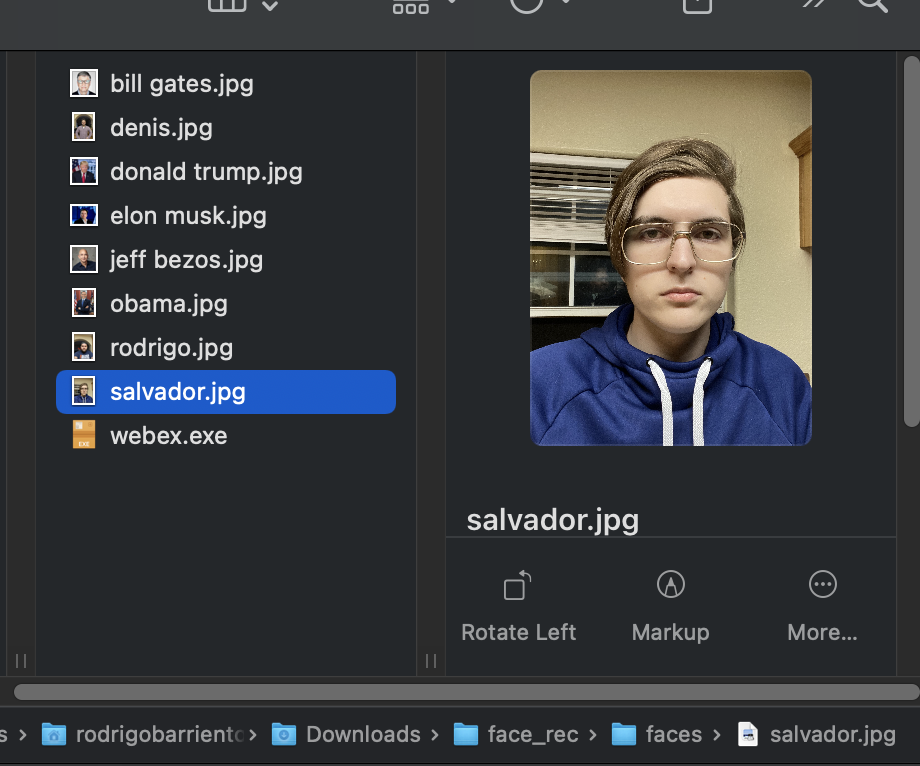
FACIAL RECOGNITION PROJECT

SALVADOR LEON LOPEZ ID: 20432704

Simple project for facial recognition based on a database of known faces. Requirements for the project are cmake, open cv, face\_recognition, numpy and dlib. Also, only png and jpg images are allowed to be compatible with the libraries.

How does it work? Given an input of known faces they get encoded and listed as a directory to serve as our known faces which will be the options to be “recognized” on a different image. Each mugshot images is named with the name of the person whom the face corresponds to.



Upon input of a different image with one or multiple faces on it, it gets encoded and each face is the compared with the directory previously created. A function called best match lets us know which face is the closest to the one on the unknown image and it displays the name and the image now has that face recognized. This process happens for each face on the image.

A picture containing person, indoor, posing

Description automatically generated

Successfully matched known faces into a different image. As the samples are minimal, the program may relate faces when people look alike and incorrectly label people. If there is no best match the face will pop out as “Unknown”.

Code:

import face\_recognition as fr

import os

import cv2

import face\_recognition

import numpy as np

from time import sleep

#Get encoded faces to recognize

*def* get\_encoded\_faces():

encoded = {}

for dirpath, dnames, fnames in os.walk("./faces"):

for picture in fnames:

if picture.endswith(".jpg") or picture.endswith(".png") or picture.endswith(".jpeg"):

face = fr.load\_image\_file("faces/" + picture)

print(picture)

encoding = fr.face\_encodings(face)[0]

encoded[picture.split(".")[0]] = encoding

return encoded

*def* unknown\_image\_encoded(*img*):

#Encode Face

face = fr.load\_image\_file("faces/" + *img*)

encoding = fr.face\_encodings(face)[0]

return encoding

*def* classify\_face(*im*):

#Check if a face in the image is already known and clasify them

faces = get\_encoded\_faces()

faces\_encoded = list(faces.values())

known\_face\_names = list(faces.keys())

img = cv2.imread(*im*, 1)

face\_locations = face\_recognition.face\_locations(img)

unknown\_face\_encodings = face\_recognition.face\_encodings(img, face\_locations)

face\_names = []

for face\_encoding in unknown\_face\_encodings:

matches = face\_recognition.compare\_faces(faces\_encoded, face\_encoding)

name = "Unknown"

face\_distances = face\_recognition.face\_distance(faces\_encoded, 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)

for (top, right, bottom, left), name in zip(face\_locations, face\_names):

cv2.rectangle(img, (left-20, top-20), (right+20, bottom+20), (255, 0, 0), 2)

cv2.rectangle(img, (left-20, bottom -15), (right+20, bottom+20), (255, 0, 0), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(img, name, (left -20, bottom + 15), font, 1.0, (255, 255, 255), 2)

# Display the resulting image

while True:

cv2.imshow('Video', img)

if cv2.waitKey(1) & *0x*FF == ord('q'):

return face\_names

print(classify\_face("test.jpg"))

print(classify\_face("test2.jpg"))

print(classify\_face("test3.jpg"))