Image Processing

Project Documentation

Project Name

Secure Door

Project Group

Salma Mohammed

Omnia Ahmed

Omnia Saeed

Ebtesam Abdel Aziz

Amira Abd El-wanis

Algorithm 1	Algorithm2
It is built using open cv and numpy modules	Built using face_recognition module
Accuracy less than 75%	High accuracy reaches 95%
Longer, a little bit more complicated and a bit faster	Shorter, less complicated and a little bit slower
The module used doesn't contain deep learning in its implementation,a training code is needed	The module used contains deep learning in its implementation
Source: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_api.html	Source: https://pypi.org/project/face_recognition/

Algorithms implementation:

1.1.1 Algorithm 1

1.1.1.1 Training code:

```
1.  import cv2
2.  import os
3.  import numpy as np
4.  from PIL import Image
5.  import pickle
6.
7.  BASE_DIR = os.path.dirname(os.path.abspath(__file__))
8.  #print(BASE_DIR)
9.  image_dir = os.path.join(BASE_DIR, "images")
10. #print(image_dir)
11.
12. face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_
    alt2.xml')
13. #print (face_cascade)
14. recognizer = cv2.face.LBPHFaceRecognizer_create()
15.
16. current_id = 0
17. label_ids = {}
18. y_labels = []
19. x_train = []
20.
21. for root, dirs, files in os.walk(image_dir):
22.     for file in files:
23.        # print ("1")
24.         if file.endswith("png") or file.endswith("JPG") or file.endswith("jp
    g"):
25.             #print("0")
26.             path = os.path.join(root, file)
27.             label = os.path.basename(root).replace(" ", "-").lower()
28.             #print(label, path)
29.             if not label in label_ids:
30.                 label_ids[label] = current_id
31.                 current_id += 1
32.                 id_ = label_ids[label]
33.                 print("id_", id_)
34.                 #y_labels.append(label) # some number
35.                 #x_train.append(path) # verify this image, turn into a NUMPY
     arrray, GRAY
36.                 pil_image = Image.open(path).convert("L") # grayscale
37.                 size = (550, 550)
38.                 final_image = pil_image.resize(size, Image.ANTIALIAS)
39.                 image_array = np.array(final_image, "uint8")
40.                 print(len(image_array))
```

```
                faces = face_cascade.detectMultiScale(image_array, scaleFactor=1.5, minNeighbors=5)
                print(len(faces))
                for (x,y,w,h) in faces:
                        roi = image_array[y:y+h, x:x+w]
                        x_train.append(roi)
                        y_labels.append(id_)


print("y_labels",y_labels)
print("x_train", x_train)

with open("pickles/face-labels.pickle", 'wb') as f:
    pickle.dump(label_ids, f)

recognizer.train(x_train, np.array(y_labels))
recognizer.save("recognizers/face-trainner.yml")
```

1.1.1.2 Idenification code:

```
import numpy as np
import cv2
import pickle

face_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_frontalface_alt2.xml')
eye_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('cascades/data/haarcascade_smile.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("./recognizers/face-trainner.yml")

labels = {"person_name": 1}
with open("pickles/face-labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    labels = {v:k for k,v in og_labels.items()}

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    gray  = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
    for (x, y, w, h) in faces:
        #print(x,y,w,h)
        roi_gray = gray[y:y+h, x:x+w] #(ycord_start, ycord_end)
        roi_color = frame[y:y+h, x:x+w]

        # recognize? deep learned model predict keras tensorflow pytorch scikit learn
        id_, conf = recognizer.predict(roi_gray)
        if conf>=4:# and conf <= 85:
            print(id_)
            print(labels[id_])
            font = cv2.FONT_HERSHEY_SIMPLEX
            name = labels[id_]
            color = (0, 0, 0)
            stroke = 2
            cv2.putText(frame, name, (x,y), font, 1, color, stroke, cv2.LINE_AA)

```

```
        img_item = "7.png"
        cv2.imwrite(img_item, roi_color)

        color = (255, 0, 0) #BGR 0-255
        stroke = 2
        end_cord_x = x + w
        end_cord_y = y + h
        cv2.rectangle(frame, (x, y), (end_cord_x, end_cord_y), color, stroke
)
        #subitems = smile_cascade.detectMultiScale(roi_gray)
        #for (ex,ey,ew,eh) in subitems:
        #   cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    # Display the resulting frame
    cv2.imshow('frame',frame)
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

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

1.1.2 Algorithm 2:

```
import cv2
import face_recognition
import time
import serial

#establish connection with arduino on port COM4
arduino = serial.Serial('COM4', 9600)
time.sleep(2)
print("Connection to arduino...")

                                    #training
#reading my pic
person1 = face_recognition.load_image_file("C:/Users/Ebtesam/Desktop/Project
/person1.jpg")
#encoding face features from my pic.
encoding_person1 = face_recognition.face_encodings(person1)[0]

person2 = face_recognition.load_image_file("C:/Users/Ebtesam/Desktop/Project
/person2.jpg")
encoding_person2 = face_recognition.face_encodings(person2)[0]


while(True):
    # Capture frame by frame
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    cv2.imshow('frame',frame)

    #saving a test frame
    img = "test.png"
    cv2.imwrite(img, frame)

    #encode features of test pic.
    picture = face_recognition.load_image_file(img)
    encoding_picture1 = face_recognition.face_encodings(picture)

    if len(encoding_picture1) > 0:
         encoding_picture = face_recognition.face_encodings(picture)[0]

    else:
        encoding_picture = encoding_person1 + 1
    #solved the problem of non-existing of any face in the frame
```

```
    #close the door if no one is there
                            #identifing
    # compare features
    r = face_recognition.compare_faces([encoding_person1], encoding_picture)

    if r[0] == True:
        data = str.encode ('1')
    else:
        r = face_recognition.compare_faces([encoding_person2], encoding_pict
ure)
        if r[0] == True:
            data = str.encode ('2')
        else:
            data = str.encode ('0')

    print (data)

    #sending data to arduino
    arduino.write(data)

    #quit on pressing Q key
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

#close every thing
cap.release()
arduino.close()
cv2.destroyAllWindows()
```

Arduino code:

```
#include<Servo.h>

Servo Door;

void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
  Door.attach(5); //Attach Vertical Servo to Pin 5
  Door.write(0);
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available() > 0) {
       char data = Serial.parseInt();
        Serial.print("py");
    if (data == 0){
     Serial.print("0");
      Door.write (0);
    }
    else{
    //if (data == 1) {
       Serial.print("1");
      Door.write (90);
    }
}
}
```

P.S.: We used Algorithm 2 because of its high accuracy