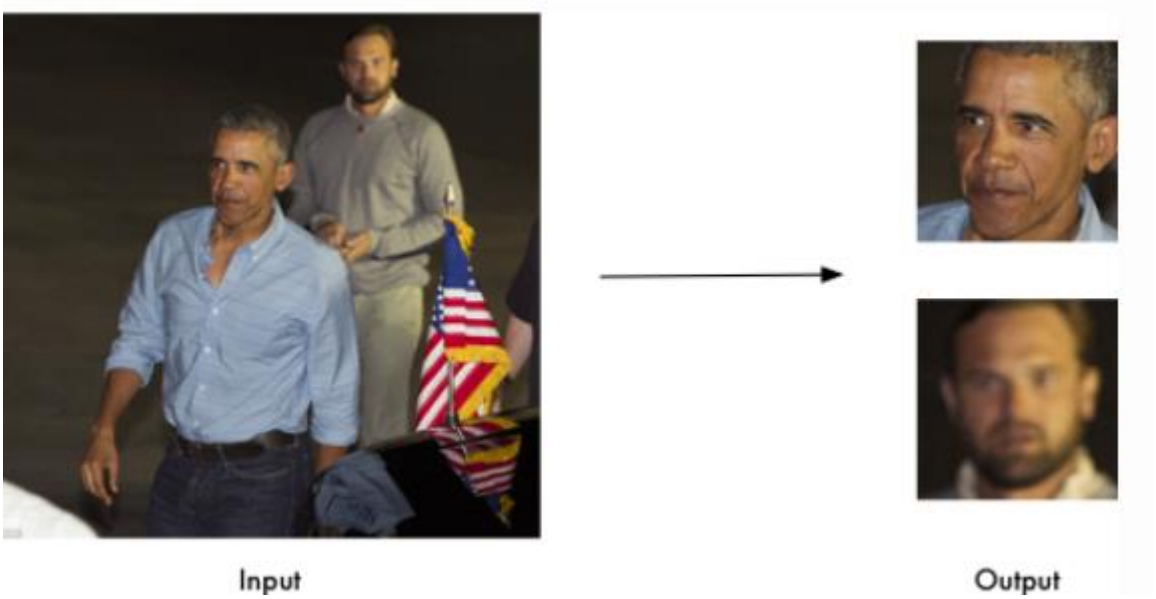
On this second phase, we must take all user data from our dataset



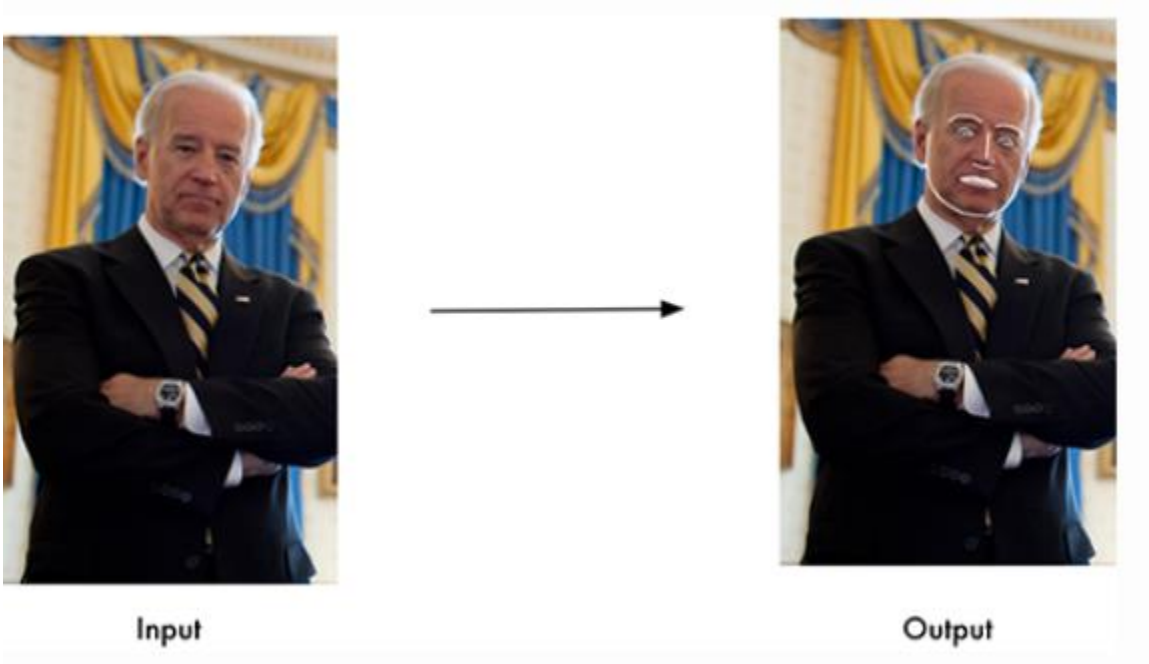
Here, we will capture a fresh face on our camera and if this person had his face captured and trained before, our recognizer will make a "prediction" returning its id and an index, shown how confident the recognizer is with this match



Find all the faces that appear in a picture:

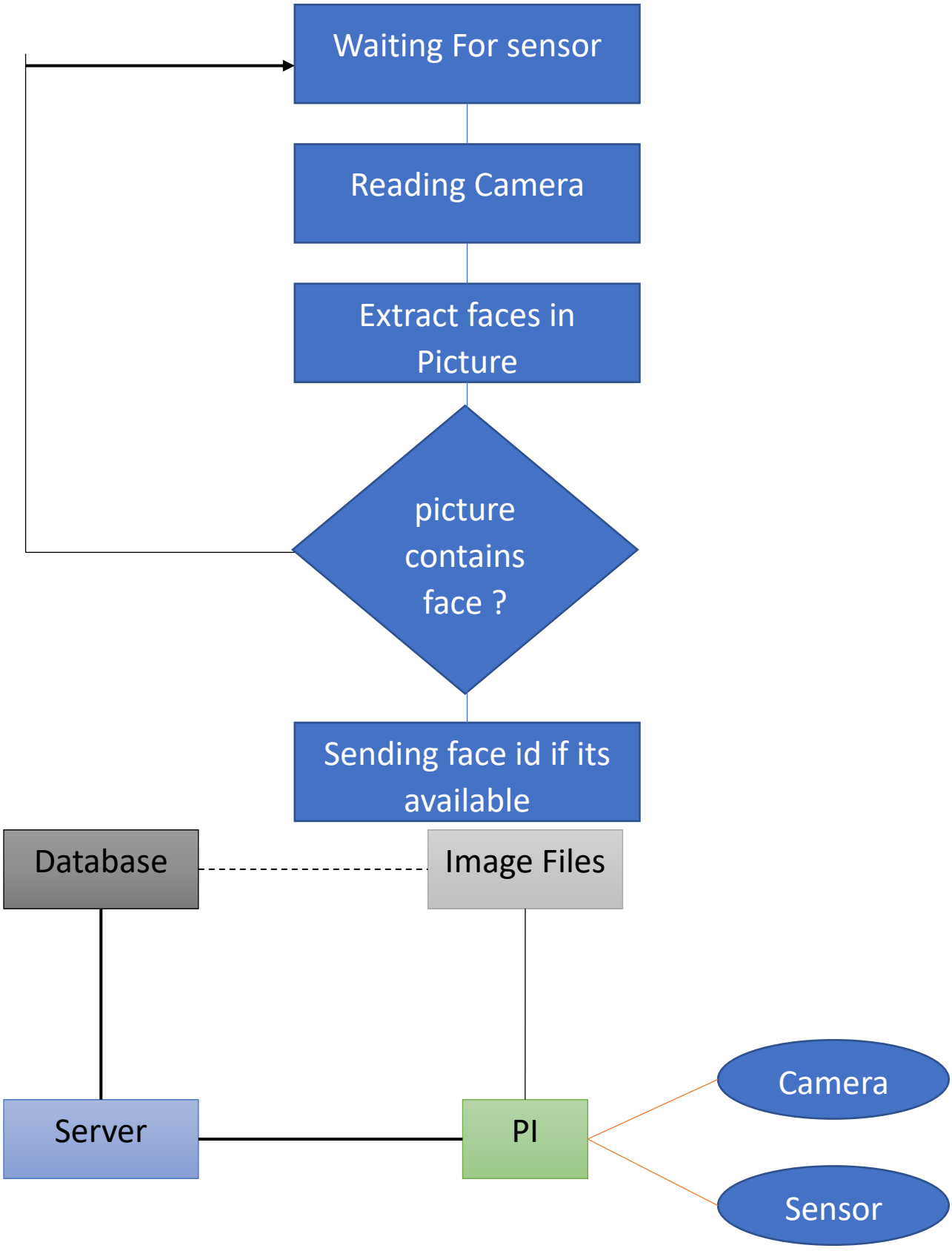


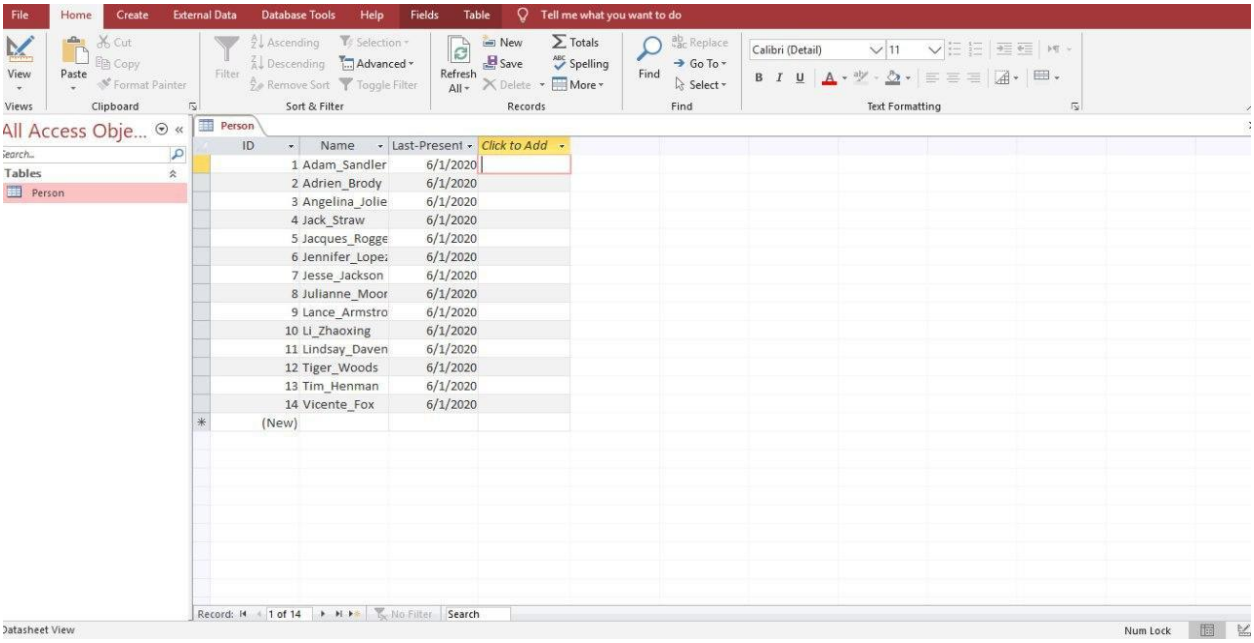
Get the locations and outlines of each person's eyes, nose, mouth and chin:



Project Plan

First, we run our face code on pi3. Every image file is named by person id and the pi will send person id to the server as it face has been detected.





Code :

```
import face_recognition
import cv2
import numpy as np
import threading
import socket
import time
from os import listdir
from os.path import isfile, join
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

#Application is Running :
Running = True

#DataSet Patch :
patch = "dataset"

#Creating a list of files in dataset folder :
onlyfiles = []
try:
    print("Listing files...")
    onlyfiles = [f for f in listdir(patch) if isfile(join(patch, f))]
except:
    print("Error listing files")
```

```
#Loading Dataset (Images and Labels) :
known_face_names = []
known_face_encodings = []
for filename in onlyfiles:
    try:
        # Load a sample picture and learn how to recognize it.
        image = face_recognition.load_image_file(patch + "/" + filename)
        encodedImage = face_recognition.face_encodings(image)[0]
        #and add it to known_face_encodings list :
        known_face_encodings.append(encodedImage)
        #Adding Label of the image to known_face_names list :
        Label = filename[0: filename.find('.')]
        known_face_names.append(Label)
        print(filename + "  -  OK  |  Label :" + Label)
    except:
        print(filename + "  -  Bad Image")

if len(known_face_names) == 0:
    print("No", end='')
else:
    print(len(known_face_names), end='')

print(" Valid Image's found\n\nLoadnig...")

if len(known_face_names) == 0:
    exit(0)

# init Socket and Opening Port:
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sck = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Checking Local IP if we are connected to LAN:
localIP = socket.gethostname()

try:
    print("Binding socket ...")
    # Binding Socket on IP:
    serversocket.bind((localIP, 1114))
    class SocketThread(threading.Thread):
        def run(self):
            global sck
            global Running
            print("{} Thread started!".format(self.getName()))
            # Waitnig for connection for 5 sec :
            print("Listening on " + localIP + " , port 1111")
            serversocket.listen()
            serversocket.settimeout(5)
            #Checking while Application is running :
            while(Running):
                try:
                    sck, addr = serversocket.accept()
                except:
                    continue
                sck.send(("Connected\r\n").encode())
                print('Socket Connected')
            print("{} Thread finished".format(self.getName()))

    #Starting Socket Thread :
```

```
    socketthread = SocketThread(name = "Socket")
    socketthread.start()
except:
   print("Error: Socket error")



def send(list):
    try:
        data = {"count" : len(list),"list":list}
        sck.send(str(data).encode())
    except:
        print('',end ='')


#Setup sensore :
TRIG = 3
ECHO = 2
LED = 17
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
GPIO.setup(LED,GPIO.OUT)
GPIO.output(TRIG, False)
GPIO.output(LED, False)

def WaitForSensore():
    GPIO.output(LED, False)
    while 1 :
        time.sleep(0.3)
        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        while GPIO.input(ECHO)==0:
          pulse_start = time.time()

        while GPIO.input(ECHO)==1:
          pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance, 2)

        if (distance > 2) & (distance < 60):
            GPIO.output(LED, True)
            break


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

# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
old_names = []

print("\n\nApplication Started")
```

```
while Running:
    # Release handle to the webcam
    try:
        video_capture.release()
    except:
        print()
    WaitForSensore()
    video_capture = cv2.VideoCapture(0)
    # Grab a single frame of video
    ret, frame = video_capture.read()

    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)
    #small_frame = frame


    # Convert the image from BGR color (which OpenCV uses) to RGB color (which
face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]

    # Find all the faces and face encodings in the current frame of video
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # 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 matches.count(True) == 1:
            first_match_index = matches.index(True)
            name = known_face_names[first_match_index]
        else:
            if matches.count(True) > 1: #Or instead
                face_distances = face_recognition.face_distance(known_face_encodings,
face_encoding)
                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                    name = known_face_names[best_match_index]

        face_names.append(name)

    if not old_names == face_names:
        FaceCount = len(face_names)
        if not FaceCount==0:
            print(FaceCount,end ='')
            print(" face detected :")
            for name in face_names:
                print("\t" + name)
            print("\n\n")
        else:
            print("Waiting fo face...")
        old_names = face_names
        send(face_names)
```