IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 8, Issue 10, October 2019

Face Recognition Using Raspberry Pi

Jagadeesh Subramanian

Anna University, Guindy, Chennai, Tamil Nadu, India

Abstract: Since lot of forgeries occur in our day to day life, security and surveillance has become a mandatory aspect. Over the last few decades, face recognition has become a popular area of research in computer vision. This system may be implemented in real time systems requiring user authentication such as attendance systems, ATM security, Electronic Passports and Network security. Our project aims in developing precise way of detecting faces using an algorithm called ”Eigen face detection” which uses Principal Component Analysis(PCA). The algorithm involves comparison of trained samples with the query images. If the trained samples matches with the query images, then the detected face is a “known face”. If there is a mismatch, then the detected face is a “new face”. PCA extracts the eigen values and eigen vectors for the given set of samples in the PCA-sub space and compares it with the new values during run time. Then the threshold value is calculated from these eigen faces. If the new face value is less than the threshold, then it is a “known face”. Otherwise, he/she must be a new user. This algorithm is implemented using python on a Raspberry pi module in which the raspberry camera is used to extract the face values. The Face values are displayed on a 659M10 LCD so that it could act as an alarm in door security system.

Keywords: Arduino, Touchless Interface, Micro Controller, Capacitive Sensing

I. INTRODUCTION

Biometrics is used in the process of authentication of a person by verifying or identifying that a user requesting a network resource is who he, she, or it claims to be, and vice versa. It uses the property that a human trait associated with a person itself like structure of finger, face details etc. By comparing the existing data with the incoming data we can verify the identity of a particular person, There are many types of biometric system like fingerprint recognition, face detection and recognition, iris recognition etc., these traits are used for human identification in surveillance system, criminal identification. Advantages of using these traits for identification are that they cannot be forgotten or lost. These are unique features of a human being which is being used widely. Generally, there are three phases for face recognition, mainly face representation, face detection, and face identification.

Face representation is the first task, that is, how to model a face. The way to represent a face determines the successive algorithms of detection and identification. For the entry-level recognition (that is, to determine whether or not the given image represents a face), a face category should be characterized by generic properties of all faces; and for the subordinate-level recognition (in other words, which face class the new face belongs to), detailed features of eyes, nose, and mouth have to be assigned to each individual face. There are a variety of approaches for face representation, which can be roughly classified into three categories: template-based, feature-based, and appearance-based.

The simplest *template-matching* approaches represent a whole face using a single template, i.e., a 2-D array of intensity, which is usually an edge map of the original face image. In a more complex way of template-matching, multiple templates may be used for each face to account for recognition from different viewpoints. Another important variation is to employ a set of smaller facial feature templates that correspond to eyes, nose, and mouth, for a single viewpoint. The most attractive advantage of template-matching is the simplicity, however, it suffers from large memory requirement and inefficient matching. In *feature-based* approaches, geometric features, such as position and width of eyes, nose, and mouth, eyebrow's thickness and arches, face breadth, or invariant moments, are extracted to represent a face. Feature-based approaches have smaller memory requirement and a higher recognition speed than template-based ones do. They are particularly useful for face scale normalization and 3D head model-based pose estimation. However, perfect extraction of features is shown to be difficult in implementation [5]. The idea of *appearance-based* approaches is to project face images onto a linear subspace of low dimensions. Such a subspace is first constructed by principal component analysis on a set of training images, with eigenfaces as its eigenvectors. Later, the concept of eigenfaces were extended to eigenfeatures, such as eigeneyes, eigenmouth, etc. for the detection of facial features. More recently, fisherface space and illumination subspace have been proposed for dealing with recognition The touch screens itself are of two types- Capacitive and Resistive. While capacitive is the more preferred one due to its smooth and flowing interface but a resistive one needs to be there for any kind of stylus interaction. It's hard to believe that just a few decades ago; touch screen technology could only be found in science fiction books and film. These days, it's almost unfathomable how we once got through our daily tasks without a trusty tablet or smart phone nearby, but it doesn't stop

DOI 10.17148/IJARCCE.2019.81012

there. Touch screens really are everywhere. Homes, cars, restaurants, stores, planes, wherever—they fill our lives in spaces public and private. The main feature which motivated me for making this project was to create something useful by using simple materials and circuitry. I'm well aware there are better ways to do capacitive distance sensing, but I wanted to make something as simple as possible that's still functional under varying illumination.

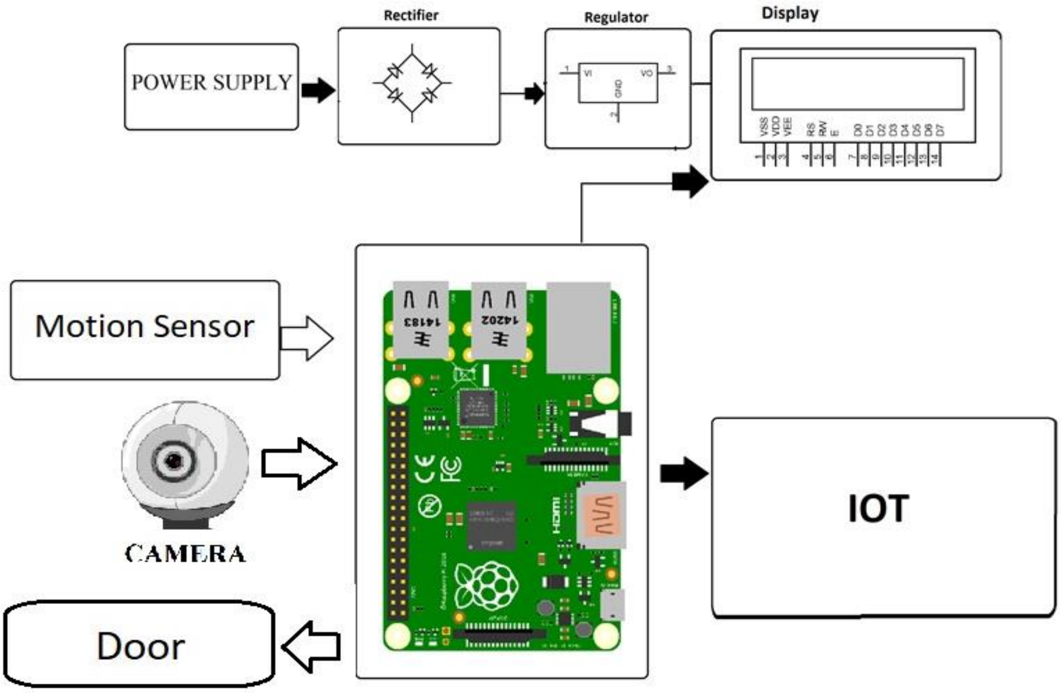


Fig 1.1 Face Recognition Model Block Diagram

II. PROJECT ANALYSIS

2.1 Feature Extraction:

In face alignment, facial components, such as eyes, nose, and mouth, and facial outline are located, and thereby the input face image is normalized in geometry and photometry. The following are the methods by which the features of the particular face can be extracted and can be used to create a data model from the given set of data trials.

-Principal Component Analysis (PCA)
-Linear Discriminant Analysis (LDA)
-Independent Component Analysis (ICA)

2.2 Linear Subspace Analysis

Three classical linear appearance-based models: PCA, LDA and ICA

•Each model has its own representation (basis vectors) of a high-dimensional face vector space based on different statistical viewpoints.

•All the three representations can be considered as a linear transformation from the original image vector to a projection feature vector.

$$Y=W^TX$$

Where Y – (d x 1), X – (n x 1) and W – (n x d), d << n

2.2.1 Principal Component Analysis:

Principal component analysis (PCA) was invented in 1901 by Karl Pearson. PCA is a variable reduction procedure and useful when obtained data have some redundancy. This will result into reduction of variables into smaller number of variables which are called Principal Components which will account for most of the variance in the observed variable. Problems arise when we wish to perform recognition in a high-dimensional space. Goal of PCA is to reduce the dimensionality of the data by retaining as much as variation possible in our original data set. On the other hand, dimensionality reduction implies information loss. The best low-dimensional space can be determined by best principal components. The major advantage of PCA is using it in eigenface approach which helps in reducing the size of the database for recognition of a test images. The images are stored as their feature vectors in the database which are found out projecting each and every trained image to the set of Eigen faces obtained. PCA is applied on Eigen face approach to reduce the dimensionality of a large data set.

DOI 10.17148/IJARCCE.2019.81012

2.2.2 Eigen Face Approach

It is adequate and efficient method to be used in face recognition due to its simplicity, speed and learning capability. Eigen faces are a set of Eigen vectors used in the Computer Vision problem of human face recognition. They refer to an appearance based approach to face recognition that seeks to capture the variation in a collection of face images and use this information to encode and compare images of individual faces in a holistic manner. The Eigen faces are Principal Components of a distribution of faces, or equivalently, the Eigen vectors of the covariance matrix of the set of the face images, where an image with N by N pixels is considered a point in N-dimensional space. Previous work on face recognition ignored the issue of face stimulus, assuming that predefined measurements were relevant and sufficient. This suggests that coding and decoding of face images may give information of face images emphasizing the significance of features. These features may or may not be related to facial features such as eyes, nose, lips and hairs. We want to extract the relevant information in a face image, encode it efficiently and compare one face encoding with a database of faces encoded similarly. A simple approach to extracting the information content in an image of a face is to somehow capture the variation in a collection of face images. We wish to find Principal Components of the distribution of faces, or the Eigen vectors of the covariance matrix of the set of face images. Each image location contributes to each Eigen vector, so that we can display the Eigen vector as a sort of face. Each face image can be represented exactly in terms of linear combination of the Eigen faces. The number of possible Eigen faces is equal to the number of face image in the training set. The faces can also be approximated by using best Eigen face, those that have the largest Eigen values, and which therefore account for most variance between the set of face images. The primary reason for using fewer Eigen faces is computational efficiency.

2.2.3 Eigen Values and Eigen Vectors

In linear algebra, the eigenvectors of a linear operator are non-zero vectors which, when operated by the operator, result in a scalar multiple of them. Scalar is then called Eigen value (λ) associated with the eigenvector (X). Eigen vector is a vector that is scaled by linear transformation. It is a property of matrix. When a matrix acts on it, only the vector magnitude is changed not the direction. $AX = \lambda X$, where A is a vector function. $(A - \lambda I)X = 0$, where I is the identity matrix. This is a homogeneous system of equations and form fundamental linear algebra. We know a non-trivial solution exists if and only if $Det(A - \lambda I) = 0$, where det denotes determinant. When evaluated becomes a polynomial of degree n. This is called characteristic polynomial of A. If A is N by N then there are n solutions or n roots of the characteristic polynomial. Thus there are n Eigen values of A satisfying the equation. $AXi = \lambda iXi$, where i=1,2,3,.....n. If the Eigen values are all distinct, there are n associated linearly independent eigenvectors, whose directions are unique, which span an n dimensional Euclidean space.

2.2.4 Face Image Representation

Training set of m images of size NxN are represented by vectors of size N 2. Each face is represented by $\Gamma 1, \Gamma 2, \Gamma 3, \Gamma M$.

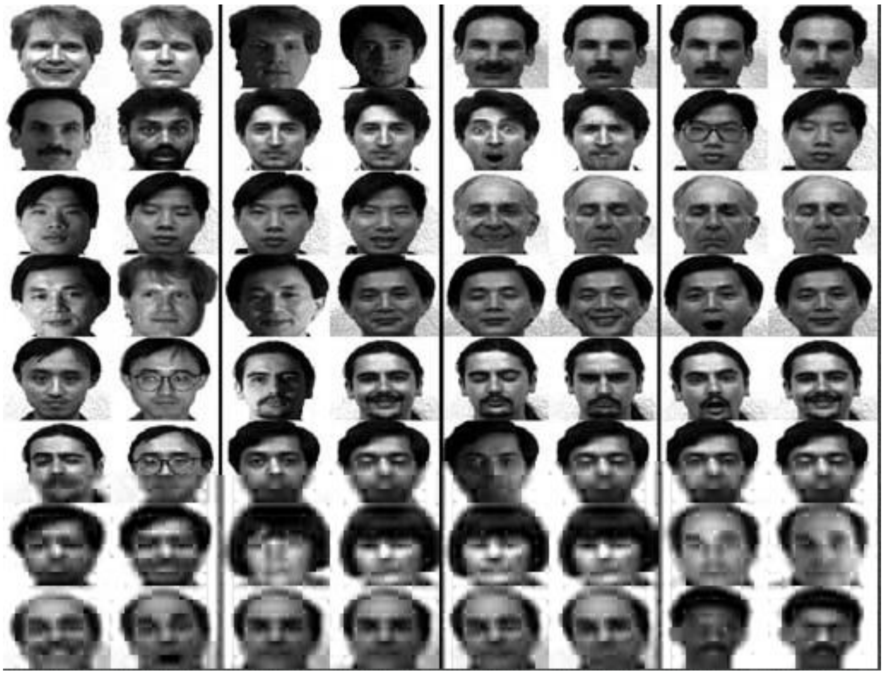


Fig 2.2.4 Face Image Representation

Feature vector of a face is stored in $N \times N$ matrix. Now, this two-dimensional vector is changed to one dimensional vector.

DOI 10.17148/IJARCCE.2019.81012

For Example-

$$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

Each face image is represented by the one dimensional vector Γi.

$$\Gamma 1 = \begin{pmatrix} 1 \\ -2 \\ 1 \\ -3 \end{pmatrix} \quad \Gamma 2 = \begin{pmatrix} 1 \\ 3 \\ -1 \\ 2 \end{pmatrix} \quad \Gamma 3 = \begin{pmatrix} 2 \\ 1 \\ -2 \\ 3 \end{pmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. \quad \Gamma M = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

2.2.5 Mean and Mean Centered Images

Average face image is calculated by

Ψ = (1/M) PM i=1 Γi .

$$\begin{pmatrix} 1 \\ -2 \\ 1 \\ -3 \end{pmatrix} + \begin{pmatrix} 1 \\ 3 \\ -1 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ -1 \\ -2 \\ 3 \end{pmatrix} +\ldots+\ldots. \begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} -1 \\ -1 \\ 2 \\ -3 \end{pmatrix}$$

Ψ = (Γ1 + Γ2 + Γ3 + ... + ΓM)/M. Each face differs from the average by Φi = Γi−Ψ which is called mean centered image.

$$\Phi = \begin{pmatrix} 2 \\ -1 \\ -1 \\ 0 \end{pmatrix} \quad \Phi 2 = \begin{pmatrix} 2 \\ 4 \\ -3 \\ 5 \end{pmatrix} \quad \Phi 3 = \begin{pmatrix} 3 \\ 2 \\ -4 \\ 6 \end{pmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad \Phi M = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 4 \end{pmatrix}$$

Then we have to find **Eigen Value and Eigen Vectors** with the help of Co-variance Matrix. The Co-variance Matrix can be found as follows:

2.2.6 Covariance Matrix

A covariance matrix is constructed as: $C = AA^T$, where A = [Φ1, Φ2 ,, ΦM] of size N 2 × N 2

$$A = \begin{pmatrix} 2 & 3 \\ -1 & -2 \\ -1 & 1 \\ 0 & 2 \end{pmatrix} \qquad A^T = \begin{pmatrix} 2 & -1 & -1 & 0 \\ 3 & -2 & 1 & 2 \end{pmatrix}$$

Size of covariance matrix will be $N2 \times N2$ (4x 4 in this case). Eigen vectors corresponding to this covariance matrix is needed to be calculated, but that will be a tedious task therefore, For simplicity we calculate A^TA which would be a 2 × 2 matrix in this case.

2.2.7 Eigen Face Space

The Eigen vectors of the covariance matrix AA^T are AXi which is denoted by Ui. Ui resembles facial images which look ghostly and are called Eigen faces. Eigen vectors correspond to each Eigen face in the face space and discard the faces for which Eigen values are zero thus reducing the Eigen face space to an extent. The Eigen faces are ranked according to their usefulness in characterizing the variation among the images.

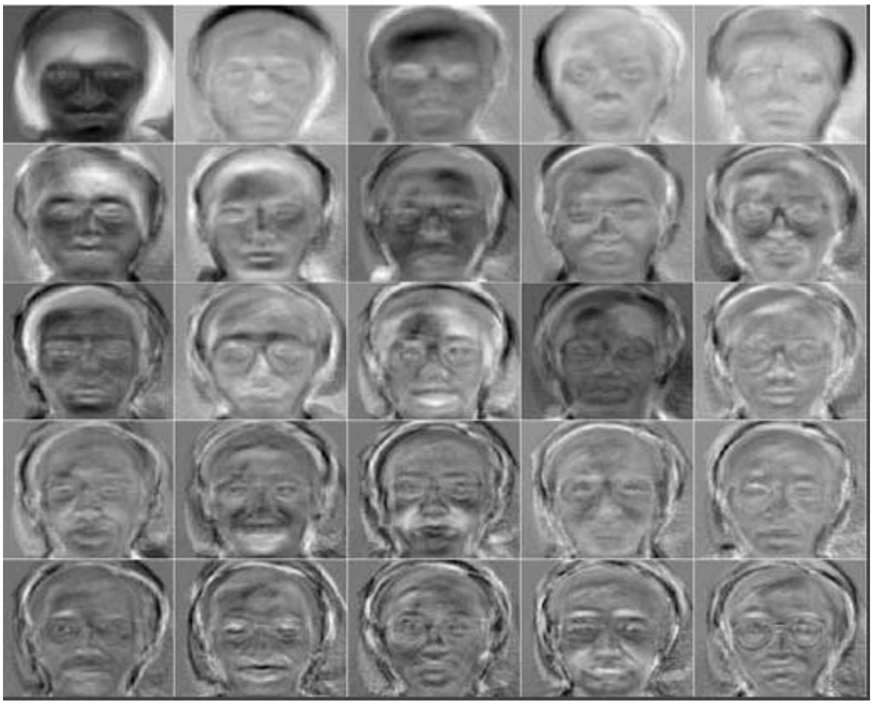


Fig 2.5 Eigen Faces

A face image can be projected into this face space by $\Omega_k = U^T(\Gamma_k - \Psi)$; k=1,....,M, where $(\Gamma_k \Psi)$ is the mean centered image.
Hence projection of each image can be obtained as Ω1 for projection of *image*1 and Ω2 for projection of *image*2 and hence forth.

2.3.8 Recognition Step

The test image, Γ, is projected into the face space to obtain a vector, Ω as

$$\Omega = U^T(\Gamma - \Psi)$$

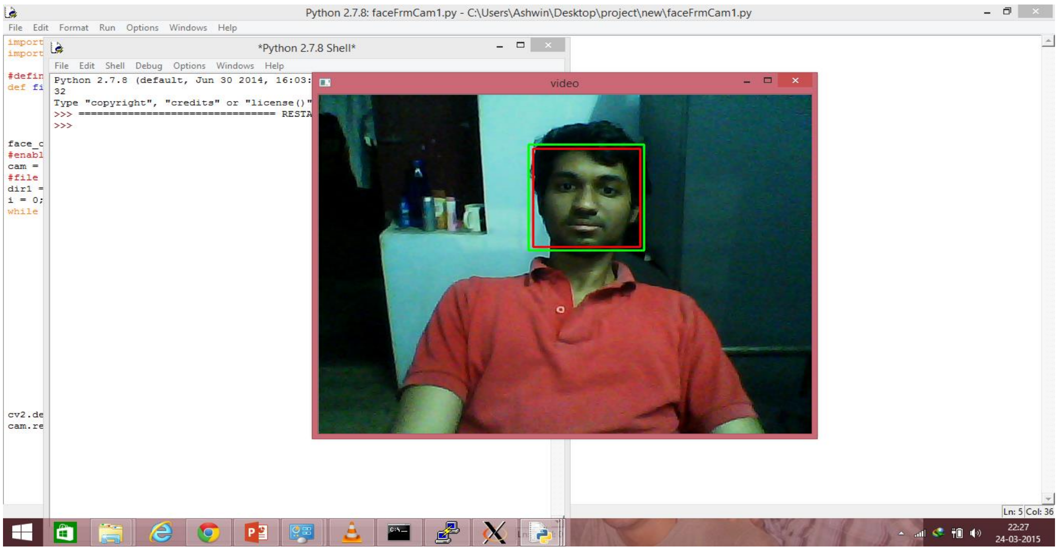
The distance of Ω to each face is called Euclidean distance and defined by

$$e^{2k} = ||\Omega - \Omega k||^2$$

k = 1,,M where Ωk is a vector describing the *kth* face class.

A face is classified as belonging to class k when the minimum ek is below some chosen threshold Θc. otherwise the face is classified as unknown.
Θc, is half the largest distance between any two face images:
$\Theta c = (1/2) maxj,k \, ||\Omega j - \Omega k||$; j,k = 1,....,M
We have to find the distance between the original test image Γ and its reconstructed image from the Eigen face Γf
$e2 = ||\Gamma - \Gamma f||2$, where $\Gamma f = U * \Omega + \Psi$
If $e \geq \Theta c$ then input image is not even a face image and not recognized.
If $e < \Theta c$ and $ek \geq \Theta$ for all k then input image is a face image but it is recognized as unknown face.
If $e < \Theta c$ and $k < \Theta$ for all k then input images are the individual face image associated with the class vector Ωk.

DOI 10.17148/IJARCCE.2019.81012


Python 2.7.8: faceFrmCam1.py - C:\Users\Ashwin\Desktop\project\new\faceFrmCam1.py
File Edit Format Run Options Windows Help
Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help
video
Ln: 5 Col: 36
22:27
24-03-2015

```
import
import

#defin
def fi



face_c
#enabl
cam =
#file
dir1 =
i = 0;
while


cv2.de
cam.re
```

```
Python 2.7.8 (default, Jun 30 2014, 16:03:
32
Type "copyright", "credits" or "license()"
>>> ================================ RESTA
>>>
```

Fig 2.3.8 Sample Detection of Face Object

III. SYSTEM SPECIFICATION

3.Implementation on Raspberry Pi

3.1 Description

The Raspberry Pi is a small, powerful and lightweight ARM based computer which can do many of the things a desktop PC can do. The powerful graphics capabilities and HDMI (High Definition Multiplexed Input) video output make it ideal for multimedia applications such as media centers and narrowcasting solutions. The Raspberry Pi is based on a Broadcom BCM2835 chip. It does not feature a built-in hard disk or solid-state drive, instead relying on an SD card for booting and long-term storage.

3.2 Model

Raspberry Pi Model B is used for our implementation.

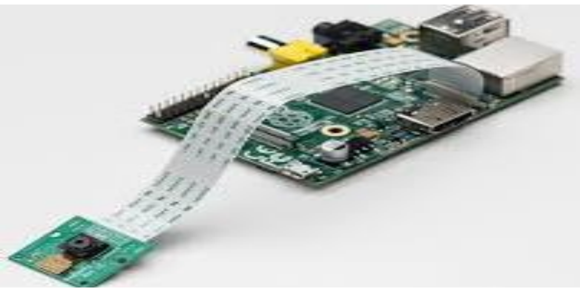


Fig 3.1 Raspberry Pi Model B

4.3.1 Specifications

Chip	Broadcom BCM2835 SoC
Core architecture	ARM11
CPU Clock	700 MHz
GPU	Dual Core VideoCore IV
Memory	512MB SDRAM
Operating System	Raspbian
Dimensions	85.6 x 53.98 x 17mm
Power	Micro USB socket 5V, 1.2A (l)
4.3.2 Connectors	
Ethernet	10/100 BaseT Ethernet Sockets
Video Output	HDMI
Audio Output	3.5mm jack
USB 2.0	Dual USB Connector

DOI 10.17148/IJARCCE.2019.81012

GPIO Connector	26-pin 2.54 mm (100 mil) expansion header: 2x13 strip. Providing 8 GPIO pins plus access to I²C, SPI & URAT as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface
Memory Card Slot	SDIO

4.3.3 Terminal Requirements

Putty, Xming are required to launch R-Pi home screen in Windows Operating System.

4.4 Raspberry Pi Camera

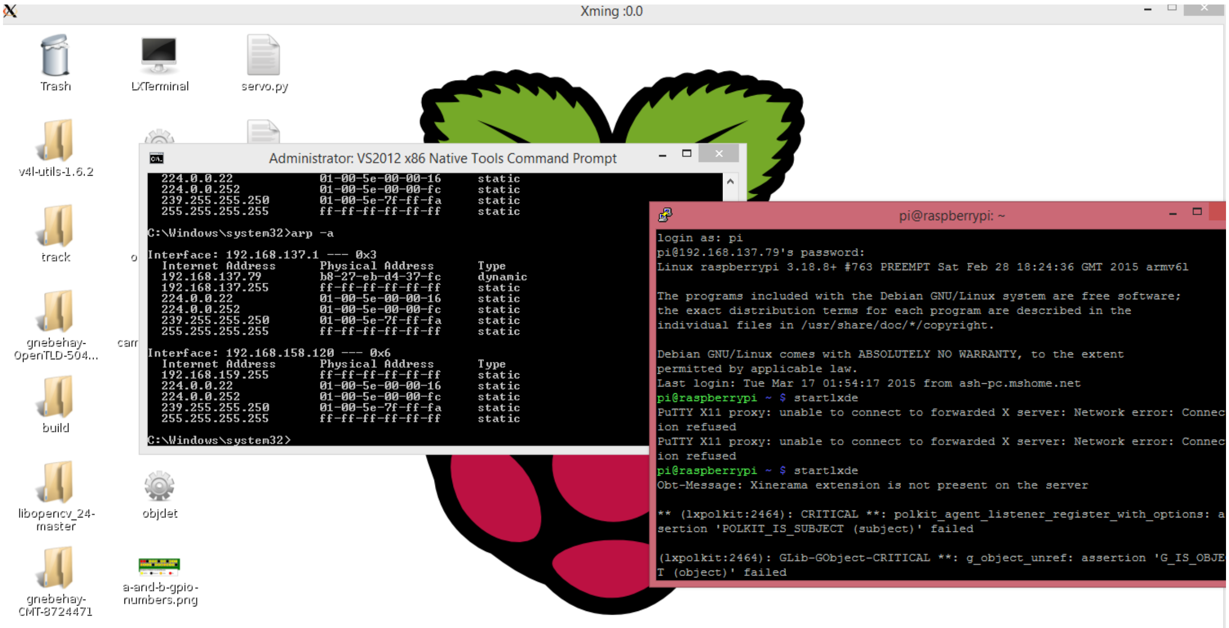
The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. You can also use the libraries we bundle with the camera to create effects. The module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

The camera works with all models of Raspberry Pi 1 and 2. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library. The camera module is very popular in home security applications, and in wildlife camera traps.

4.5 Setting up Raspberry Pi

The Raspberry Pi Camera is mounted on the Raspberry pi board with SD card loaded with Raspbian OS. We can either use a HDMI cable as a video output through devices like Television, projector etc. We can also use Ethernet Cable to access our R-Pi Kit through laptop. The steps involved are as follows.

Step 1: Change the network adapter settings to share Wi-Fi connection via Ethernet.
Step 2: Open "Command Prompt" (Run as administrator) and type arp –a
Step 3: A Dynamically generated IP Address will be displayed on the screen. Note that we can also configure static IP Address for user's convenience
Step 4: Open Putty Terminal and type the configured IP address. In order to view the raspberry pi home page, we have to enable X11 Forwarding in SSH settings.
Step 5: Now Open Xlaunch and type startlxde to enable a new lxsession



```
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Windows\system32>arp -a

Interface: 192.168.137.1 --- 0x3
  Internet Address      Physical Address      Type
  192.168.137.79        b8-27-eb-d4-37-fc     dynamic
  192.168.137.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.158.120 --- 0x6
  Internet Address      Physical Address      Type
  192.168.159.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

C:\Windows\system32>
```

```
login as: pi
pi@192.168.137.79's password:
Linux raspberrypi 3.18.8+ #763 PREEMPT Sat Feb 28 18:24:36 GMT 2015 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 17 01:54:17 2015 from ash-pc.mshome.net
pi@raspberrypi ~ $ startlxde
PuTTY X11 proxy: unable to connect to forwarded X server: Network error: Connec
ion refused
PuTTY X11 proxy: unable to connect to forwarded X server: Network error: Connec
ion refused
pi@raspberrypi ~ $ startlxde
Obt-Message: Xinerama extension is not present on the server

** (lxpolkit:2464): CRITICAL **: polkit_agent_listener_register_with_options: a
sertion 'POLKIT_IS_SUBJECT (subject)' failed

(lxpolkit:2464): GLib-GObject-CRITICAL **: g_object_unref: assertion 'G_IS_OBJE
T (object)' failed
```

Fig 4.2 Getting Started with Raspbian OS

Click on lxterminal to execute various functionalities of the R-Pi Modules.

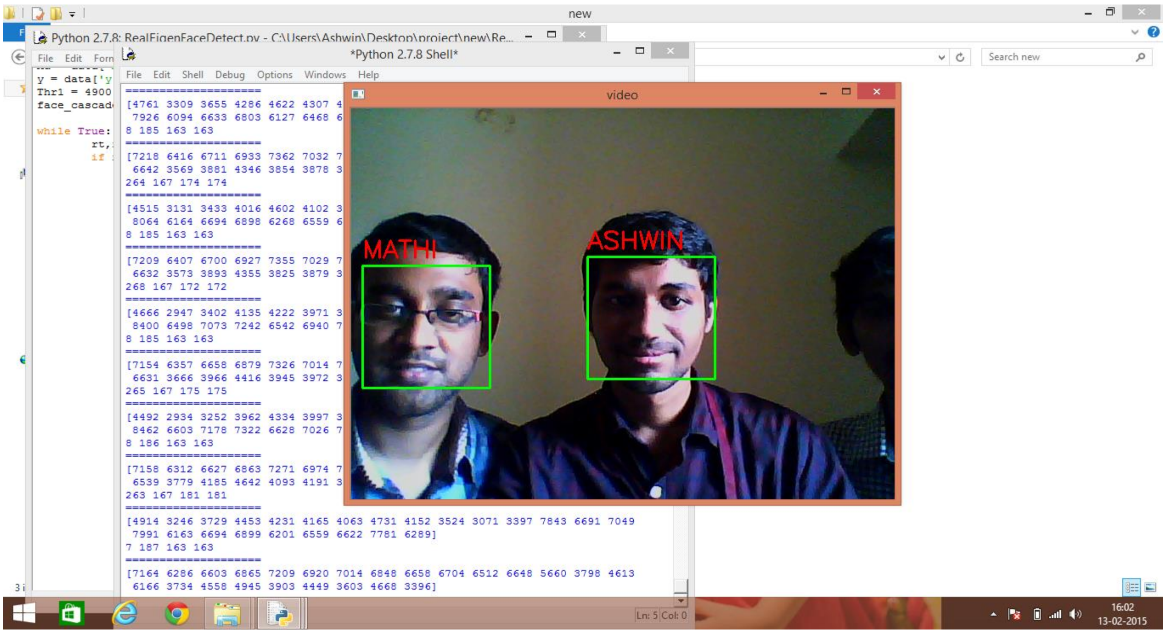
DOI 10.17148/IJARCCE.2019.81012

V. INFERENCES

The following results are obtained based on the implementation on the R-Pi board

4.6.1 Detecting a Known Face

After image acquisition, the PCA algorithm is implemented and threshold value is calculated accordingly. When a new query image is acquired, it compares the values with the stored values and matches it. The message "Known Face" will be displayed only if the value is within a certain threshold value range. If not, the message "Unknown user" is displayed.


new
Python 2.7.8: RealEigenFaceDetect.py - C:\Users\Ashwin\Desktop\project\new\Re...
File Edit Form
Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help
video
MATHI
ASHWIN
Search new
Ln: 5 Col: 0
16:02
13-02-2015

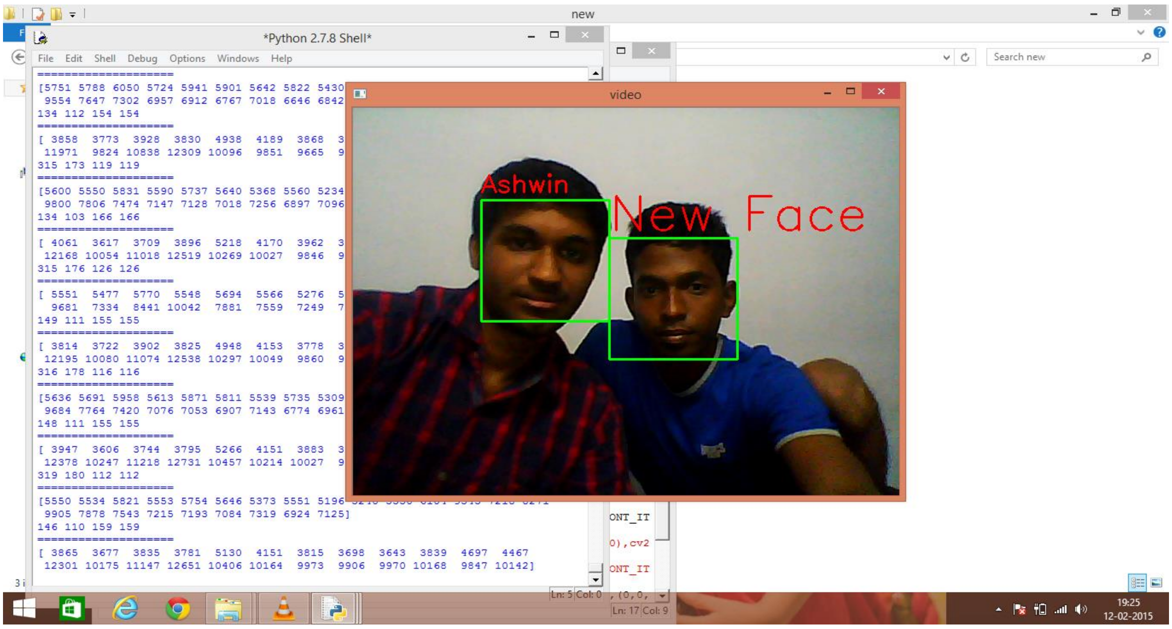
```
y = data['y
Thr1 = 4900
face_cascad

while True:
        rt,
        if
```

```
=====================
[4761 3309 3655 4286 4622 4307 4
 7926 6094 6633 6803 6127 6468 6
8 185 163 163
=====================
[7218 6416 6711 6933 7362 7032 7
 6642 3569 3881 4346 3854 3878 3
264 167 174 174
=====================
[4515 3131 3433 4016 4602 4102 3
 8064 6164 6694 6898 6268 6559 6
8 185 163 163
=====================
[7209 6407 6700 6927 7355 7029 7
 6632 3573 3893 4355 3825 3879 3
268 167 172 172
=====================
[4666 2947 3402 4135 4222 3971 3
 8400 6498 7073 7242 6542 6940 7
8 185 163 163
=====================
[7154 6357 6658 6879 7326 7014 7
 6631 3666 3966 4416 3945 3972 3
265 167 175 175
=====================
[4492 2934 3252 3962 4334 3997 3
 8462 6603 7178 7322 6628 7026 7
8 186 163 163
=====================
[7158 6312 6627 6863 7271 6974 7
 6539 3779 4185 4642 4093 4191 3
263 167 181 181
=====================
[4914 3246 3729 4453 4231 4165 4063 4731 4152 3524 3071 3397 7843 6691 7049
 7991 6163 6694 6899 6201 6559 6622 7781 6289]
7 187 163 163
=====================
[7164 6286 6603 6865 7209 6920 7014 6848 6658 6704 6512 6648 5660 3798 4613
 6166 3734 4558 4945 3903 4449 3603 4668 3396]
```

Fig 4.6.1 Implementation Model of Known face pattern.

4.6.2 Detecting a New User

Consider the next case, where a new user is trying to access the system. It must display a message "I can't recognize you" once the user's face gets recognized. The values obtained as a result of image acquisition are compared with the new query image and since it doesn't match any stored values, the message new face is displayed.



```
=====================
[5751 5788 6050 5724 5941 5901 5642 5822 5430
 9554 7647 7302 6957 6912 6767 7018 6646 6842
134 112 154 154
=====================
[ 3858  3773  3928  3830  4938  4189  3868  3
 11971  9824 10838 12309 10096  9851  9665  9
315 173 119 119
=====================
[5600 5550 5831 5590 5737 5640 5368 5560 5234
 9800 7806 7474 7147 7128 7018 7256 6897 7096
134 103 166 166
=====================
[ 4061  3617  3709  3896  5218  4170  3962  3
 12168 10054 11018 12519 10269 10027  9846  9
315 176 126 126
=====================
[ 5551  5477  5770  5548  5694  5566  5276  5
  9681  7334  8441 10042  7881  7559  7249  7
149 111 155 155
=====================
[ 3814  3722  3902  3825  4948  4153  3778  3
 12195 10080 11074 12538 10297 10049  9860  9
316 178 116 116
=====================
[5636 5691 5958 5613 5871 5811 5539 5735 5309
 9684 7764 7420 7076 7053 6907 7143 6774 6961
148 111 155 155
=====================
[ 3947  3606  3744  3795  5266  4151  3883  3
 12378 10247 11218 12731 10457 10214 10027  9
319 180 112 112
=====================
[5550 5534 5821 5553 5754 5646 5373 5551 5196
 9905 7878 7543 7215 7193 7084 7319 6924 7125]
146 110 159 159
=====================
[ 3865  3677  3835  3781  5130  4151  3815  3698  3643  3839  4697  4467
 12301 10175 11147 12651 10406 10164  9973  9906  9970 10168  9847 10142]
```

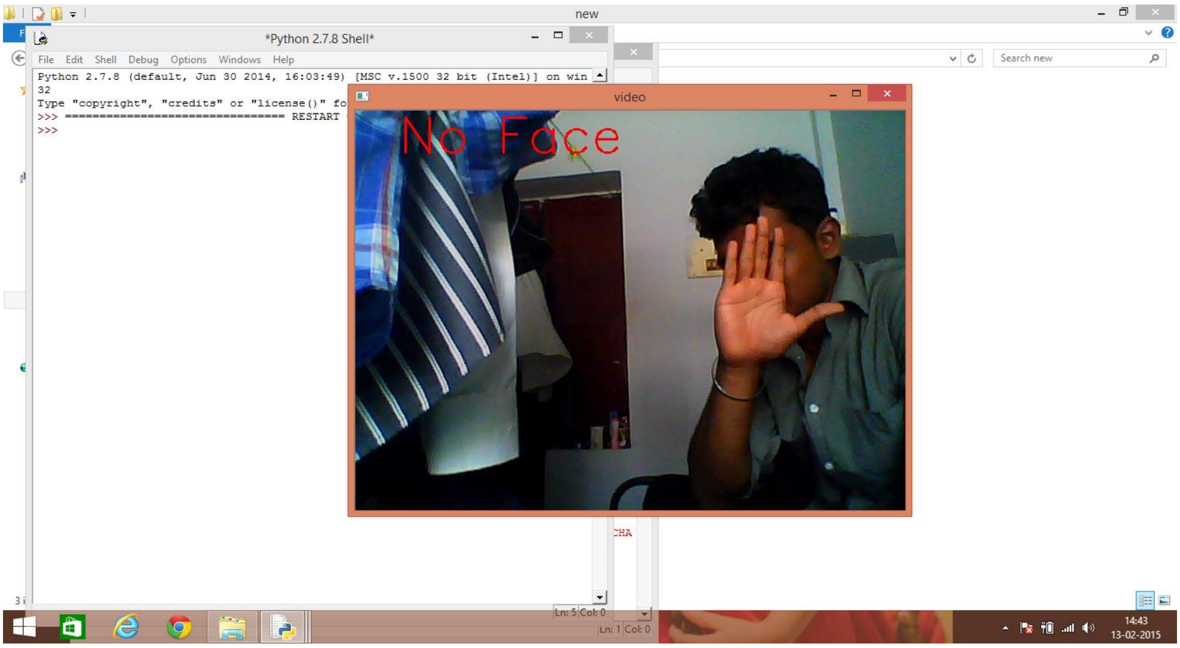
```
ONT_IT
0),cv2
ONT_IT
,(0,0,
```

Fig 4.6.2 Implementation Model of New face pattern

DOI 10.17148/IJARCCE.2019.81012

4.6.3 No Face Condition

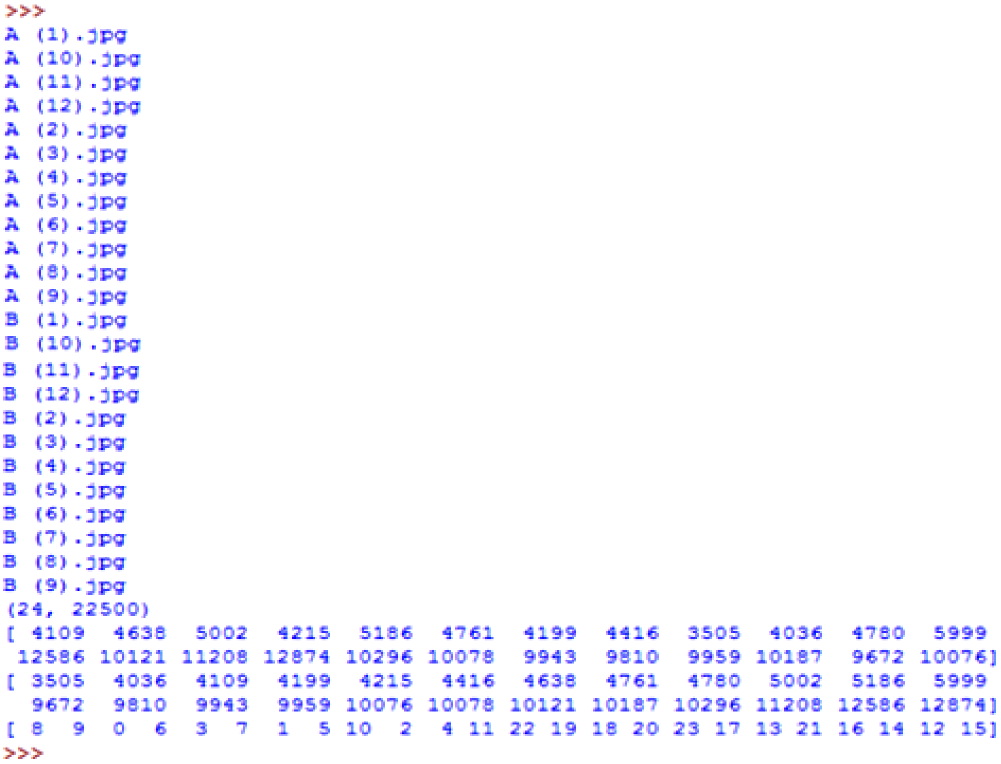
By including the haar cascade XML file, it is evident that it will detect only if a face is found. In case, if there is no clear view of the face, the message “Sorry! No Face Found is Displayed”



```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" fo
>>> ================================ RESTART
>>>
```

Fig 4.6.3 Implementation Model of No face pattern

4.6.5 Threshold Value Calculation



```
>>>
A (1).jpg
A (10).jpg
A (11).jpg
A (12).jpg
A (2).jpg
A (3).jpg
A (4).jpg
A (5).jpg
A (6).jpg
A (7).jpg
A (8).jpg
A (9).jpg
B (1).jpg
B (10).jpg
B (11).jpg
B (12).jpg
B (2).jpg
B (3).jpg
B (4).jpg
B (5).jpg
B (6).jpg
B (7).jpg
B (8).jpg
B (9).jpg
(24, 22500)
[ 4109  4638  5002  4215  5186  4761  4199  4416  3505  4036  4780  5999
 12586 10121 11208 12874 10296 10078  9943  9810  9959 10187  9672 10076]
[ 3505  4036  4109  4199  4215  4416  4638  4761  4780  5002  5186  5999
  9672  9810  9943  9959 10076 10078 10121 10187 10296 11208 12586 12874]
[ 8  9  0  6  3  7  1  5 10  2  4 11 22 19 18 20 23 17 13 21 16 14 12 15]
>>>
```

Fig 4.7 Threshold value Calculation

The values above represent the threshold values (Ɵ). It is clear that first set of images has a certain range which deviates from the next set of values. Since, the threshold value Ɵc, is half the largest distance between any two face images, the value in between the two sets of faces can be set as threshold.

DOI 10.17148/IJARCCE.2019.81012

VI. CONCLUSION

Thus over the past few decades with the growing need to secure information, face recognition has become an essential need. Further enhancement can be done by extracting the key features or point detection over face matrix. But Eigen Face Algorithm has got its unique importance since it can recognise ‘n’ faces in a single frame. With an easy computational method and relatively simple algorithm, Principal Component Analysis Technique can be used for variety of real time applications.

REFERENCES

[1]. A. Bazen, R. Veldhuis, and G. Croonen. Likelihood ratio-based detection of facial features. In Proc. ProRISC 2003, 14th Annual Workshop on Circuits, Systems and Signal Processing, pages 323–329, Veldhoven, The Netherlands, nov 2003.
[2]. P. N. Belhumeur, J. Hespanha, & D.J.Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. In ECCV 2, 1996
[3]. G. Beumer, A. Bazen, and R. Veldhuis. On the accuracy ofeers inface recognition andthe importance of reliable registration. In SPS 2005. IEEE Benelux/DSP Valley, April 2005.
[4]. D. Cristinacce and T. Cootes. A comparison of shape constrained facial feature detectors. In 6th International Conference on Automatic Face and Gesture Recognition 2004, Seoul, Korea, pages 375–380, 2004.
[5]. D. Cristinacce, T. Cootes, and I. Scott. A multi-stage approach to facial feature detection. In 15th British Machine Vision Conference, London, England, pages 277– 286, 2004.
[6]. P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge
[7]. Pankaj R.Bhusari, Vrushali G. Raut**,** “Real Time Face Detection and Recognition Systems”
[8]. M. Turk and A. Pentland. Eigenfaces for recognition. Journal of cognative neuroscience
[9]. R. C. Gonzalez, R. E. Woods, "Digital Image Processing".

DOI 10.17148/IJARCCE.2019.81012