VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

PROJECT REVIEW 3

TEAM MEMBERS

19BCE2425 - P VIJAY NARASIMHA REDDY

19BCE2370 - B SAI YASHWANTH REDDY

19BCE0570 - S SAI TEJA

19BCE2069 - P LOHITH SASI ANVESH

TITLE

Face recognition

INTRODUCTION

Now a days the digitalization of the world created many benefits to the society. And due to this digitalization, we are meeting with many challenges to make our living more luxurious. Due to this, we are in a rapid development phase in all kinds of fields. One of developing fields is image processing which we are using in our daily day lives like when we take photos in our mobile phone we increase the brightness and recently there are also apps which help us to remove or blur the parts we don’t like in the pictures, there is also the possibility of changing the background of the picture like these there are many areas in which the image processing is used by us without knowing that we are doing

it in our daily lives. This paper is about detecting the number of faces in the images to know number of people present in the picture for convenience. We created a python code for detecting the number of faces and storing these faces to see the faces of the people present in the image.
Here in this project we are using python language to code the program to detect the faces in the image. And for the python to detect the image we have to download OpenCV library and the haar cascade classifier for easy calculations. OpenCV library is used for 2D and 3D toolkits, , facial recognition, gesture recognition, HCI, mobile robotics, motion
understanding, object identification, segmentation and recognition, stereopsis stereo vision: depth perception from 2 cameras, motion tracking, augmented reality. We can download OpenCV from the home website available publicly to everyone. The OpenCV library application is written in python language that is to say its primary interface is python but it still retains a less comprehensive though extensive older C interface. It works on a wide variety of hardware platforms, including Intel Atom platform, Intel Core processor family, and Intel Xeon processor family.

OBJECTIVES

Records different faces with names , and seperate different persons photos mixed in a folder

PROBLEM STATEMENT

now a days face recognition and finger print detection such kind of biometrics are used widely in all fields for various purposes.

LITERATURE SURVEY

Face Detection using Haar-cascades in OpenCV

Face Detection using Haar feature based cascade classifiers is quite an efficient facial detection method and a machine learning algorithm based on an system in which a cascade function which is trained using a lot of positive and negative images. These are then used to detect facial features in other images. Firstly, the algorithm requires many positive images and negative images. The positive images are images with facial features and the negative images are images without any facial features. These images are then used to train the classifier and then extract features from it. Each of these features are single values obtained by subtracting the sum of pixels under the white rectangle from sum of pixels under the black rectangle.

All possible sizes and locations of each kernel are used to calculate lots of features. For each feature calculation, find the sum of the pixels under white and black squares. To solve this, the image is needed. However large your image is, it reduces the calculations for a given. It makes things super-fast.

we apply each and every feature on all the training images. For each feature, it finds which will classify the faces to positive and negative. Obviously, there will be errors. We select the features with minimum error rate.

The process involves each image which are given equal weight initially and after each classification, weights of misclassified images are increased.

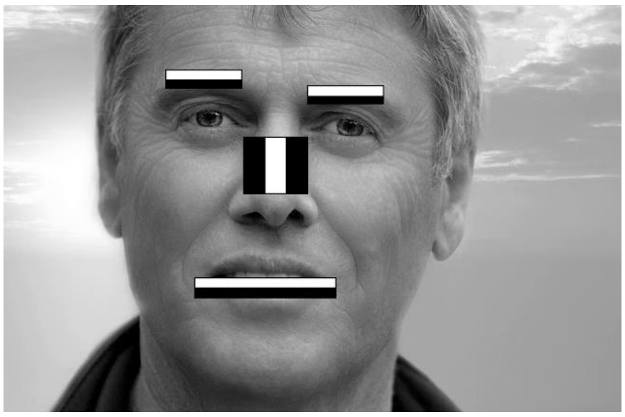
Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found. The final classifier is a sum of these weak classifiers. It is weak because it alone can't classify the image, but together with others forms a strong classifier.

In an image, most of the image is non-face region. So it is a better idea to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions. For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. If a window fails the first stage, discard it. We don't consider the remaining features on it.

OpenCV has a lot of pre-trained classifiers for face, eyes, smile and a lot of other features which help create the face detector by loading the required XML classifiers. Then the input image or the webcam video needs to be loaded in grayscale mode required for real time face detection.

When doing facial recognition which need some images of faces which can obtained by getting datasets from the internet or can create your own dataset. Here I am using my own dataset. Now create the algorithm or function to get or prepare or obtain the training set containing the images for the facial recognition. This function takes or gets the path to the facial image database which is used as an input argument. This can be used to recognise the faces with the help of the webcam. Here, there are two steps involved which are; capturing the video from the webcam and compare them with the dataset using the Fisher face recogniser to train the dataset.

Training and prediction can be easily done on grayscale images and cropped images to compare the the images without much difficulty. The Fisher face algorithm or method makes the assumption that training images and the test images are of the equal size which ensures that the input data has the correct shape and has a meaningful exception.

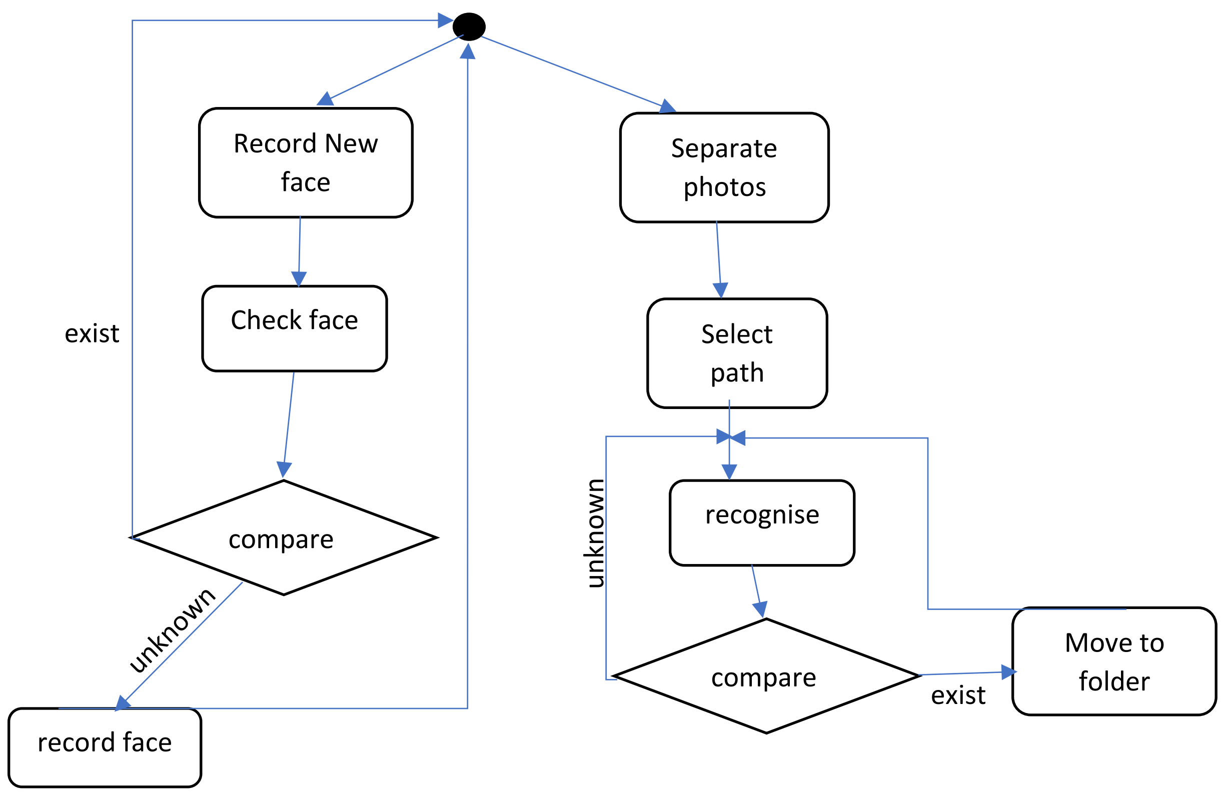


Face Recognition System

• The input of a face recognition system is always an image or video stream.

• The output is an identification or verification of the subject or subjects that appear in the image or video.

DESIGN



COMPARISON WITH OTHER STATES OF ART WORK

Google photos is also similar to this, it will show faces separately and same photo also mixed in all files. but with this project, when some photos are merged in same folder, then if you want to split them without disturbing other folders, you can use this. It will send all persons identified to their respective folders with their names and leaving behind unidentified faces and unknown faces.

CODES:

Check.py

```
import cv2
import numpy as np
from numpy import load
from os import listdir
from os.path import isfile, join
from PIL import Image as img
def face_detector(img, size = 0.5):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return img,[]

    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,0),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200,200))

    return img,roi



def face_extractor(img):
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return None

    for(x,y,w,h) in faces:
        cropped_face = img[y:y+h, x:x+w]

    return cropped_face



Labels =[]
for i in range(0,100):
    Labels.append(i)
Training_Data=load('data.npy')
names=load('names.npy')
l=len(names)



face_classifier =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(Training_Data[0]), np.asarray(Labels))
```

```
cap = cv2.VideoCapture(0)
while True:
    ret,frame=cap.read()
    face=face_extractor(frame)
    if face is not None:
        break
face = cv2.resize(face,(200,200))
face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
c=0
for i in range(0,l):
    model = cv2.face.LBPHFaceRecognizer_create()
    model.train(np.asarray(Training_Data[i]),
np.asarray(Labels))
    name=names[i]
    result = model.predict(face)
    if result[1] < 500:
        confidence = int(100*(1-(result[1])/300))
    if confidence > 82:
        c=1
        break
    del model
if c==1:
    print(name,"face is already recorded!")
```

```
else:
    print('new face please record using dataset')



cap.release()
cv2.destroyAllWindows()
```

dataset.py

```
import cv2
import numpy as np
from numpy import save
from numpy import load

face_classifier =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

def face_extractor(img):

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return None
    for(x,y,w,h) in faces:
        cropped_face = img[y:y+h, x:x+w]

    return cropped_face



cap = cv2.VideoCapture(0)
count = 0
data=[]
while True:
    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count+=1
        face = cv2.resize(face_extractor(frame),(200,200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        data.append(face)
    else:
        print("Face not found")
        pass

    if cv2.waitKey(1)==13 or count==100:
        break
```

```
cap.release()
cv2.destroyAllWindows()
print('Samples Colletion Completed ')
name=str(input('name : '))
n=load('names.npy')
n=n.tolist()
n.append(name)
d=load('data.npy')
d=d.tolist()
d.append(data)
save('data.npy',d)
save('names.npy',n)
```

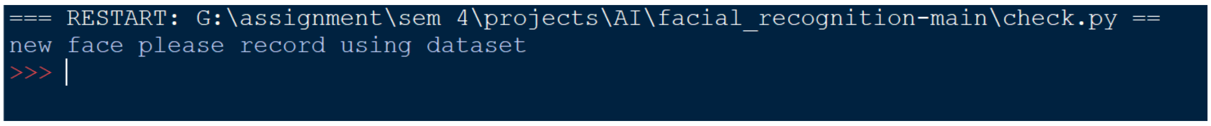
detection.py

```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join
from numpy import load
from PIL import Image as img
import os,shutil

Labels = []
```

```
for i in range(100):
    Labels.append(i)

Labels = np.asarray(Labels, dtype=np.int32)

data=load('data.npy')
names=load('names.npy')

face_classifier =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
for i in range(len(names)):
    if not os.path.exists(names[i]):
        os.mkdir(names[i])
def face_detector(img, size = 0.5):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return img,[]

    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,0),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200,200))

    return img,roi
def check(model,face):
    result=model.predict(face)
    if result[1] < 500:
        confidence = int(100*(1-(result[1])/300))
    if confidence > 82:
        return(1)
    else:
        return(0)
data_path = "G:/assignment/sem 4/projects/AI/New
folder/facial_recognition-main/newfolder/New
folder/facial_recognition-main/test/"
onlyfiles = [f for f in listdir(data_path) if
isfile(join(data_path,f))]
for i in range(0,len(onlyfiles)):
    file=data_path+onlyfiles[i]
    im=img.open(file)
    frame=np.array(im)
    image, face = face_detector(frame)

    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        for j in range(0,len(names)):
            Training_Data=data[j]
            name=names[j]
            model = cv2.face.LBPHFaceRecognizer_create()
            model.train(np.asarray(Training_Data),
np.asarray(Labels))
            result=check(model,face)
            if result==1:
                shutil.move(file,name)
                break
            if(j==len(names)-1):
                print(onlyfiles[i],"  -  unknown")
            del model,j,result,name

    except:
        print(onlyfiles[i],"  -  face not found")
    del file,im,frame,image,face
```

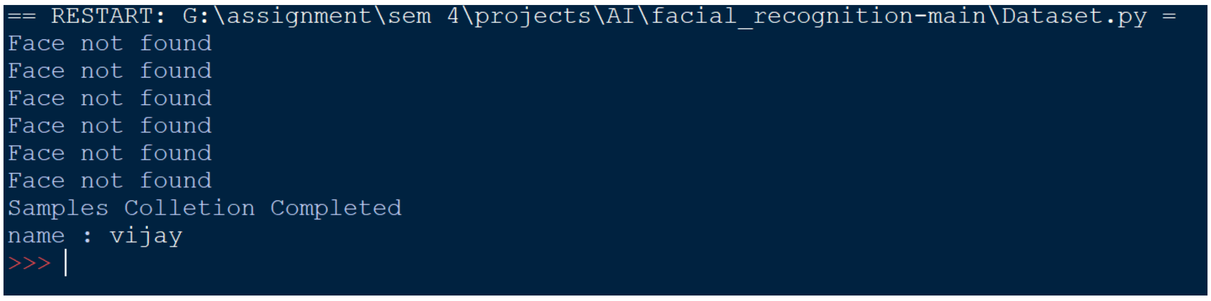
<u>OUTPUT</u>

<u>Using check for unknown face</u>



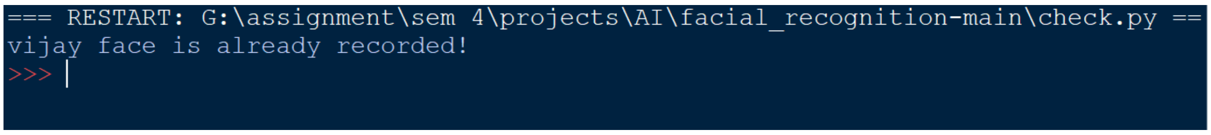
```
=== RESTART: G:\assignment\sem 4\projects\AI\facial_recognition-main\check.py ==
new face please record using dataset
>>> |
```

Use dataset to detect new face



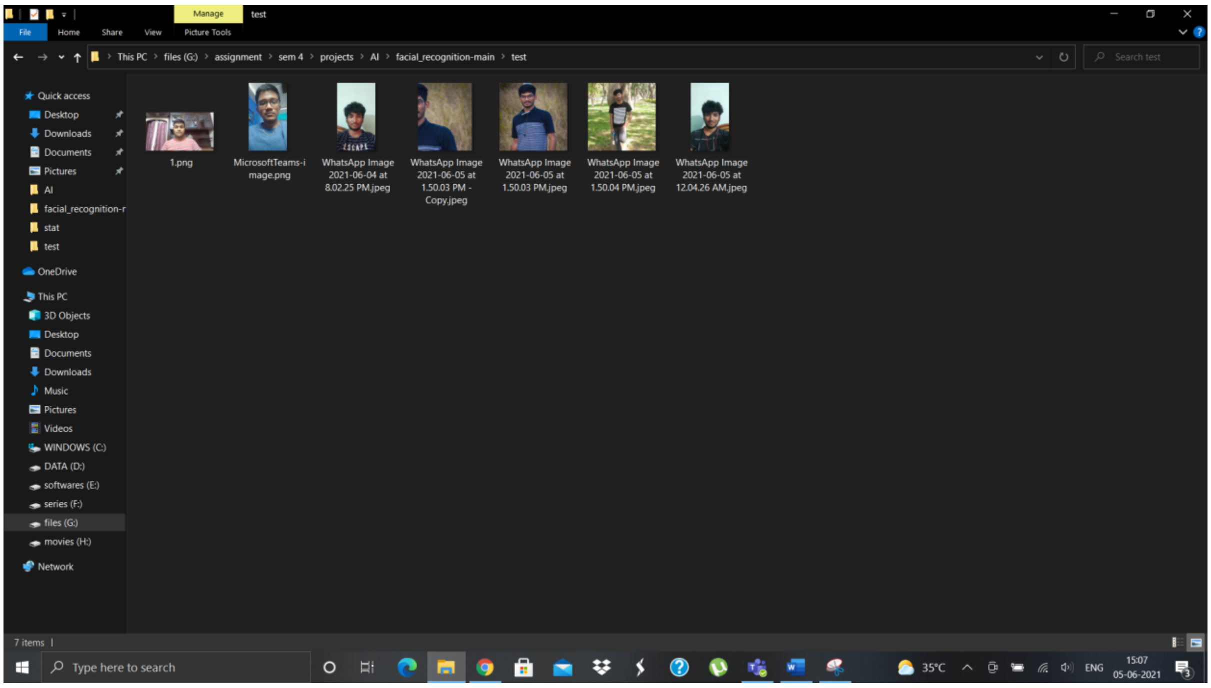
```
== RESTART: G:\assignment\sem 4\projects\AI\facial_recognition-main\Dataset.py =
Face not found
Face not found
Face not found
Face not found
Face not found
Face not found
Samples Colletion Completed
name : vijay
>>> 
```

Use check for known face

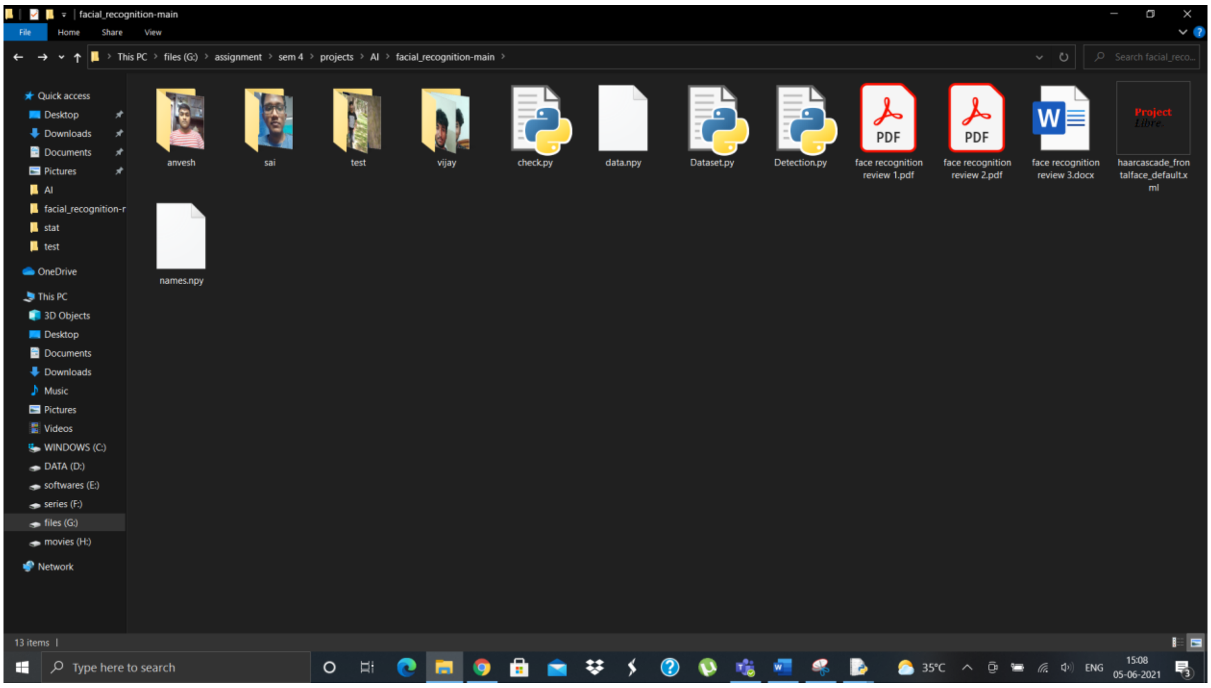


```
=== RESTART: G:\assignment\sem 4\projects\AI\facial_recognition-main\check.py ==
vijay face is already recorded!
>>> 
```

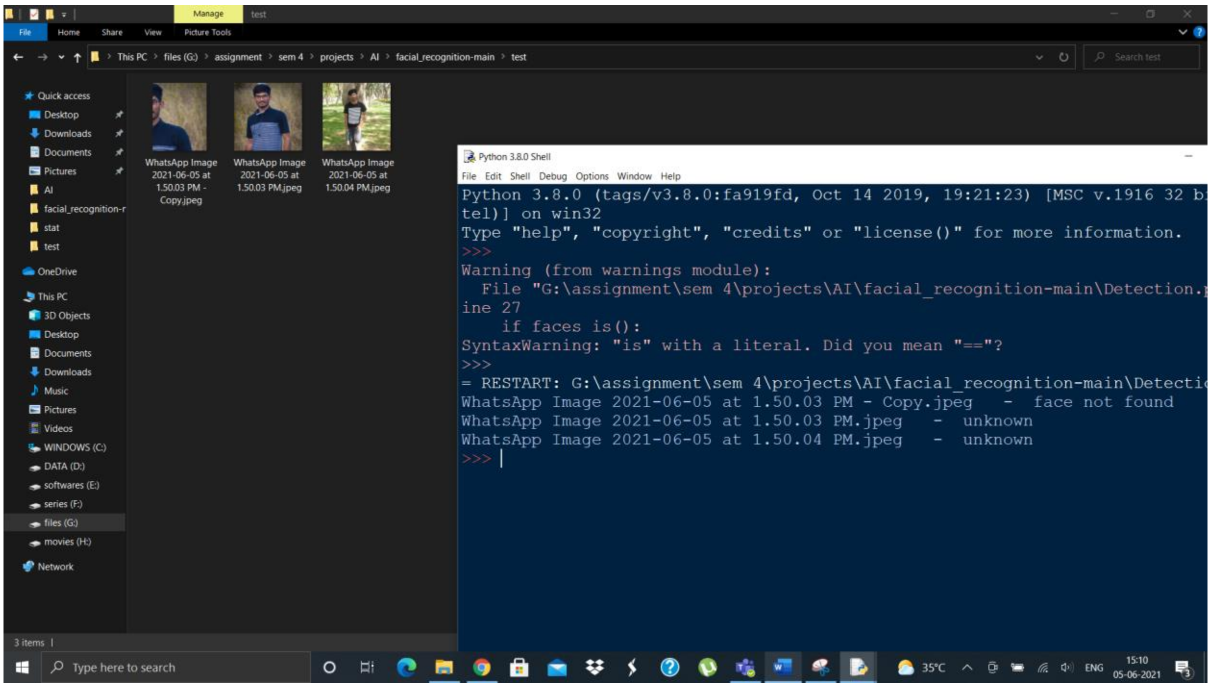
Folder mixed with photos


This PC > files (G:) > assignment > sem 4 > projects > AI > facial_recognition-main > test
1.png
MicrosoftTeams-image.png
WhatsApp Image 2021-06-04 at 8.02.25 PM.jpeg
WhatsApp Image 2021-06-05 at 1.50.03 PM - Copy.jpeg
WhatsApp Image 2021-06-05 at 1.50.03 PM.jpeg
WhatsApp Image 2021-06-05 at 1.50.04 PM.jpeg
WhatsApp Image 2021-06-05 at 12.04.26 AM.jpeg
7 items

After running detection


facial_recognition-main
This PC > files (G:) > assignment > sem 4 > projects > AI > facial_recognition-main
anvesh
sai
test
vijay
check.py
data.npy
Dataset.py
Detection.py
face recognition review 1.pdf
face recognition review 2.pdf
face recognition review 3.docx
haarcascade_frontalface_default.xml
names.npy
13 items

Unknown and no facerecognised images after running detection


This PC > files (G:) > assignment > sem 4 > projects > AI > facial_recognition-main > test
WhatsApp Image 2021-06-05 at 1.50.03 PM - Copy.jpeg
WhatsApp Image 2021-06-05 at 1.50.03 PM.jpeg
WhatsApp Image 2021-06-05 at 1.50.04 PM.jpeg
Python 3.8.0 Shell
3 items

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 b
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Warning (from warnings module):
  File "G:\assignment\sem 4\projects\AI\facial_recognition-main\Detection.
ine 27
    if faces is():
SyntaxWarning: "is" with a literal. Did you mean "=="?
>>>
= RESTART: G:\assignment\sem 4\projects\AI\facial_recognition-main\Detecti
WhatsApp Image 2021-06-05 at 1.50.03 PM - Copy.jpeg   -  face not found
WhatsApp Image 2021-06-05 at 1.50.03 PM.jpeg   -  unknown
WhatsApp Image 2021-06-05 at 1.50.04 PM.jpeg   -  unknown
>>> |
```

<u>CONCLUSION</u>

Separated mixed photos in a folder according to the faces and names scanned before

<u>REFERENCES</u>

[1] A. Eriksson and P. Wretling, How flexible is the human voice? A case study of mimicry, in Proceeding Europian Conference of Speech Technology, Rhodes, pp. 10431046, 1997.

[2] A.K. Jain, A. Ross, S. Prabhakar, “An introduction to biometric recognition,” IEEE Trans. Circuits Syst. Video Technol. 14 (1), pp.420,2004.

[4] A. Pentland, Looking at people: Sensing for ubiquitous and wearable computing, IEEE Transaction Pattern Analysis and Machine Intelligence, vol. 22, pp. 107119, Jan. 2000.

[5] .J. L. Obermayer, W. T. Riley, O. Asif, and J. Jean-Mary, ‘‘College smokingcessation using cell ….. phone text messaging,’’ J. Amer. College Health, vol. 53, no.

[6], pp. 71–78, 2004. 2. S. Haug, C. Meyer, G. Schorr, S. Bauer, and U. John, ‘‘Continuous individual support of smoking cessation using text messaging: A pilot experimental study,’’ Nicotine Tobacco Res., vol. 11, no. 8, pp. 915– 923, 2009.

[7]. D. Scherr, R. Zweiker, A. Kollmann, P. Kastner, G. Schreier, and F. M. Fruhwald, ‘‘Mobile phone-based surveillance of cardiac patients at home,’’ J. Telemedicine Telecare, vol. 12, no. 5, pp. 255–261, 2006.

[8]. D. Aradhana, K. Karibasappa and A. Chennakeshva Reddy, Face recognition using soft computing tools: A survey, UbiCC J. 6(3) (2009) 854863.

[9]. R. Basri and D. Jacobs, Lambertian re°ection and linear subspaces, IEEE Trans. Pattern Anal. Mach. Intell. 25(3) (2003) 218233.