Real Time Face Recognition

A PROJECT REPORT

Submitted by

AK. KARTHIK

Roll No: R131080

in partial fulfilment of project for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & COMMUNICATION ENGINEERING

Under the Guidance of

Mr. Mohan Raju

Lecturer

AP IIIT RKVALLEY, RGUKT

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, ANDHRA PRADESH

April - 2019

Rajiv Gandhi University of Knowledge Technologies
(A.P. Government Act 18 of 2008)
IIIT RK Valley, RGUKT-A.P.
RK Valley (Idupulapaya), Vempalli (M), Y.S.R. Kadapa (Dist.), A.P – 516330

CERTIFICATE

This is certified that the project report on "**Real Time Face Recognition**" is the bonafied work of **Ak. Karthik (R131080)** final year B.Tech in Electronics and Communication Engineering of IIIT-AP,Rajiv Gandhi University of Knowledge Technologies (RGUKT) RK Valley, Andhra Pradesh has carried out the work under the supervision.

Mr.Mohan Raju,

Lecturer of ECE .

Mr.A.Sreekanth Reddy,

Head of the Department.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES,

R.K VALLEY, KADAPA, AP, INDIA, PINCODE: 516330.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected Director, **Prof. H. Sudarsana Rao** for fostering an excellent academic climate in our institution. I also express my sincere gratitude to our respected Head of the Department **Mr. A. Sreekanth Reddy** for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide **Mr. Mohan Raju** for his guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

Last but not the least we also thank our friends and family members for helping us in completing the project.

AK. KARTHIK (R131080)

DECLARATION

I do here by declare that the final year project work titled by **"REAL TIME FACE RECOGNITION"** submitted by AK. KARTHIK (R131080) to the RGUKT in partial fulfilment of the requirement for the award of bachelor of technology in Electronics & Communication Engineering is my original work where others' ideas or words included, I have adequately cited and referenced the original sources. The analysis, design and implementation of this project have been done by me and my batch mates and it has not submitted anywhere else for the award of the degree.

(Signature of the student)

Ak. Karthik (R131080)

DATE: 27/04/2019

ABSTRACT

The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity.

Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. The area of this project face detection system with face recognition is Image processing. The software requirements for this project is python software.

Keywords: face detection, LBPH, Python, OpenCV, Haar Cascade classifier

Extension: There are vast number of applications from this face detection project, this project can be extended that the various parts in the face can be detect which are in various directions and shapes.

TABLE OF CONTENTS **Page No.**

CHAPTER 1

INTRODUCTION

1.1 Aim of the project:

The goal of this project is to provide an easier human-machine interaction routine when user authentication is needed through face detection and recognition.

With the aid of a regular web camera, a machine is able to detect and recognize a person's face; a custom login screen with the ability to filter user access based on the users' facial features will be developed.

The objectives of this thesis are to provide a set of detection algorithms that can be later packaged in an easily- portable framework amongst the different processor architectures we see in machines (computers) today. These algorithms must provide at least a 95% successful recognition rate, out of which less than 3% of the detected faces are false positives.

In these project we created a Real time face recognition system for that we have to create dataset of face that we have to recognize. And the camera to detect faces from the camera framers. And we have to train the dataset to detect the faces. Then the faces which are detected in the current frame are searched in our dataset if the face matched with any image in the dataset it shows their respective name.

1.2 The History of Face Recognition:

Face recognition began as early as 1977 with the first automated system being introduced By Kanade using a feature vector of human faces. In 1983, Sirovich and Kirby introduced the principal component analysis(PCA) for feature extraction. Using PCA, Turk and Pentland Eigenface was developed in 1991 and is considered a major milestone in technology. Local binary pattern analysis for texture recognition was introduced in 1994 and is improved upon for facial recognition later by incorporating Histograms(LBPH). In 1996 Fisherface was developed using Linear discriminant analysis (LDA) for dimensional reduction and can identify faces in different illumination conditions, which was an issue in Eigenface method. Viola and Jones introduced a face detection technique using HAAR cascades and ADABoost. In 2007, A face recognition technique was developed by Naruniec and Skarbek using Gabor Jets that are similar to mammalian eyes. In This project, HAAR cascades are used for face detection and Eigenface, Fisherface and LBPH are used for face recognition.

1.2 Problem Definition:

Over the past decade face detection and recognition have transcended from esoteric to popular areas of research in computer vision and one of the better and successful applications of image analysis and algorithm based understanding.

Because of the intrinsic nature of the problem, computer vision is not only a computer science area of research, but also the object of neuro-scientific and psychological studies also, mainly because of the general opinion that advances in computer image processing and understanding research will provide insights into how our brain work and vice versa.

A general statement of the face recognition problem (in computer vision) can be formulated as follows: given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces.

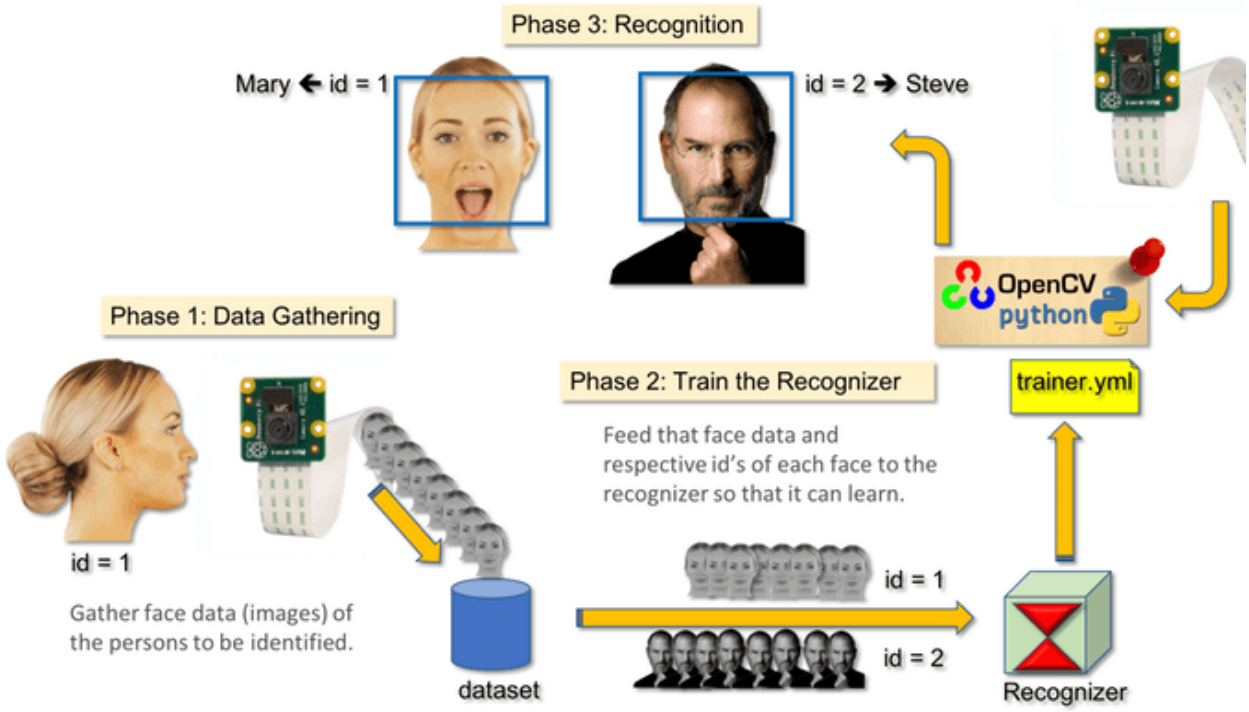
Facial recognition generally involves two stages:

1) **Face Detection** where a photo is searched to find a face, then the image is processed to crop and extract the person's face for easier recognition.
2) **Face Recognition** where that detected and processed face is compared to a database of known faces, to decide who that person is.

CHAPTER 2

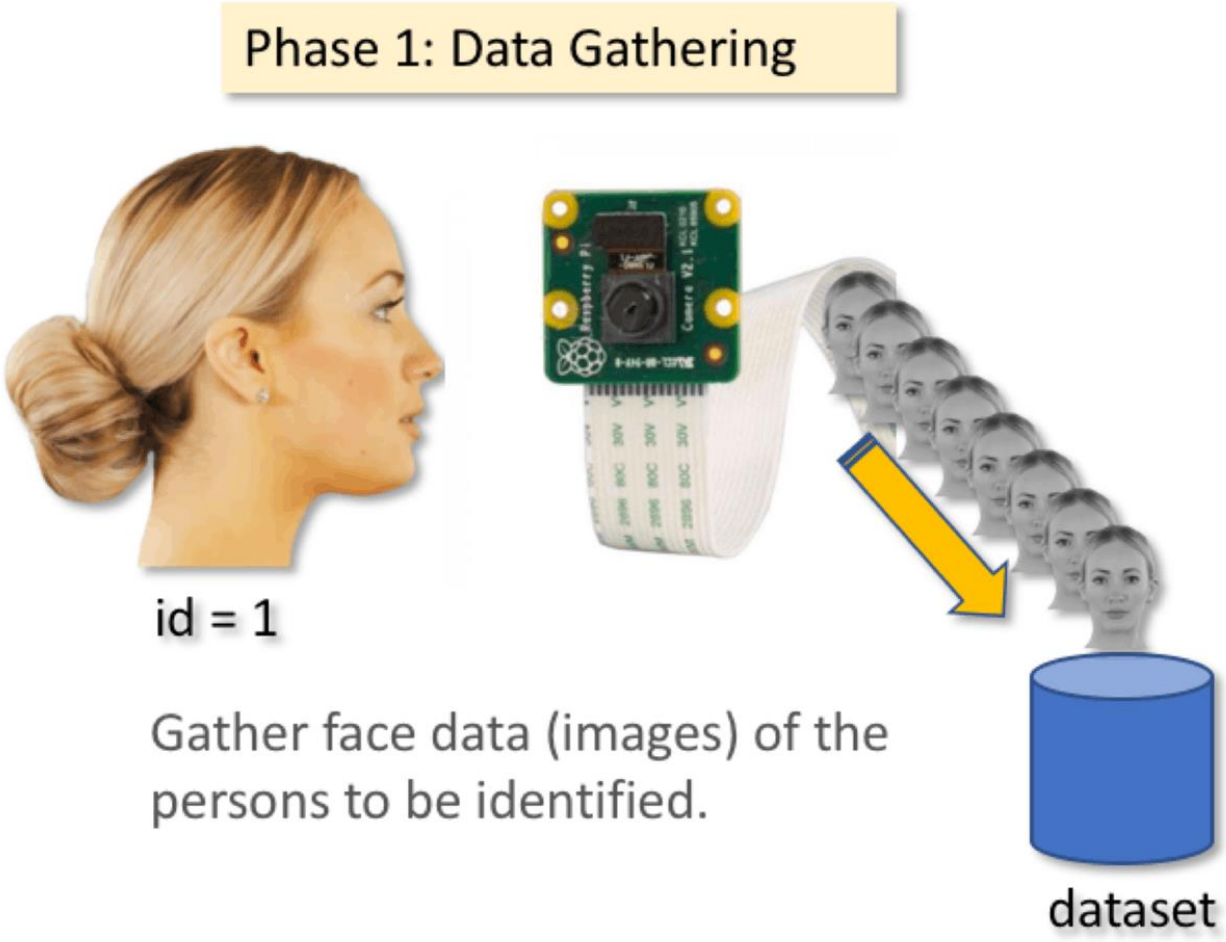
WORKING PRINCIPLE

To make a face recognition program, first we need to train the recognizer with dataset of previously captured faces along with its ID, for example we have two person then first person will have ID 1 and 2nd person will have ID 2, so that all the images of person one in the dataset will have ID 1 and all the images of the 2nd person in the dataset will have ID 2, then we will use those dataset images to train the recognizer to predict the 1 of an newly presented face from the live video frame.



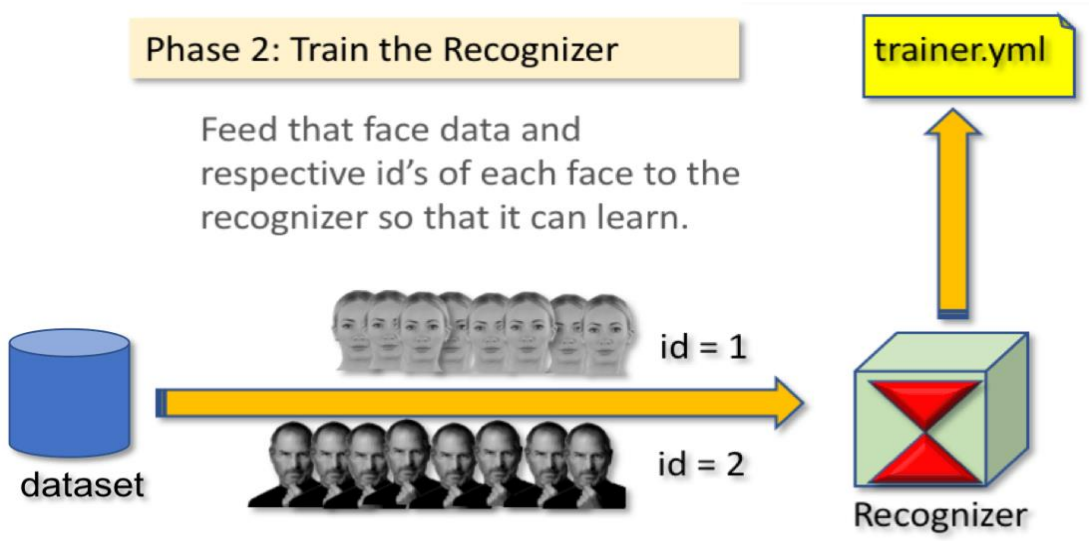
In this project our entire task is divided into 3 parts

1.1 Dataset Creater:



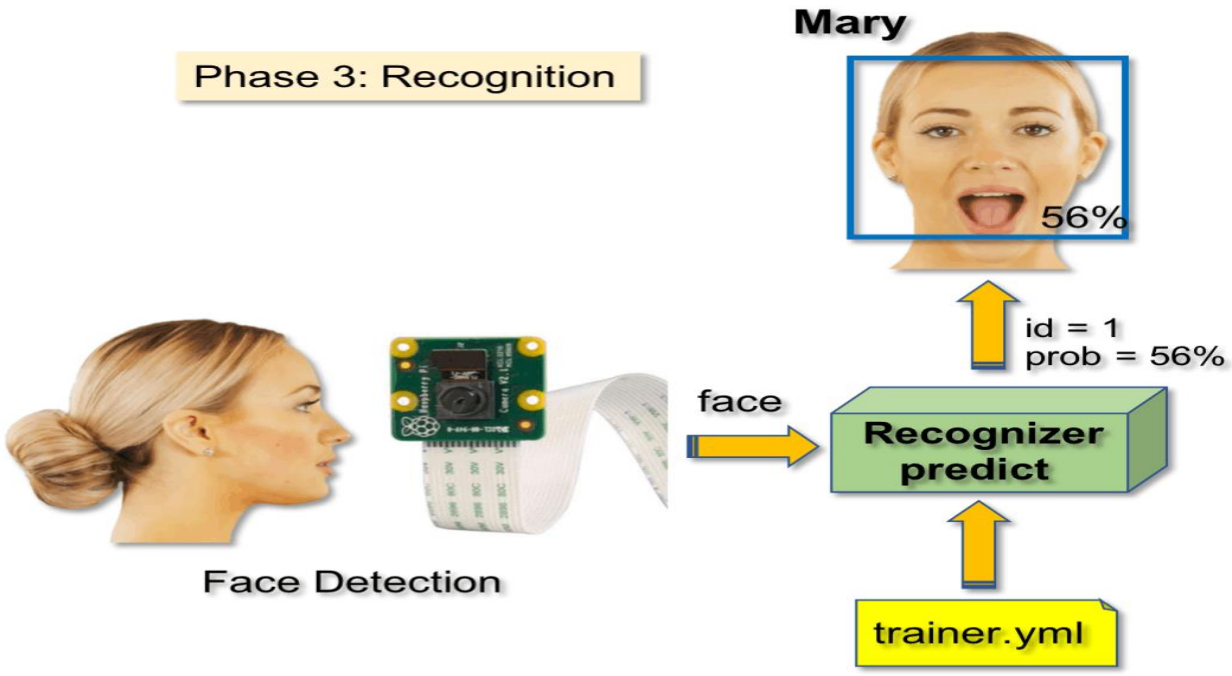
To train the person we have to take his sample and has to save with a unique id so we are taking 25 number of face samples from the user and saving with his unique id as dataset.

2.2 Dataset Trainer:



To train the persons in the dataset we have to read each image in dataset and has to save it with it's appropriate id we are storing all this in a file trainner.yml.

2.3 Detector:



We are reading frame from camera object and finding the faces with harscade model and doing the face recognization for each face in this process we are loading trainner.yml the faces found in camera searches their nearhood face in trainner.yml and if probability is less than 50 it predicts as that id otherwise it just skips or show it as Unknown.

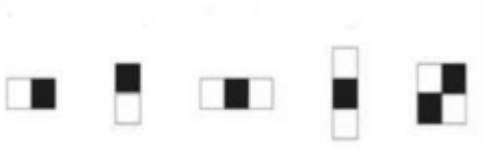
CHAPTER 3

HAAR CASCADE CLASSIFIER

3.1 Haar features:

For the detection of the face, haar features are the main part of haar cascade classifier.Haar features are the set of adjacent rectangles of pixels.Haar features are used to detect the presence of feature in the given image.Each feature results in a single value which is calculated by subtracting the sum of pixels under white rectangle from sum of pixels under black rectangle.

P(x)=Sum(black rectangle pixels)-Sum(white rectangle pixels



Haar features

In human face eyes are darker than the cheek and forehead. So to detect the eyes we will use the second feature. This feature results a maximum value only in the eyes region. Nose is brighter than the edges. So that we can detect nose by using one of these haar features.

Eyes and nose detection using haar features

Haar features starts scanning the image from the top left corner and ends at the right corner of the image. Viola Jones algorithm uses a 24x24 window as base window size to start evaluating these features in any given image.

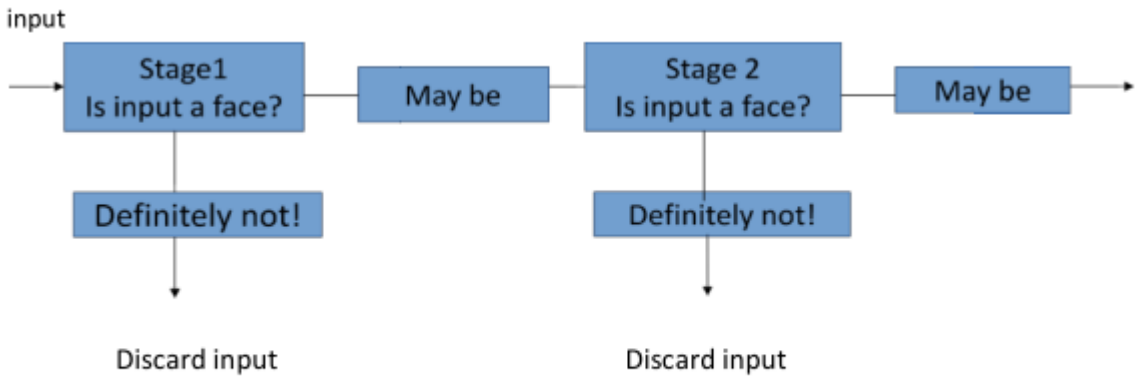
Scanning the image

Features can be extracted at any location with any scale. If we consider location, scale and type of the haar feature there are about 1.60,000+ features need to be calculated with each 24x24 window. But not all of them are useful.

3.2 Cascading:

The basic principle of the viola jones face detection algorithm is to scan image many times-each time with new feature with new size. Even the image doesn't contain a face we are calculating all the features which increases the computation cost. So the algorithm should concentrate on discarding non-faces quickly and spend more time on probable face region. Hence a single strong classifier formed out of linear combination of all best features is not good to evaluate on each window because of computation cost.

A cascade classifier is used which is composed of stages each contaning a strong classifier.So all the features are grouped into several stages where each stage has certain number of features. The job of each stage is to determine whether a given sub window is definitely not a face or may be a face.



A Haar wavelet is a mathematical fiction that produces square-shaped waves with a beginning and an end and used to create box shaped patterns to recognise signals with sudden transformations. An example is shown in figure 1. By combining several wavelets, a cascade can be created that can identify edges, lines and circles with different colour intensities. These sets are used in Viola Jones face detection technique in 2001 and since then more patterns are introduced for object detection as shown in figure 1. To analyse an image using Haar cascades, a scale is selected smaller than the target image. It is then placed on the image, and the average of the values of pixels in each section is taken. If the difference between two values pass a given threshold, it is considered a match. Face detection on a human face is performed by matching a combination of different Haar-like-features. For example, forehead, eyebrows and eyes contrast as well as the nose with eyes as shown below in figure A single classifier is not accurate enough. Several classifiers are combined as to provide an accurate face detection system as shown in the block diagram below in figure 3.

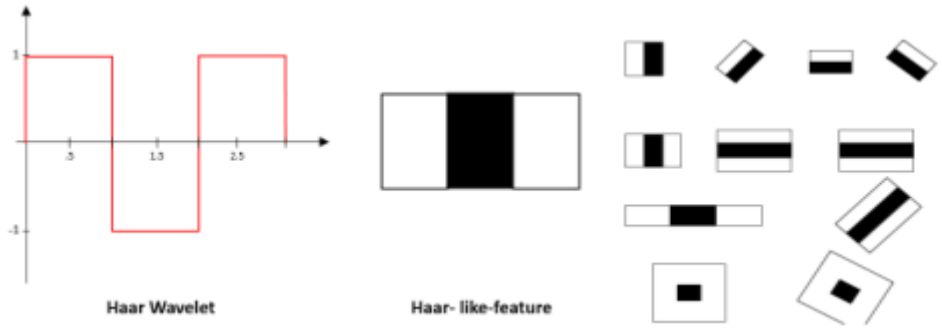


Figure 1: A Haar wavelet and resulting Haar-like features.

In this project, a similar method is used effectively to by identifying faces and eyes in combination resulting better face detection. Similarly, in viola Jones method , several classifies were combined to create stronger classifiers. ADA boost is a machine learning algorithm that tests out several week classifiers on a selected location and choose the most suitable. It can also reverse the direction of the classifier and get better results if necessary. Furthermore, Weight-update-steps can be updated

Figure 2: Several Haar-like-features matched to the features of authors face

only on misses to get better performance. The cascade is scaled by 1.25 and re-iterated in order to find different sized faces. Running the cascade on an image using conventional loops takes a large amount of computing power and time. Viola Jones used a summed area table (an integral image) to compute the matches fast. First developed in 1984 , it became popular after 2001 when Viola Jones implemented Haar-cascades for face detection. Using an integral image enables matching features with a single pass over the image.

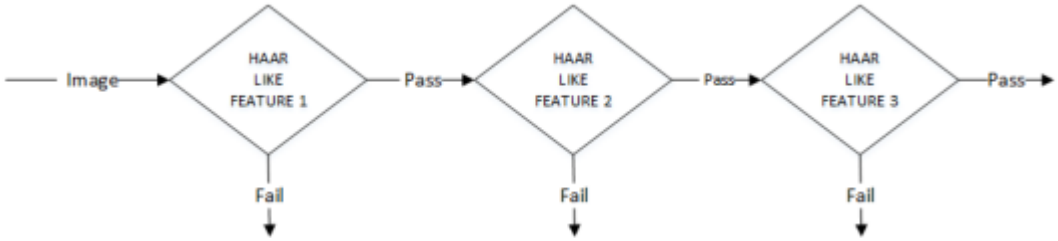


Figure 3: Haar-cascade flow chart

CHAPTER 4

FACE RECOGNITION USING LBPH

Local Binary Pattern Histogram(LBPH):

Local binary patterns were proposed as classifiers in computer vision and in 1990 By Li Wang. The combination of LBP with histogram oriented gradients was introduced in 2009 that increased its performance in certain datasets. For feature encoding, the image is divided into cells (4 x 4 pixels). Using a clockwise or counter-clockwise direction surrounding pixel values are compared with the central as shown in figure 6. The value of intensity or luminosity of each neighbour is compared with the centre pixel. Depending if the difference is higher or lower than 0, a 1 or a 0 is assigned to the location. The result provides an 8-bit value to the cell. The advantage of this technique is even if the luminosity of the image

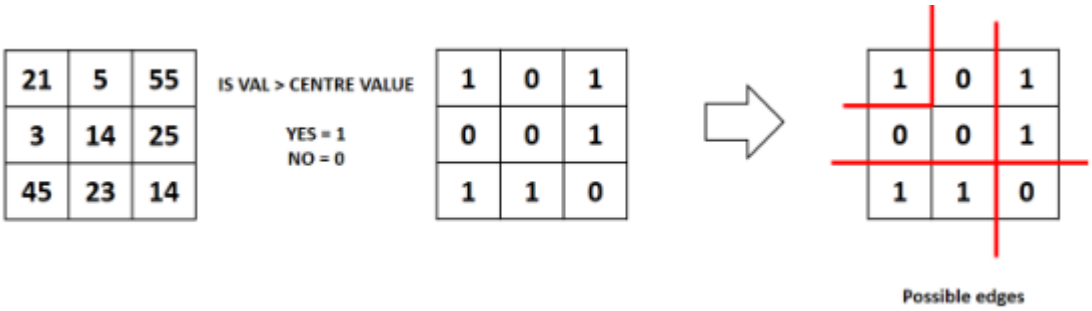


Figure 6: Local binary pattern histogram generating 8-bit number

is changed as in figure 7, the result is the same as before. Histograms are used in larger cells to find the frequency of occurrences of values making process faster. By analysing the results in the cell, edges can be detected as the values change. By computing the values of all cells and concatenating the histograms, feature vectors can be obtained. Images can be classified by processing with an ID attached. Input images are classified using the same process and compared with the dataset and distance is obtained. By setting up a threshold, it can be identified if it is a known or unknown face. Eigenface and Fisherface compute the dominant features of the whole training set while LBPH analyse them individually.

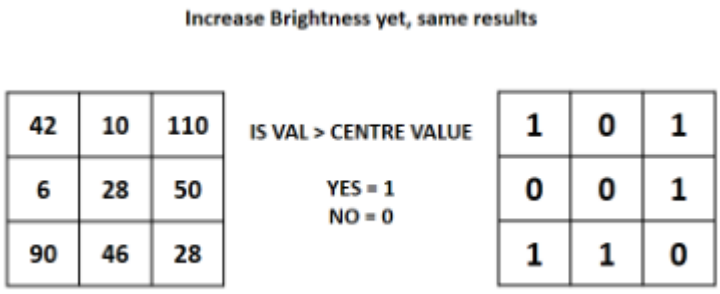


Figure 7: The results are same even if brightness is changed

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number. Using the LBP combined with histograms we can represent the face images with a simple data vector. Now that we know a little more about face recognition and the LBPH, let's go further and see the

steps of the algorithm:

Parameters: the LBPH uses 4 parameters:

• **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.

• **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind the more sample points you include, the higher the computational cost. It is usually set to 8.

• **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

• **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

CHAPTER 5

METHODOLOGY

Below are the methodology and descriptions of the applications used for data gathering, face detection, training and face recognition. The project was coded in Python using a mixture of IDLE and PYCharm IDEs.

5.1 Face Detection:

First stage was creating a face detection system using Haar-cascades. Although, training is required for creating new Haar-cascades, OpenCV has a robust set of Haar-cascades that was used for the project. Using face-cascades alone caused random objects to be identified and eye cascades were incorporated to obtain stable face detection. The flowchart of the detection system can be seen in figure 8. Face and eye

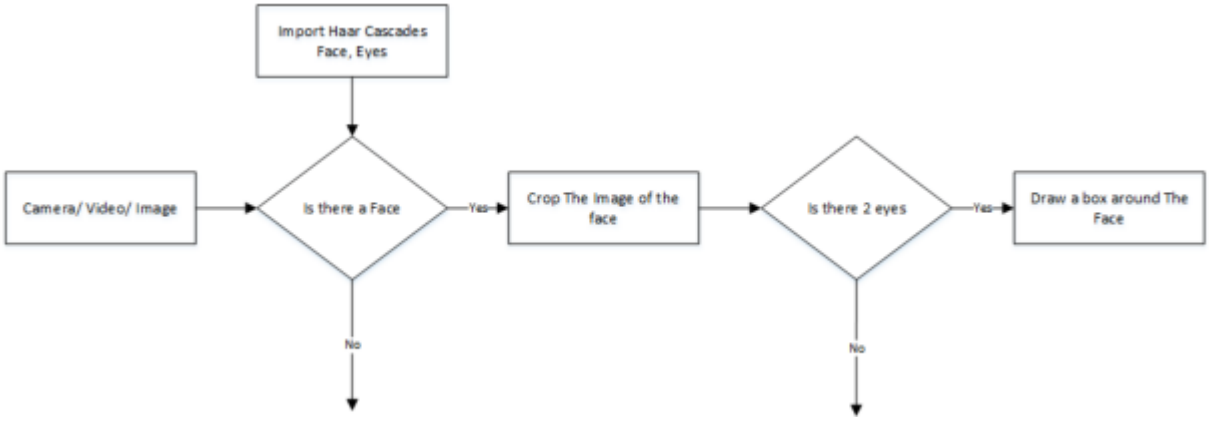


Figure 8: The Flow chart of the face detection application

classifier objects are created using classifier class in OpenCV through the cv2.CascadeClassifier() and loading the respective XML files. A camera object is created using the cv2.VideoCapture() to capture images. By using the CascadeClassifier.detectMultiScale() object of various sizes are matched and location is returned. Using the location data, the face is cropped for further verification. Eye cascade is used to verify there are two eyes in the cropped face. If satisfied a marker is placed around the face to illustrate a face is detected in the location.

5.2 Face Recognition Process:

For this project we have used Linear Binary Pattern Histogram is used. This is implemented using OpenCV libraries.

There are three stages for the face recognition as follows:

1. Collecting images IDs

2. Extracting unique features, classifying them and storing in XML files

3. Matching features of an input image to the features in the saved XML files and predict identity

5.2.1 Collecting the image data:

Collecting classification images is usually done manually using a photo editing software to crop and resize photos. Furthermore, PCA and LDA requires the same number of pixels in all the images for the correct operation. This time consuming and a laborious task is automated through an application to collect 50 images with different expressions. The application detects suitable expressions between 300ms, straightens any existing tilt and save them. The Flow chart for the application is shown in figure 9.

Application starts with a request for a name to be entered to be stored with the ID in a text file. The face detection system starts the first half. However, before the capturing begins, the application check for the brightness levels and will capture only if the face is well illuminated. Furthermore, after the face is detected, the position of the eyes are analysed. If the head is tilted, the application automatically corrects the orientation. These two additions were made considering the requirements for Eigenface algorithm. The Image is then cropped and saved using the ID as a filename to be identified later. A loop runs this program until 50 viable images are collected from the person. This application made data collection efficient.

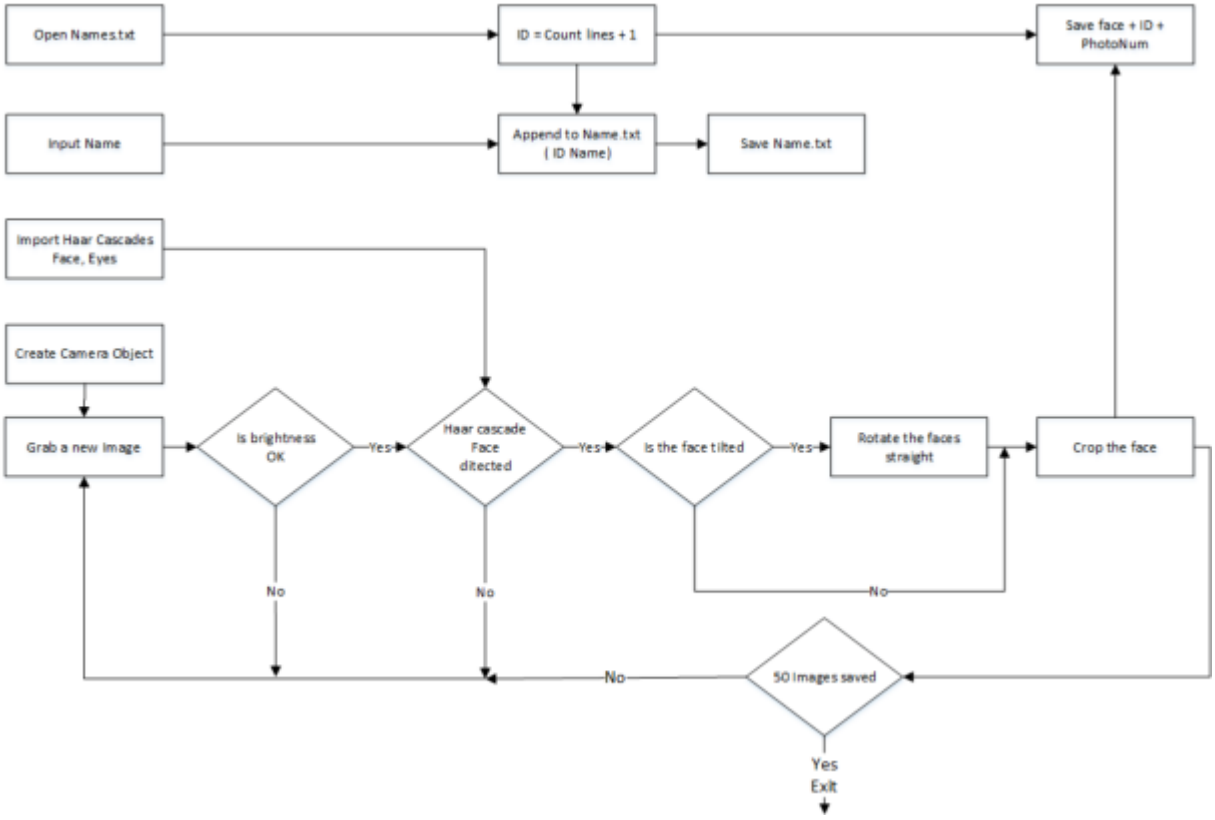


Figure 9: The Flowchart for the image collection

5.2.2 Training the classifiers:

OpenCV enables the creation of XML files to store features extracted from datasets using the FaceRecognizer class. The stored images are imported, converted to grayscale and saved with IDs in two lists with same indexes. FaceRecognizer objects are created using face recogniser class. Each recogniser can take in parameters that are described below:

cv2.face.createLBPHFaceRecognizer()

1. The radius from the centre pixel to build the local binary pattern.

2. The Number of sample points to build the pattern. Having a considerable number will slow down the computer.

3. The Number of Cells to be created in X axis.

4. The number of cells to be created in Y axis.

Recogniser objects are created and images are imported, resized, converted into numpy arrays and stored in a vector. The ID of the image is gathered from splitting the file name, and stored in another vector.By using FaceRecognizer.train(NumpyImage, ID) all three of the objects are trained.Next, the configuration model is saved as a XML file using Face recogniser.Save In this project, all three are trained and saved through one application for convenience.

5.2.3 The Face Recognition:

Face recogniser object is created using the desired parameters. Face detector is used to detect faces in the image, cropped and transferred to be recognised. This is done using the same technique used for the image capture application. For each face detected, a prediction is made using FaceRecognizer.predict() which return the ID of the class and confidence. The process is same for all algorithms and if the confidence his higher than the set threshold, ID is -1. Finally, names from the text file with IDs are used to display the name and confidence on the screen. If the ID is -1, the application will print unknown face without the confidence level. The flow chart for the application is shown in figure 11.

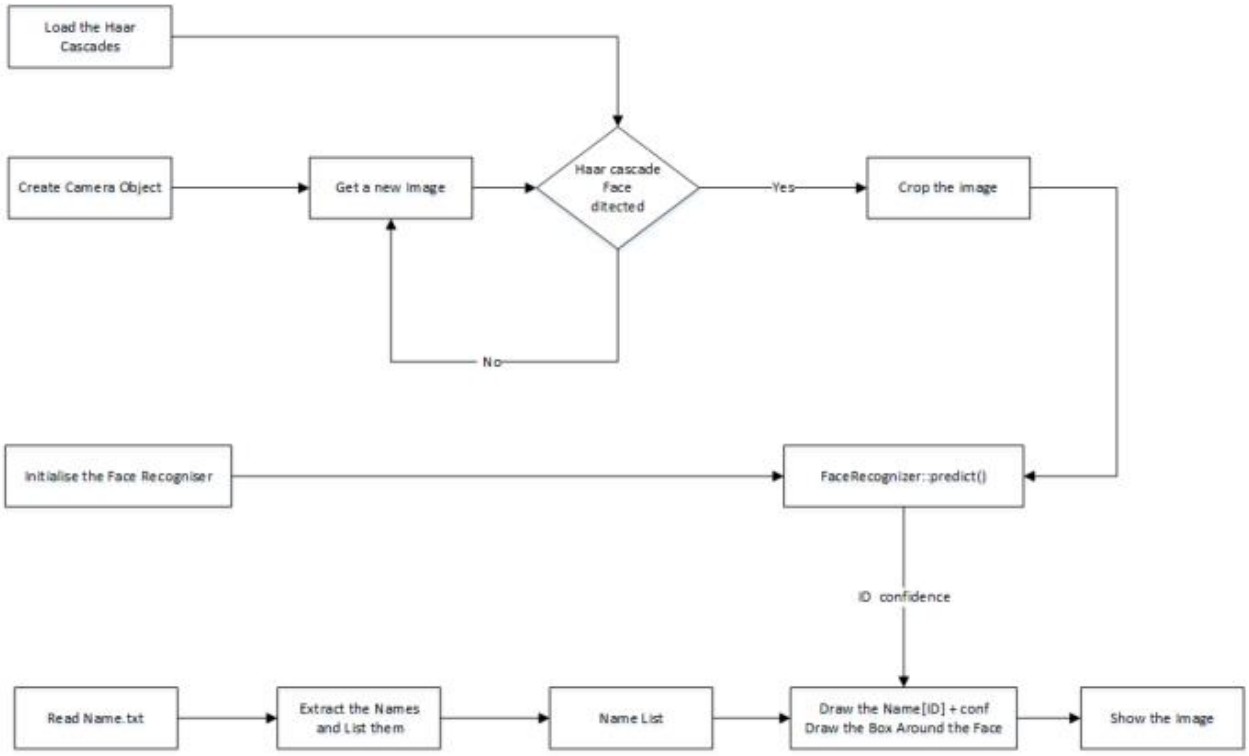


Figure 11: Flowchart of the face recognition application

CHAPTER 6

CODING

6.1 Dataset Create:

We are using 2 libraries in this project

1. Opencv

2. Numpy

```
import cv2
import numpy as np
```

To train the recognizer we have to find face in a frame so we are using Haarcascade frontalface_default.xml file which helps us to detect the face.

```
faceDetect=cv2.CascadeClassifier('harscade_frontalface_default.xml')
```

We have to initialize camera object and we are assigning each person with a specific id and we are storing 25 samples of each person to recognize from different angles.

```
faceDetect=cv2.CascadeClassifier()
cam=cv2.VideoCapture(0);
id=input('enter userId:')
sampleNum=0;
```

Now we have to take 25 sample and store them in folder with specific name containing user id and we are writing these in a loop. We are breaking loop when the sample images is 25.

```
While (True):
   ret,img=cam.read();
   gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
   faces=faceDetect. detectMultiScale(gray,1.3,5)
   for(x,y,w,h) in faces:
      sampleNum=sampleNum+1;
      cv2.imwrite("dataSet/User."+str(id)+"."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])
      cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
      cv2.waitKey(1);
   cv2.imshow("Face",img);
   cv2.waitKey(40);
   if(sampleNum>25 ):
     break
```

We have to release camera object and destroy all windows.

```
cam.release()
cv2.destroyAllWindows()
```

EntireCode:-

Dataset.py

import cv2

import numpy as np

faceDetect=cv2.CascadeClassifier('harscade_frontalface_default.xml')

```
faceDetect=cv2.CascadeClassifier()

cam=cv2.VideoCapture(0);

id=input('enter userId:')

sampleNum=0;

While (True):

ret,img = cam.read();

gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces = faceDetect. detectMultiScale(gray, 1.3,5 )

 for (x,y,w,h) in faces:

sampleNum = sampleNum+1;

cv2.imwrite("dataSet/User."+str(id)+"."+str(sampleNum)+".jpg"
,gray[y:y+h,x:x+w])

cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

cv2.waitKey(1);

cv2.imshow("Face",img);

cv2.waitKey( 40 );

if (sampleNum> 25 ):

break
```

cam.release()

cv2.destroyAllWindows()

6.2 Trainer:

We are using 4 libraries in this project

1.os(to read directories)

2.opencv

3.numpy

4.Pil.Image(read Images)

We are loading LBPHFaceRecognizer and harscade frontal face model these are helpful to detect faces.

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector=
cv2.CascadeClassifier("harscade_frontalface_default.xml");
path="dataSet"
```

Now we are creating a functions which reads dataSet folder and returns face faces

Faces is a array which contains the pixel values of face with and Ids is a array containing Ids as arrays.

```
def defImagesWithId(path):
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    faces=[]
    IDs=[]
    for imagePath in imagePaths:
        faceImg=Image.open(imagePath).convert('L');
        faceNp=np.array(faceImg,'uint8')
        ID=int(os.path.split(imagePath)[-1].split('.')[1])
        faces.append(faceNp)
        IDs.append(ID)
        cv2.imshow("training",faceNp)
        cv2.waitKey(10)
    return IDs,faces
```

Now by using this function we get ids and their faces

We are training this ids, faces by LBPH Face recognizer and saving in 'trainer.yml' file.

```
Ids,faces=getImagesWithID(path)
recognizer.train(faces,np.array(Ids))
recognizer.save('recognizer/trainingData.yml')
cv2.destroyAllWindows()
```

Entire Code:-

trainer.py

import os

import cv2

```
import numpy as np
from PIL import Image
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector= cv2.CascadeClassifier(" harscade_frontalface_default.xml ");
path=" dataSet "
def defImagesWithId (path):
imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
faces=[]
IDs=[]
for imagePath in imagePaths:
faceImg=Image.open(imagePath).convert('L');
faceNp=np.array(faceImg,'uint8')
ID=int(os.path.split(imagePath)[-1].split('.')[1])
faces.append(faceNp)
IDs.append(ID)
cv2.imshow("training",faceNp)
cv2.waitKey(10)
 return IDs,faces
Ids,faces=getImagesWithID(path)
recognizer.train(faces,np.array(Ids))
recognizer.save('recognizer/trainingData.yml')
cv2.destroyAllWindows()
```

6.3 Detector:

We are using 3 libraries in this project

1.time

2.opencv

3.numpy

```
import cv2
import numpy as np
import time
```

We are loading LBPHFaceRecognizer which we saved earlier, and harscade frontal face model these are helpful to detect faces

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("recognizer/trainingData.yml")
detector= cv2.CascadeClassifier("harscade_frontalface_default.xml")
```

We are going to run this in live video streaming so we have to initialize the camera object.

```
cap=cv2.VideoCapture(0)
```

We have to take each frame and do the process for Person recognition for infinite time so we are using infinite loop.

```
While True:
```

We are going to read each frame from camera object and converted it into gray color

And detecting faces in a frame by using haar cascade frontal face model.

```
ret,img=cam.read()
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces=faceDetect.detectMultiScale(gray,1.3,5);
```

Faces contains the coordinates of the faces in a frame. Now we are drawing a rectangle over that face and predicting the face by LBPH face recogniser and showing the respective name for the matching id.

```
for(x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
    id,conf=recognizer.predict(gray[y:y+h,x:x+w])
    if(conf<50):
      if(id==1):
        id="JVNR"
      elif(id==2):
        id="Karthik"
      elif(id==3):
        id="Phani"
    else:
      id="Unknown"
    cv2.putText(img,str(id),(x,y+h),font,0.55,(0,255,0),1)
  cv2.imshow("Face",img);
  end=time.time()
  print("took "+str((end - start)*1000)+"mil.sec")
  if(cv2.waitKey(1)==ord('q')):
    break;
```

We have to release camera object and destroyallwindows.

```
cam.release()
cv2.destroyAllWindows()
```

Entire Code:

Recogniser.py

```
import  cv2
import  numpy  as  np
import time
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read(" recognizer/trainingData.yml ")
detector= cv2.CascadeClassifier(" harscade_frontalface_default.xml ")
cap=cv2.VideoCapture(0)
While True:
ret,img=cam.read()
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces=faceDetect.detectMultiScale(gray,1.3,5);
for(x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
id,conf=recognizer.predict(gray[y:y+h,x:x+w])
print(conf)
if(conf<50):
if(id==1):
id="JVNR"
elif(id==2):
id="Karthik"
elif(id==3):
id="Phani"
else:
id="Unknown"
cv2.putText(img,str(id),(x,y+h),font,0.55,(0,255,0),1)
```

```
cv2.imshow("Face",img);
if(cv2.waitKey(1)==ord('q')):
Break;
cam.release()
cv2.destroyAllWindows()
```

CHAPTER 7

CONCLUSION

This project for visual perception and autonomy module. Next, it explains the technologies used in the project and the methodology used. Finally, it shows the results, discuss the challenges and how they were resolved followed by a discussion. Using Haar-cascades for face detection worked extremely well even when subjects wore spectacles. Real time video speed was satisfactory as well devoid of noticeable frame lag. Considering all factors, LBPH combined with Haar-cascades can be implemented as a cost effective face recognition platform. An example is a system to identify known troublemakers in a mall or a supermarket to provide the owner a warning to keep him alert or for automatic attendance taking in a class.

CHAPTER 8

REFERENCES

[1] Takeo Kanade. Computer recognition of human faces, volume 47. Birkhäuser Basel, 1977.

[2] Dong chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. IEEE Transactions on Geoscience and Remote Sensing, 28(4):509–512, Jul 1990.

[3] Meng Xiao He and Helen Yang. Microarray dimension reduction, 2009.

[4] John P Lewis. Fast template matching. In Vision interface, volume 95, pages 15–19, 1995.

[5] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In Image Processing. 2002. Proceedings. 2002 International Conference on, volume 1, pages I–I. IEEE, 2002.

[6] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proceed-ings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, pages I–511–I–518 vol.1, 2001.

[7] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In 2009 IEEE 12th International Conference on Computer Vision, pages 32–39, Sept 2009.