**House Safety using Face Recognition by Opencv and Numpy**

While today’s burglary statistics show an overall decrease in burglary rates, thousands of homes (roughly 325,000) are still being broken into every year. There are roughly 2.5 million burglaries a year, 66% of those being home break ins.

Police solve only 13% of reported burglary cases due to lack of a witness or physical evidence. Thus, the objective of this project is to prevent break ins like these, and ensure a way to secure our home from miseries like these, we initiate a way to have record of all the people coming to our house.

This project is made as contribution to the security sector, with the burglary rate still being in thousands and to keep ourselves and our families safe, we can develop this project using “face recognition” with opencv and numpy. So, we consider making a project to deal with the security of people entering and leaving the house and alerting people in the house for any unknown break-ins.

**What is Face Recognition?**

Facial recognition system is a system, made to match human face from a set of data of images, to serving many purposes through ID verification services, works by proportionally measuring the dimensions of a face from an image. Applications of face recognition includes Security system, Aid forensic investigations, Diagnose diseases, Track school attendance, Secure transactions and etc.

The process for face recognition is as follows:

1. Face detection
2. Face Analysis
3. Facial feature extraction
4. Face Recognition

**Building Dataset (by capturing face dimensions):**

The dataset for this project does not involve any real-time highly used data records, instead it’s our and our roommate’s faces that is to be recorded as the dataset. We make this possible by a series of steps involving reading the dimensions of face through haar-cascades, converting to grayscale, capturing the images of face, and then building the dataset.

The dataset can include you, your family, roommates and the other people who leave and enter your house. So, when the images are taken, the dataset will look something like set of images captured in grey color.

**Modules needed:**

**• Opencv:** This library is a major pillar when working with Face recognition projects, as it is a aimed on focussing real-time applications and highly used python library, though developed in C/C++.

**• Numpy:** Numpy is more of mathematically used in for arrays and matrices with many functionalities helping in building the dimensions of the captured face.

**• OS:** A standard module, used in most of the python projects made to interacting with the operating system of the computer.

**• OS Path:** Just as the os module, this is mend to providing path parameters in handling large files or documents.

• **Playsound:** As the name suggests, this module is made for playing simple audio files.

**Steps to installing the packages:**

Open Search bar and start typing ‘Command prompt’, make sure you have the latest pip version installed in your computer. Now, start installing packages simultaneously by the following convention:

Pip install opencv-python

Pip install numpy

Pip install playsound

**Steps involved in coding the modules (short summary):**

1. Building the dataset with user’s face, starting by importing the required above mentioned packages.

2. We use haar-cascades in classifying the components of the face.

3. Now, we convert the face to gray scale, because it involves less complications and each pixel can be significant enough for the computer to read.

4. Marking the co-ordinates of the face features to be captured and read into a separate file (make sure it’s in the same folder as the other file).

5. Once, the dimensions of face and camera monitoring is done, voila your dataset is built!

6. Now, we train the module to detect face, by running through the dataset built.

7. But for the module to detect the face it should be done in grayscale, as in the above case.

8. Lastly, make sure the face matches with dataset by mentioning 'the user' and adding a “alert sound”, for an 'unknown' person.

**Let's start building dataset-**

**Step-1:** import the packages required.

Python

#imporing the modules

import cv2

import numpy as np

**Step-2:** Use haar-cascades in detecting the image, you can find 'haarcascade\_frontalface\_default.xml' file on your computer or even download the file from the internet.

Python

#converting the captured images to grayscale

face\_classifier=cv2.CascadeClassifier(cv2.data.haarcascades + '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\_faces=img[y:y+h,x:x+w]

return cropped\_faces

cap=cv2.VideoCapture(0)

count=0

**Step-3:** Now add a new file in the same folder to store your data where all the images will be stored in .jpg file, through specifying the co-ordinates of the face to be taken, as mentioned below.

Python

#preparing a new file for dataset and collecting samples of the face

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)

file\_name\_path= '\*\*Add a path where you want to store the dataset\*\*'+str(count)+'.jpg'

cv2.imwrite(file\_name\_path,face)

cv2.putText(face,str(count),(50,50),cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 0, 0), 1)

cv2.imshow('Face Cropper',face)

else:

print("face not found")

pass

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

break

cap.release()

cv2.destroyAllWindows()

print('Collecting samples completed!')

**Output:** The file of the dataset taken would look something like this,

Dataset of the model

**Training the model:**

**Step-5:** importing the modules for the second part of the project.

Python

#importing the required modules

import cv2

import numpy as np

from os import listdir

from os.path import isfile, join

from playsound import playsound

**Step-6:** Add the path to the before made dataset and convert it to grayscale.

Python

#Refering the path to the Dataset

data\_path='\*\*path file of the dataset\*\*'

onlyfiles=[f for f in listdir(data\_path) if isfile(join(data\_path,f))]

Training\_Data,Labels=[],[]

for i, files in enumerate(onlyfiles):

image\_path=data\_path+onlyfiles[i]

images=cv2.imread(image\_path,cv2.IMREAD\_GRAYSCALE)

Training\_Data.append(np.asarray(images,dtype=np.uint8))

Labels.append(i)

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

model=cv2.face.LBPHFaceRecognizer\_create()

model.train(np.asarray(Training\_Data),np.asarray(Labels))

print('model training complete!!')

**Step-7:** Using haar-cascades you help detecting the face and then appoint the below dimensions to read the facial features.

Python

def face\_detector(img,size=0):

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),(0,255,255),2)

roi=img[y:y+h,x:x+w]

roi=cv2.resize(roi,(200,200))

return img,roi

cap=cv2.VideoCapture(0)

**Step-8:** Finally, you train the module to match it with the dataset of images of your face and download a alerting 'beep sound' for the cause of a unknown.

Python

while True:

ret, frame=cap.read()

image, face=face\_detector(frame)

try:

face=cv2.cvtColor(face,cv2.COLOR\_BGR2GRAY)

result=model.predict(face)

if result[1]<500:

confidence=int(100\*(1-(result[1])/300))

display\_string=str(confidence)+'% confidence it is user'

cv2.putText(image, display\_string,(100,120),cv2.FONT\_HERSHEY\_COMPLEX,1,(250,120,255),2)

if confidence>75:

cv2.putText(image,"the user",(250,450),cv2.FONT\_HERSHEY\_COMPLEX,1,(0,255,0),2)

cv2.imshow("Face Cropper",image)

else:

cv2.putText(image,"unknown",(250,450),cv2.FONT\_HERSHEY\_COMPLEX,1,(0,0,255),2)

cv2.imshow("Face Cropper",image)

playsound('\*\*Add the path to your audio file\*\*')

except:

cv2.putText(image,"Face not found",(250,450),cv2.FONT\_HERSHEY\_COMPLEX,1,(255,0,0),2)

cv2.imshow("Face Cropper",image)

pass

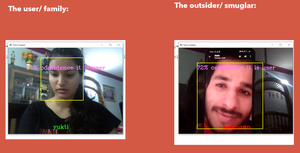
if cv2.waitKey(1)==13:

break

cap.release()

cv2.destroyAllWindows()

**Output:** Now, when you run the code it would start to scan your face, and likewise tell if you are 'the user' or alert you with a sound incase it is a stranger. The implementation should look something like this,



Final output of the model

**Future more developments to implement:**

• This is just the software part of the project, we further want to enable this as a full-fledged project with the working of the Arduino, raspberry pie (and other hardware) as including the hardware part of the project would require to some resources.

• We can build interface, like a web application or an android application to further making it accessible to everyone (with less expense) and having a record/history.

• Further changes like adding more dataset of people with criminal records from legal authorities can be merrier.

• Data can be updated with respect to number of people coming and going, from time to time due to the simplicity of the project.

**References:**

Following references are just for inspiration and nothing of them is directly copied or taken from.

<https://en.wikipedia.org/wiki/Facial_recognition_system>

<https://ieeexplore.ieee.org/document/8697695/figures#figures>

<https://www.ooma.com/home-security/facial-recognition-part-of-home-security/>

<https://ieeexplore.ieee.org/document/8982466>

<https://link.springer.com/chapter/10.1007/978-981-15-5232-8_7>

\*\*\*