Implementation of Safety Survilliance System with Face Recognition using IoT

Kakarla Deepti*
*Department of ECE, Vasavi College of Engineering,
Osmania University, Hyderabad – 500 031, Telangana, India;

Abstract-: **In recent times, the importance of automated self-intelligent systems has been proven to be very effective in applications like private security and surveillance. This research article aims to provide safety surveillance with the aid of facial recognition technology and the Internet of Things. The hardware and software requirements include an open source technology which comprises of LBPH algorithm, SMTP, Raspberry pi 3, pi camera. The utilization applications are intended to be at local levels like homes, offices and campuses. The proposed system provides real time face detection and recognition. The captured image is analyzed using LBPH algorithm with the available database and if it is a match, the individual is granted access. On the other hand, if the face does not match anything in the database, the captured image is then sent to the user mail using SMTP. A SMS is also sent to the registered number to make sure the house owner gets the alert. The system then waits for the response from the user for a stipulated time. The message is retrieved on Raspberry pi using IMAP. Based on the retrieved message, access is either granted or denied.**

Keywords: Face detection, Face recognition, Face authentication, Raspberry Pi, IoT

INTRODUCTION

Nowadays, lots of individuals keep moving around the world due to their jobs and various other reasons. Therefore, monitoring what's happening at our homes and offices is now an essential. Today, there is a requirement for efficient design and development of authentication technology in light of the ever increasing terrorism and criminal activities. Aauthentication using biometrics is one of the best automated identification or verification of individual identity. Physiological and behavioural characteristics such as face, fingerprints and voice can be used for this purpose[1]. This form of authentication is used widely used for security of computer networks, international border safeguarding, restricted and controlled access to facilities, etc.

The advantage of face authentication is that the images can be captured even from a distance, which is not the case with other biometrics. This does not require special actions as can the captured images can be recorded and the identity of any person can be easily determined which can be used for crime reduction[2]. This makes it highly effective for extensive usage in both security related applications like video surveillance and also for the purpose of image retrieval, indexing and user interface.

Previously, researchers have considered the head and shoulders of the object to detect its movement more appropriately since they are the most unvarying part of the human body[3]. The range of monitoring in this study is up to five meters; hence it is difficult to detect a face at a distance much further than that since the face will be small and blurry[4]. This implementation uses a Raspberry Pi Model to connect the web camera to capture the footage and to send an email when movement is detected.

As shown in Fig.1 the movement is first detected using a passive infrared sensor. The camera switches ON, and a picture of the person is captured. The face of the person is detected in the image and processed. This face is then compared with the existing faces in the database[5]. If a reliable match is found within the database, access or authentication is granted to the individual automatically. On the other hand, if a match is not found, the image is saved and sent to the users through the registered email id. A secondary text alert is also sent to the owner's mobile. The owner can then review the image and decide to grant or deny access to the individual.

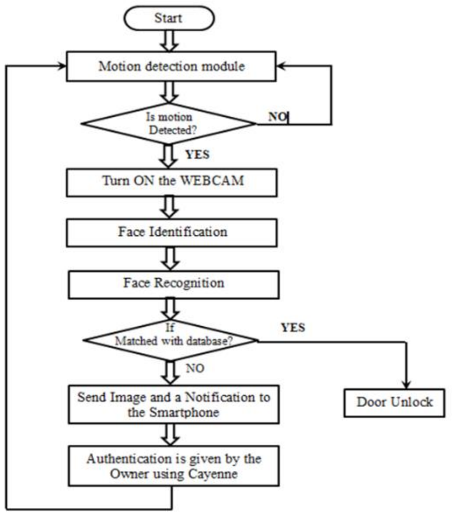

Start
Motion detection module
Is motion Detected?
NO
YES
Turn ON the WEBCAM
Face Identification
Face Recognition
If Matched with database?
YES
NO
Door Unlock
Send Image and a Notification to the Smartphone
Authentication is given by the Owner using Cayenne

Fig.1. Flow diagram of implementation process

I. SYSTEM ARCHITECTURE

Developing a Face recognition system requires incorporation of software and hardware components. This section discusses the Implementation stack required to build the system as shown in the Fig.2. The implementation involves hardware as well as a software component. Haar Cascades and LBPH are respectively used for face detection and face recognition. The Wi-Fi is used for wireless communication between raspberry pi and user. The implementation includes to detect face from a real-time video. In this approach, the Haar cascade classifier is applied, and the database maintained with 100 face images of each user. Training the model with the available user's database. In this approach, the LBPH algorithm is used, which is available in the Open CV. Once the training gets completed, the trainer.yml extension file gets generated, and this is utilized in face recognition. The "Local Binary Pattern Histogram" algorithm is used for recognizing face and training purposes. The person whoever comes in front of the real-time webcam must be detected and matched with the existing trained images in order to maintain security. If the match is found, then the owner remotely provides permission. On the contrary, if the face does not match anything in the database, the taken picture is then sent to the user mail using SMTP protocol. A mobile SMS is also sent to the registered number to make sure the owner gets the alert. Now the user is in a position to authorize the visitor whether to lock or unlock.

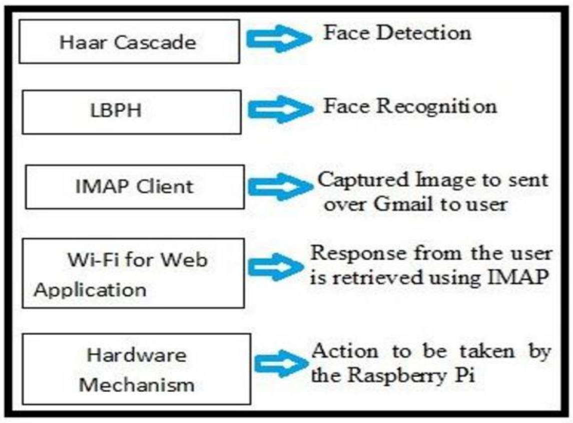
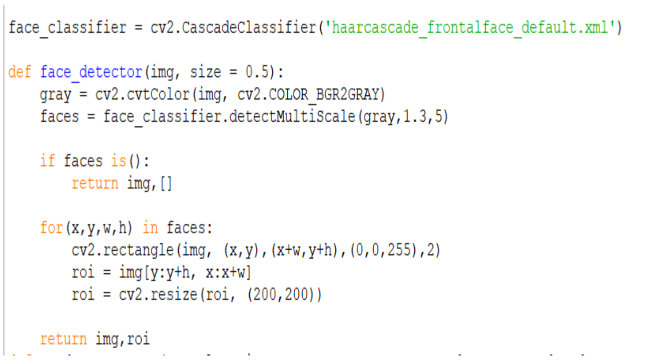

Haar Cascade
Face Detection
LBPH
Face Recognition
IMAP Client
Captured Image to sent over Gmail to user
Wi-Fi for Web Application
Response from the user is retrieved using IMAP
Hardware Mechanism
Action to be taken by the Raspberry Pi

Fig.2. Implementation Stack

A. *Face detection*

The initial process in recognizing faces is face detection. The Machine Learning algorithm "Haar Cascade Classifier" is used for detecting the face which is pre-trained with a lot of positive and negative images. For implementation Haar Cascade classifier is used to detect a face from a real time image which was captured using the Raspberry Pi camera. Haar cascade classifier extorts Haar features to sense a face from the taken real time image. Once the face is detected, pixel positions are noted. These pixel positions are used to resize the image so that it has only the face part other unwanted information from the real time image is discarded. This resized image is then stored into the database with the person's name as label. A 45 face images from each person to create the database for training the algorithm. The camera detects the face of the person and will capture the image of targeted the face alone. The code snippet for face detection is shown in Fig.3



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

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

    return img,roi
```

Fig.3.Code Snippet used for face detection

In the above code snippet, the captured face is resized to a resolution of 200 x 200. This is done to so that the faces can be batched together with the images in the database.Once the training of the face images is complete, it will generate a file with .yml extension, and if we have this trained file, then the model doesn't need to be trained each time, simply the file is used in the recognition code. The yml stands for "YAML Markup Language" that stores the face pixel values.

The LBPH algorithm is available within the OpenCV, and it is interfaced in the program using the following command.

```
path = 'dataset'
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
```

The dataset is provided as the input path and the LBPH recognizer is loaded along with the haar cascade detector. The array is created for faces and ids and using the "pillow" library they are manipulated and saved. The commands for creating the array and manipulation of images are shown below

```
facesamples = [ ]
 ids = [ ]
 for imagePath in imagePaths:
    PIL_img = Image.open(Path).convert('L')
    img_numpy = np.array(PIL_img,'uint8')
```

The Image.open( ) function is available in the "pillow" library and used to link the image path provided, i.e., dataset. The np.array( ) function is used to extract the pixel values for the image opened using Image.open( ) function and later stored in the yml file in the form of arrays. After the extraction of pixel values using np.array the respective ids are appended to those pixel values. So the id from the label is extracted using the below command.

```
id  = int (os.path.split (imagePath) [-1]. split(".") [1])
```

The split operation is performed on the label name of the face. For instance, the label name "User.1.58.jpg" assigned to id = 1 and count as 58. The jpg extension is initially split from the label, and then the count gets split.

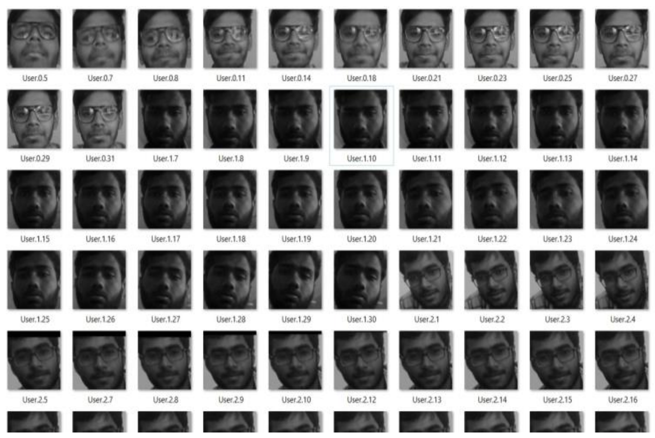
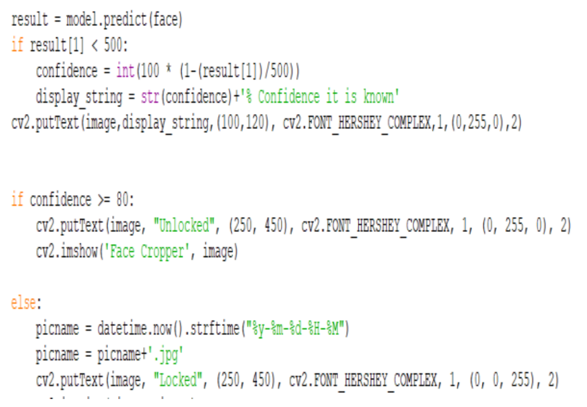


Fig.4.Sample database collected to execute the algorithm

B. *Face Recognition*

The Local binary pattern histogram (LBPH) is used for both training and recognition purposes. The database which was created using Haar cascade classifier is used here for the training purpose. LBPH divides the image into small windows. It computes new value for the central pixel of the window using local threshold value as the central pixel value. After computing Local Binary Pattern, a new image with improved characteristics over the previous is achieved.

Now this LBP image is divided into multiple grids. One single histogram called the Local Binary Pattern Histogram of the image is formed by computing every grid and putting them together. For face recognition, the LBPH of the real time image is computed and then matched with the LBPH of the stored images in the database. The figure shows the camera detecting the face of a person and recognizing it against the stored database and displaying the correct label and confidence level.



```
result = model.predict(face)
if result[1] < 500:
    confidence = int(100 * (1-(result[1])/500))
    display_string = str(confidence)+'% Confidence it is known'
cv2.putText(image,display_string,(100,120), cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)


if confidence >= 80:
    cv2.putText(image, "Unlocked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
    cv2.imshow('Face Cropper', image)

else:
    picname = datetime.now().strftime("%y-%m-%d-%H-%M")
    picname = picname+'.jpg'
    cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
```

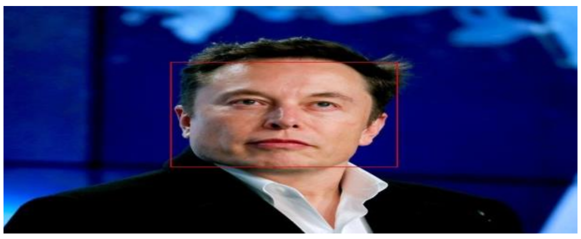


Fig.5.Code Snippet used for face Recognition

The LBPH algorithm is available within the OpenCV, and it is interfaced in the program using the following command.

```
recognizer = cv2.face.LBPHFaceRecognizer_create( )
recognizer.read('trainer/trainer.yml')
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
```

The trainer.yml file which is generated in previous section is interfaced in recognition program using read ( ) function. The recognizer also created from LBPHFaceRecognizer using create ( ) function.

ii) Face detection is performed first, which is described in section 4.3, and then the recognition is performed. The ids of those detected images are predicted using recognizer.predict( ) function, and it is shown below

```
for(x,y,w,h) in faces:
    cv2.rectangle (img, (x,y), (x+w,y+h), (0,255,0), 2)
    id, confidence = recognizer.predict (img)
```

The recognizer. Predict( ) function returns two parameters when the face is detected, i.e., the respective id and the match confidence level. Ideally, if the confidence value is '0', then it is considered to be a perfect match, but it is not possible in real-time surroundings due to changes in the environment. The confidence value is predicted using the histogram comparison, i.e., the histogram of the input image is computed and then compared with the histogram of the database images. The minimum histogram distance is calculated using the Euclidean distance formula, and then it is assigned as the confidence value. The threshold is set to a particular value, i.e., says 50, and then if the confidence value is below the threshold value, then the match occurs. If the confidence value is greater than the threshold, then the id is taken as unknown, and the image has sent to the owner as shown. The array is created with the label "names," and if the match is found, it will return the name associated with the particular id.

```
id, confidence = recognizer.predict(img)
if (confidence < 50):
  id = Names[id]
else:
  id = "unknown"
```

II. RESULTS

The results obtained after the implementation are presented in this section.

Fig.6. Single face detection

In Fig. 6 shows the face detection performed on an image which consists of a single face and the bounding rectangle box is drawn to indicate the detected face region. The Haar Cascade Classifier has detected the single face accurately. This is the initial test performed for face detection and later on the multiple face detection tests are also performed and the result is shown in Fig.7.

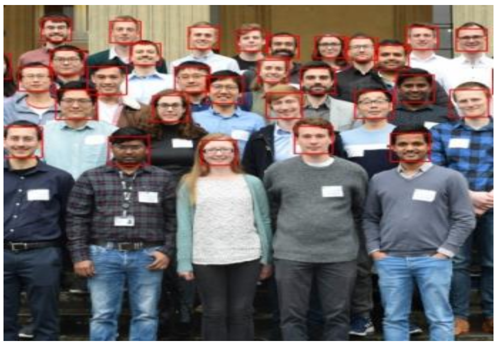


Fig.7. Result of multiple face detection

The detect Multi Scale function parameters are varied and the tests have been performed on multiple faces and the parameters which show the accurate multiple face detection are noted for further implementation of face recognition.

From the data base created as shown in Fig.4. the test is performed to identify the person from data base and the result obtained is shown in Fig.8.

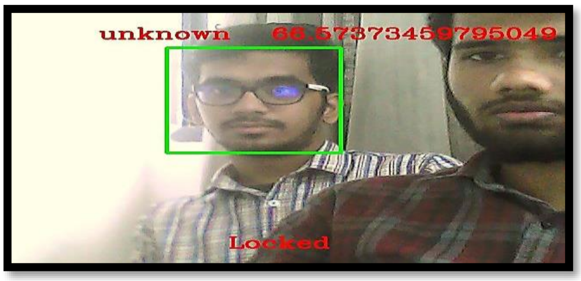
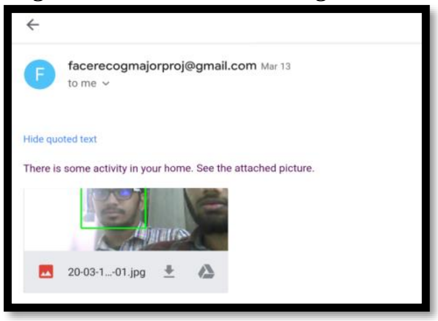


Fig.8. Result indicating an Unrecognized Face and Recognized face

The face goes unrecognized because the person is not in the database. The threshold is set to 50 and the confidence value i.e., 66.5 is greater than the threshold. The access is not granted to a person and further immediately the mail is sent with this image attached as shown in Fig.9.


facerecogmajorproj@gmail.com Mar 13
to me

Hide quoted text

There is some activity in your home. See the attached picture.

20-03-1...-01.jpg

Fig.9. Email sent to the owner

The email is sent through SMTP protocol because the face goes unrecognized. The quoted text "There is some activity in your home. See the attached picture." is also sent along with the image. The image is labeled with the current date and time using the timestamp library. The owner can refer the image label to know the time of image captured. There is no guarantee that the internet is turned 'ON' all the time and if the internet is off then there will be delay in noticing the intruder. To overcome this, the secondary SMS alert is also sent to the owner mobile at the time of sending email as shown in Fig.10.

Fig.10. SMS Sent to the owner

Conclusion

The results validate that the system detects movement and provides real time face detection and recognition. The captured image is analyzed using the LBPH algorithm and matched with pictures available in the database, and if a match is found, the individual is granted access. On the other hand, if the face does not match with any face in the database, the captured image is then sent to the user mail using SMTP. An SMS is also sent to the registered number to make sure the house owner checks his mail. This security feature is also used to inform the home owner if someone tries to tamper with the security system by turning off the power. The system can retrieve information from the home owner and then grant or deny access to a person.

References

[1] Jia-Jing Lin, and Shih-Chang Huang , "The Implementation of the Visitor Access Control System for the Senior Citizen Based on the LBP Face Recognition," 2017 International Conference on Fuzzy Theory and Its Applications (iFUZZY), IEEE, March 2018.

[2] Sushma Jaiswal, Dr. (Smt.) Sarita Singh Bhadauria, Dr. Rakesh Singh Jadon," Comparison Between Face Recognition Algorithm-Eigenfaces, Fisherfaces And Elastic Bunch Graph Matching", Volume 2, No. 7, Journal of Global Research in Computer Science July 2016.

[3] Fellous, J., Krueuger, L., Wiskott, and von der Malsburg,C. Face recognition by elastic bunch graph matching. IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(7):775–779, 1997.

[4] Belhumeur, V., Hespanda, J., Kiregeman, D.," Eigenfaces vs. fisherfaces: recognition using class specific linear pojection", IEEE Trans. on PAMI, V. 19, 1997.

[5] Gupta, I., Patil, V., Kadam, C., & Dumbre, S. (2016). Face detection and recognition using Raspberry Pi, 2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECONECE), Pune, pp. 83-86.