|  |
| --- |
| import cv2 |
|  | import numpy as np |
|  | import face\_recognition |
|  | import os |
|  | from datetime import datetime |
|  |  |
|  | # from PIL import ImageGrab |
|  |  |
|  | path = 'Training\_images' |
|  | images = [] |
|  | classNames = [] |
|  | myList = os.listdir(path) |
|  | print(myList) |
|  | for cl in myList: |
|  | curImg = cv2.imread(f'{path}/{cl}') |
|  | images.append(curImg) |
|  | classNames.append(os.path.splitext(cl)[0]) |
|  | print(classNames) |
|  |  |
|  |  |
|  | def findEncodings(images): |
|  | encodeList = [] |
|  |  |
|  |  |
|  | for img in images: |
|  | img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) |
|  | encode = face\_recognition.face\_encodings(img)[0] |
|  | encodeList.append(encode) |
|  | return encodeList |
|  |  |
|  |  |
|  | def markAttendance(name): |
|  | with open('Attendance.csv', 'r+') as f: |
|  | myDataList = f.readlines() |
|  |  |
|  |  |
|  | nameList = [] |
|  | for line in myDataList: |
|  | entry = line.split(',') |
|  | nameList.append(entry[0]) |
|  | if name not in nameList: |
|  | now = datetime.now() |
|  | dtString = now.strftime('%H:%M:%S') |
|  | f.writelines(f'\n{name},{dtString}') |
|  |  |
|  | #### FOR CAPTURING SCREEN RATHER THAN WEBCAM |
|  | # def captureScreen(bbox=(300,300,690+300,530+300)): |
|  | # capScr = np.array(ImageGrab.grab(bbox)) |
|  | # capScr = cv2.cvtColor(capScr, cv2.COLOR\_RGB2BGR) |
|  | # return capScr |
|  |  |
|  | encodeListKnown = findEncodings(images) |
|  | print('Encoding Complete') |
|  |  |
|  | cap = cv2.VideoCapture(0) |
|  |  |
|  | while True: |
|  | success, img = cap.read() |
|  | # img = captureScreen() |
|  | imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25) |
|  | imgS = cv2.cvtColor(imgS, cv2.COLOR\_BGR2RGB) |
|  |  |
|  | facesCurFrame = face\_recognition.face\_locations(imgS) |
|  | encodesCurFrame = face\_recognition.face\_encodings(imgS, facesCurFrame) |
|  |  |
|  | for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame): |
|  | matches = face\_recognition.compare\_faces(encodeListKnown, encodeFace) |
|  | faceDis = face\_recognition.face\_distance(encodeListKnown, encodeFace) |
|  | # print(faceDis) |
|  | matchIndex = np.argmin(faceDis) |
|  |  |
|  | if matches[matchIndex]: |
|  | name = classNames[matchIndex].upper() |
|  | # print(name) |
|  | y1, x2, y2, x1 = faceLoc |
|  | y1, x2, y2, x1 = y1 \* 4, x2 \* 4, y2 \* 4, x1 \* 4 |
|  | cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2) |
|  | cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED) |
|  | cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 255, 255), 2) |
|  | markAttendance(name) |
|  |  |
|  | cv2.imshow('Webcam', img) |
|  | cv2.waitKey(1) |