Iris Recognition Biometric System using Back-propagation Neural Network

A Project

Submitted in the partial fulfilment of the requirements for the award of
BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

Under the guidance of:

Mr. S. K. Chatterjee
Assistant Professor
BIT Mesra

Date of Submission: 21 November 2017

Submitted by

Md. Mobashshir Jawaid (BE-CSE/10027/2014)

Ankit Kr. Choudhary (BE-CSE/10417/2014)

Tanmay Joshi (BE-CSE/10446/2014)

Department of Computer Science and Engineering
Birla Institute of Technology, Mesra

DECLARATION CERTIFICATE

This is to certify that the work presented in the project report entitled "**Iris Recognition Biometric System using Back-propagation Neural Network"** in partial fulfilment of the requirement of the award of Degree of Bachelor of Engineering in Computer Science and Engineering of Birla Institute of Technology, Mesra, Ranchi is an authentic work carried out under my supervision and guidance.

To the best of my knowledge, the content of this report does not form a basis for the award of any previous Degree to anyone else.

Head	Mr. S. K. Chatterjee
Dept. of CSE	Dept. of CSE
Birla Institute of Technology	Birla Institute of Technology
Mesra, Ranchi	Mesra, Ranchi

CERTIFICATE OF APPROVAL

The foregoing report entitled "**Iris Recognition Biometric System using Back-propagation Neural Network"** is hereby approved as a creditable study of research topic and has been presented in satisfactory manner to warrant its acceptance as prerequisite to the degree for which it has been submitted.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed therein, but approve the report for the purpose for which it is submitted.

(Internal Examiner) **(External Examiner)**

(Chairman)

Head of the Department

Abstract

A biometric system provides automatic identification of an individual based on a unique feature or characteristic possessed by the individual. Iris recognition is regarded as the most reliable and accurate biometric identification system available. Most commercial iris recognition systems use patented algorithms developed by Mr. Daugman, and these algorithms are able to produce perfect recognition rates.

The work presented involved developing an ‘open-source’ iris recognition system in order to verify both the uniqueness of the human iris and also its performance as a biometric. For determining the recognition performance of the system a database of digitised greyscale eye images was used. The iris recognition system consists of an automatic segmentation system that is based on the Hough transform, and is able to localise the circular iris and pupil region. The extracted iris region was then normalised into a rectangular block with constant dimensions to account for imaging inconsistencies.

Located iris is extracted from an eye image, and, after normalization and enhancement, it is represented by a data set. Using this data set a Neural Network (NN) is used for the classification of iris patterns. The adaptive learning strategy is applied for training of the NN. The results of simulations illustrate the effectiveness of the neural system in personal identification.

Acknowledgements

We would like to express our sincere gratitude to our project supervisor and guide Mr. Sujit Kumar Chatterjee, Assistant Professor, BIT Mesra, who guided us, all through the project and showed us the right path. We are really gratified that he has spent so much time with us during our project work.

Our honourable thanks to Dr. Vandana Bhattacharjee, Professor and Head of Department of CSE, BIT Mesra for providing us access to excellent labs and library.

We thank BIT Mesra for providing us with the necessary infrastructure and technical support that was required for this project.

Finally, we express our gratitude and appreciation towards our parents and friends for their valuable suggestions and moral support.

Md Mobashshir Jawaid (BE/10027/2014)

Ankit Kr Choudhary (BE/10417/2014)

Tanmay Joshi (BE/10446/2014)

Contents

Abstract 3

Acknowledgements 4

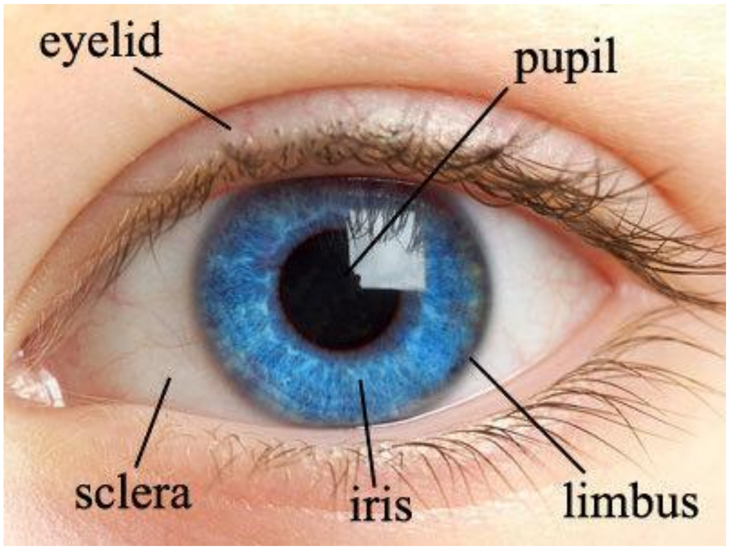
Contents 5

1. **Introduction** 6
    a. Subject Objective and Overview
2. **Segmentation** 8
    a. Hough Transform
    b. Implementation
    c. Example
3. **Normalization** 11
    a. Daugman's Rubber Sheet Model
    b. Implementation
    c. Example
4. **Neural Network based Classification** 14
    a. Network Architecture and Parameter learning
5. **Experimental Results** 16
    a. Dataset used
    b. Results
6. **Conclusion** 18
    a. Summary
    b. Challenges faced
    c. Future work
7. **References** 19
8. **Code** 20

1. Introduction

Biometrics technology plays important role in public security and information security domains. Using various physiological characteristics of human, such as face, facial thermograms, fingerprint, iris, retina, hand geometry etc., biometrics accurately identify each individual and distinguishes one from another. Iris recognition is one of important biometric recognition approach in a human identification is becoming very active topic in research and practical application.

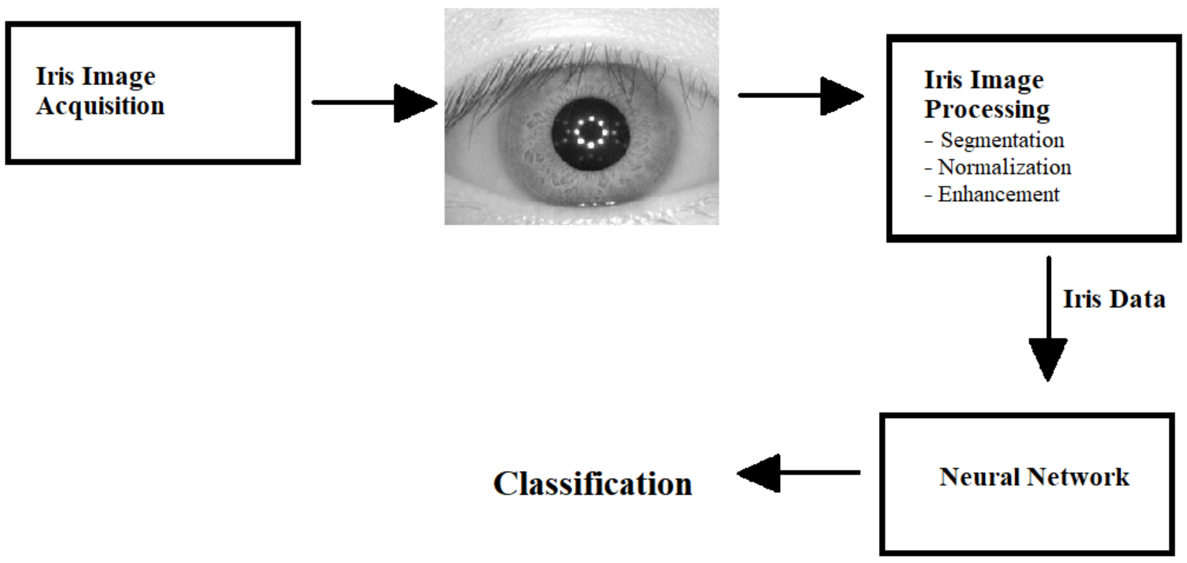
Iris region is the part between the pupil and the white sclera. This field is sometimes called iris texture. The iris texture provides many minute characteristics such as freckles, coronas, stripes, furrows, crypts, etc. These visible characteristics are unique for each subject. Such unique feature in the anatomical structure of the iris facilitates the differentiation among individuals. The human iris is not changeable and is stable. From one year of age until death, the patterns of the iris are relatively constant over a person's lifetime. Due to the epigenetic nature of iris patterns, the two eyes of an individual contain completely independent iris patterns, and identical twins possess uncorrelated iris patterns. Because of this uniqueness and stability, iris recognition is a reliable human identification technique.



a. System Objective and Overview

The objective will be to implement an open-source iris recognition system in order to verify the claimed performance of the technology. The development tool used will be Python and OpenCV library for image processing, and emphasis will be only on the software for performing recognition, and not hardware for capturing an eye image. To test the system, a data set of eye images is used as inputs; a database of greyscale eye images courtesy of The Chinese Academy of Sciences– Institute of Automation (CASIA).

The system is to be composed of a number of sub-systems, which correspond to each stage of iris recognition. These stages are segmentation – locating the iris region in an eye image, normalisation – creating a dimensionally consistent representation of the iris region. Located iris is extracted from an eye image, and, after normalization and enhancement, it is represented by a data set. Normalized image after enhancement is represented by the matrix that describes greyscale values of the iris image. This matrix becomes the training data set for the neural network. Backpropagation Neural Network (BPNN) is used for the classification of iris patterns. The iris recognition system includes two operation modes: training mode and online mode. In the first stage, the training of recognition system is carried out using greyscale values of iris images. Neural network is trained with all iris images. After training, in online mode, neural network performs classification and recognizes the patterns that belong to a certain person's iris.



2. Segmentation

The first stage of iris recognition is to isolate the actual iris region in a digital eye image. The iris region, can be approximated by two circles, one for the iris/sclera boundary and another, interior to the first, for the iris/pupil boundary. The eyelids and eyelashes normally occlude the upper and lower parts of the iris region. Also, specular reflections can occur within the iris region corrupting the iris pattern. A technique is required to isolate and exclude these artefacts as well as locating the circular iris region.

The success of segmentation depends on the imaging quality of eye images. Images in the CASIA iris database do not contain specular reflections due to the use of near infra-red light for illumination.

The segmentation stage is critical to the success of an iris recognition system, since data that is falsely represented as iris pattern data will corrupt the biometric templates generated, resulting in poor recognition rates.

a. Hough Transform

The Hough transform is a standard computer vision algorithm that can be used to determine the parameters of simple geometric objects, such as lines and circles, present in an image. The circular Hough transform can be employed to deduce the radius and centre coordinates of the pupil and iris regions.

Firstly, an edge map is generated by calculating the first derivatives of intensity values in an eye image and then thresholding the result. From the edge map, votes are cast in Hough space for the parameters of circles passing through each edge point.

These parameters are the centre coordinates x_c and y_c, and the radius r, which are able to define any circle according to the equation

$$(x-x_c)^2+(y-y_c)^2 - r^2=0$$

A maximum point in the Hough space will correspond to the radius and centre coordinates of the circle best defined by the edge points. There are a few problems with the Hough transform method. It requires threshold values to be chosen for edge detection, and this may result in critical edge points being removed, resulting in failure to detect circles/arcs if suitable threshold values are not chosen.

b. Implementation

We used circular Hough transform for detecting the iris and pupil boundaries. This involves first employing Canny edge detection to generate an edge map. Gradients were biased in the vertical direction for the outer iris/sclera boundary. A slightly modified Hough transform method was used, OpenCVs Hough Gradient Method which uses the gradient information of edges.

The range of radius values to search for was set manually, depending on the database used. For the CASIA database, values of the iris radius range from 90 to 150 pixels, while the pupil radius ranges from 28 to 75 pixels. After this process was complete, six parameters are stored, the radius, and x and y centre coordinates for both circles.

c. Example

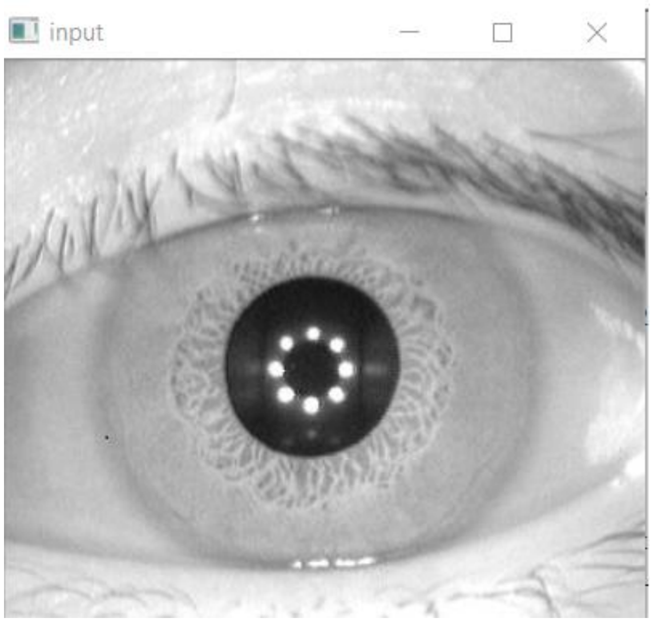


Fig 1. Before Segmentation

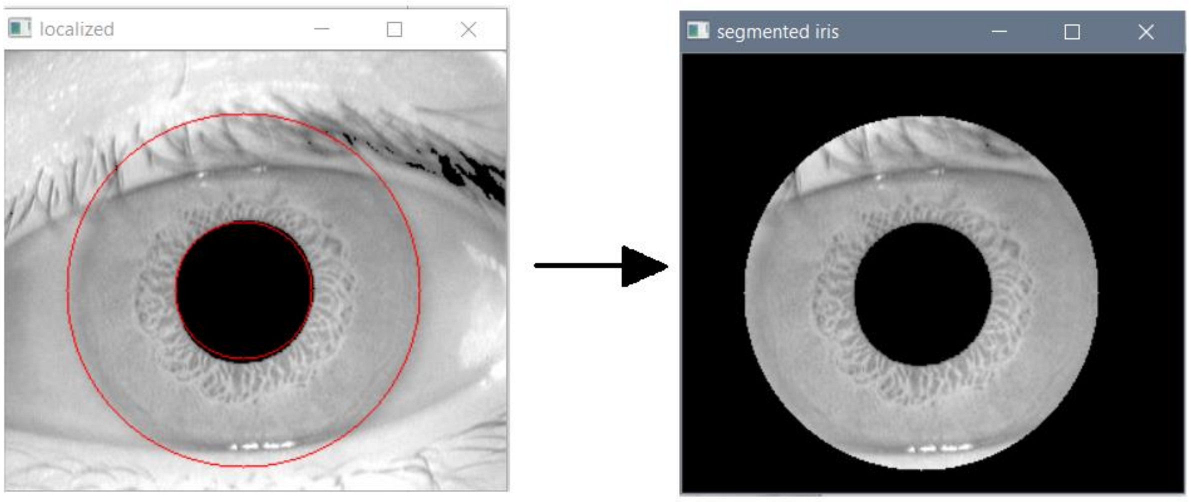


Fig 2. Detecting iris boundaries and Segmentation

3. Normalization

Once the iris region is successfully segmented from an eye image, the next stage is to transform the iris region so that it has fixed dimensions in order to allow comparisons. The dimensional inconsistencies between eye images are mainly due to the stretching of the iris caused by pupil dilation from varying levels of illumination. Also the irises captured from the different people have different sizes. These factors may affect the result of iris matching. In order to avoid these factors and achieve more accurate recognition, the normalization of iris images is implemented. In normalization, the iris circular region is transformed to a rectangular region with a fixed size. The iris region is normalized from Cartesian coordinates to polar representation. The normalisation process will produce iris regions, which have the same constant dimensions, so that two photographs of the same iris under different conditions will have characteristic features at the same spatial location.

a. Daugman's Rubber Sheet Model

The homogenous rubber sheet model devised by Daugman remaps each point within the iris region to a pair of polar coordinates (r,θ) where r is on the interval [0,1] and θ is angle [0,2π].

The remapping of the iris region from (x,y) Cartesian coordinates to the normalised non-concentric polar representation is modelled as

$$I(x(r,\theta), y(r,\theta)) \rightarrow I(r,\theta)$$

With $x(r,\theta) = (1-r)\,x_p(\theta) + r\,x_i(\theta)$, $y(r,\theta) = (1-r)\,y_p(\theta) + r\,y_i(\theta)$

where, $x_p(\theta) = x_{p0}(\theta) + r_p\cos(\theta)$, $y_p(\theta) = y_{p0}(\theta) + y_p\cos(\theta)$

$x_i(\theta) = x_{i0}(\theta) + r_i\cos(\theta)$, $y_i(\theta) = y_{i0}(\theta) + y_i\cos(\theta)$

Where I(x,y) is the iris region image, (x,y) are the original Cartesian coordinates, (r,θ) are the corresponding normalised polar coordinates, and $x_{p,}$ y_p and $x_{i,}$ y_i are the coordinates of the pupil and iris boundaries along the θ direction. The rubber sheet model takes into account pupil dilation and size inconsistencies in order to produce a normalised representation with constant dimensions. In this way the iris region is modelled as a flexible rubber sheet anchored at the iris boundary with the pupil centre as the reference point. The homogenous rubber sheet model accounts for pupil dilation, imaging distance and non-concentric pupil displacement.

b. Implementation

For normalisation of iris regions a technique based on Daugman's rubber sheet model was employed. The centre of the pupil was considered as the reference point, and radial vectors pass through the iris region. A constant number of points are chosen along each radial line, so that a constant number of radial data points are taken, irrespective of how narrow or wide the radius is at a particular angle. The normalised pattern was created by backtracking to find the Cartesian coordinates of data points from the radial and angular position in the normalised pattern. From the 'doughnut' iris region, normalisation produces a 2D array with horizontal dimensions of angular resolution and vertical dimensions of radial resolution.

c. Example

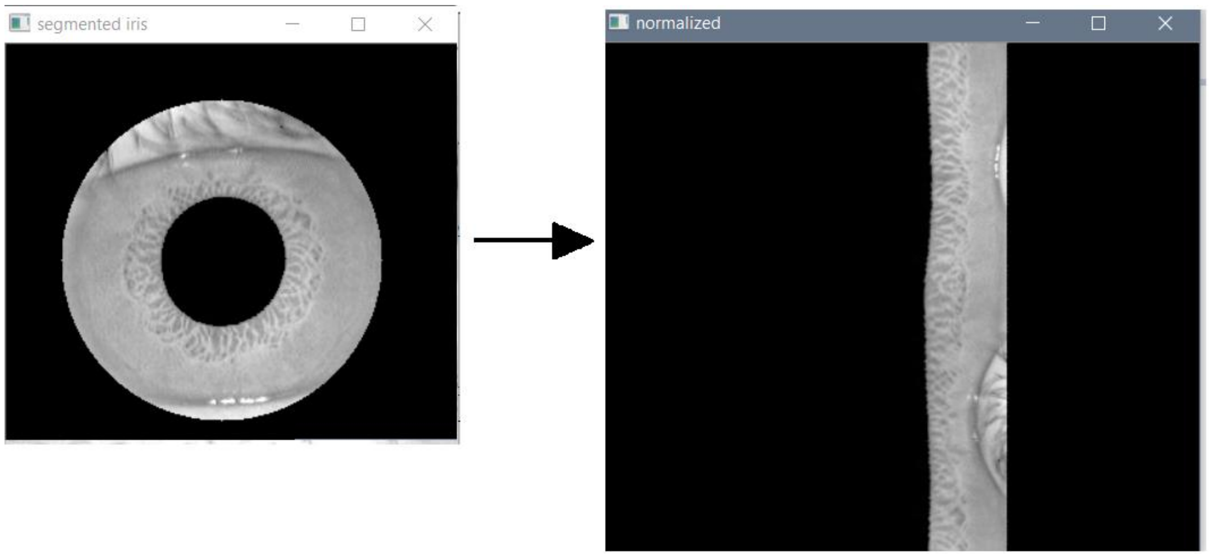


Fig 3. Normalization

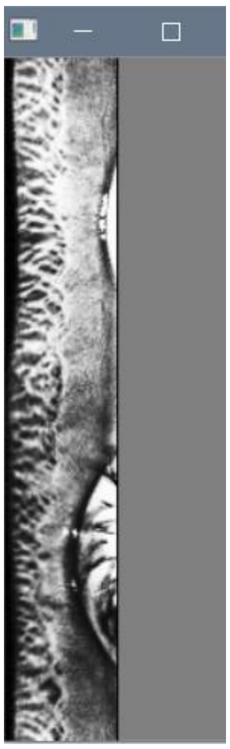


Fig 4. Normalized iris after enhancement

During enhancement, background illumination is subtracted from the normalized image to compensate for a variety of lighting conditions. Then the lighting corrected image is enhanced by histogram equalization. As a result, the texture characteristics of iris image are shown more clearly. Such pre-processing compensates for the non-uniform illumination and improves the contrast of the image.

4. Neural Network based Classification

In this project, a Backpropagation Neural Network (BPNN) is used to recognise the iris patterns. In this approach, the normalized and enhanced iris image is represented by a two-dimensional array. This array contains the greyscale values of the texture of the iris pattern. These values are input signals for the neural network. The iris recognition system includes two operation modes: training mode and online mode. In first stage, the training of recognition system is carried out using greyscale values of iris images. Neural network is trained with all iris images. After training, in online mode, neural network performs classification and recognizes the patterns that belong to a certain person's iris.

a. Network architecture and parameter learning

The Architecture of NN is given as follows. Two hidden layers are used in the NN. In this structure, x1, x2 ... xm are greyscale values of input array that characterizes the iris texture information, P1, P2 ... Pn are output patterns that characterize the irises.

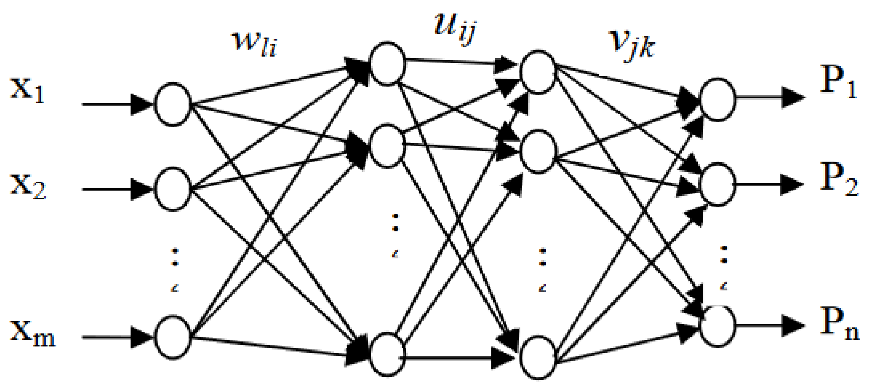


Fig5. Network Architecture

The k-th output of neural network is determined by the formula

$$P_k = f_k(\sum_{j=1}^{h2} v_{jk} \cdot f_j(\sum_{i=1}^{h1} u_{ij} \cdot f_i(\sum_{l=1}^{m} w_{li} x_l)))$$

where v_{jk} are weights between the output and second hidden layers of network, u_{ij} are weights between the hidden layers, w_{li} are weights between the input and first hidden layers, f is the activation function that is used in neurons, xl is input signal. Here k=1,..,n, j=1,..,h2, i=1,..,h1, l=1,..,m, m is number of neurons in input layer, n is number of neurons in output layer, h1 and h2 are number of neurons in first and second hidden layers, correspondingly.

For our NN the activation function used is the Sigmoid function and the number of neurons in the hidden layers are according to the formula proposed by Huang et al. (2003) which is

h1 = sqrt[(n+2)m]+2sqrt[m/(n+2)]   and   h2 = n*sqrt[m/(n+2)]

where h1 and h2 are number of neurons in first and second hidden layers, m is number of neurons in input layer, n is number of neurons in output layer. During training, the value of the cost function used is the mean squared error cost function. Gradient descent Back Propagation algorithm was used for parameter learning in the training of the neural network (BPNN).

5. Experimental Results

a. Dataset used

In order to evaluate the iris recognition algorithms, the CASIA iris image database is used. Currently this is largest iris database available in the public domain. The Chinese Academy of Sciences – Institute of Automation (CASIA) eye image database contains greyscale eye images with unique eyes or classes and different images of each unique eye. Images from each class are taken from two sessions with one month interval between sessions. The images were captured especially for iris recognition research using specialised digital optics developed by the National Laboratory of Pattern Recognition, China. The eye images are mainly from persons of Asian descent, whose eyes are characterised by irises that are densely pigmented, and with dark eyelashes. Due to specialised imaging conditions using near infra-red light, features in the iris region are highly visible and there is good contrast between pupil, iris and sclera regions.

b. Results

Experiments are performed in two stages: iris segmentation and iris recognition. At first stage the above described rectangular area algorithm is applied for the localization of irises. The detected irises after normalization and enhancement are scaled by using averaging. This helps to reduce the size of neural network. Then the images are represented by matrices. These matrices are the input signal for the neural network. The outputs of the neural network are classes of iris patterns. The normalized iris image was 60x180 pixels in resolution. The greyscale values were used as input vector to the NN. Thus the number of input neurons was 10,800 (60*180). In second stage the iris pattern classification using NN is performed. 20 person's irises are selected from iris database for classification, with each person having 10 eye images. Thus the output layer has 20 neurons. Two hidden layers are used in neural network. The numbers of neurons in first and second hidden layers are 530 and 440, correspondingly. Each class characterizes the certain person's iris. Backpropagation Neural

Network learning algorithm is applied in order to solve iris classification. From each set of iris images, 7 images were used for training and 3 for testing. Thus a total of 140 images were used for training of the NN and 60 images used for testing. After training the remaining images are used for testing. The recognition rate of NN system was 96.67%. The obtained recognition result is compared with the recognition results of other methods that utilize the same iris database. The results of this comparison are given in the table. As shown in the table, the identification result obtained using the neural network approach illustrates the success of its efficient use in iris recognition.

Methodology	**Accuracy**
Daugman	99.90%
Boles	92.64%
Li Ma	94.90%
Avila	97.89%
Neural Network	96.67%

6. Conclusion

a. Summary

This project has presented an iris recognition system, which was tested using a database of greyscale eye images in order to verify the claimed performance of iris recognition technology. 20 person's irises were selected from iris database for classification with each person having 10 eye images. The recognition rate of NN system was 96.67%. Through the gathered accuracy percentage from the set of trials, it can be concluded that the implemented iris recognition system is reliable.

b. Challenges faced

- Segmentation is the most critical stage of iris recognition, since areas that are wrongly identified as iris regions will corrupt biometric templates resulting in very poor recognition. The threshold values needed for detecting iris region boundaries had to be chosen through trial and can change depending on database and image acquisition method used. Although if the same acquisition method is used throughout the system then this is only needed to be set once for the whole database.
- Noise in images caused due to eyelashes caused slight decrease in accuracy of the system. Although our algorithm minimized the error produced due to such noise, even more advanced methods to handle noise due to eyelashes can be used to further improve accuracy of the system.

c. Future work

- An authentication system to unlock a user's phone or computer can be implemented using the recognition system proposed in this project.
- Another extension to the system would be to interface it to an iris acquisition camera. Now rather than having a fixed set of iris images from a database, a frame grabber can be used to capture a number of images, possibibly improving the recognition rate.
- In order to improve the automatic segmentation algorithm, a more elaborate eyelid and eyelash detection system could be implemented.

7. References

- J. Daugman. How iris recognition works. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, No. 1.
- Libor Masek. Recognition of Human Iris Patterns for Biometric Identification. School of Computer Science and Soft Engineering, The University of Western Australia.
- Rahib H.Abiyev, Koray Altunkaya. Personal Iris Recognition Using Neural Network. International Journal of Security and its Applications Vol. 2, No. 2, April, 2014.
- Guang-Bin Huang. Learning Capability and Storage Capacity of Two-Hidden-Layer Feedforward Networks. IEEE Transactions on Neural Networks, vol. 14, no. 2.
- S.Mahajan et al. An LBP based Iris Recognition System using Feed Forward Backpropagation Neural Network. International Journal on Recent and Innovation Trends in Computing and Communication Vol. 5, No. 5, 2016.
- Bhawna Chouhan et al. Iris Recognition System using canny edge detection for Biometric Identification. International Journal of Engineering Science and Technology (IJEST), 2016.
- J. Daugman. New methods in iris recognition. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 37(5):1167-1175, 2007.
- W. Boles and B. Boashash, “A Human Identification Technique Using Images of the Iris and Wavelet Transform,” IEEE Trans. Signal Processing, vol. 46, no. 4, pp.1185-1188.
- L.Ma, Y.H.Wang, T.N.Tan, “Iris recognition based on multichannel Gabor filtering”. Proceedings of the Fifth Asian Conference on Computer Vision, Australia, pp.279-283.

8. Code

Iris_imgprocess.py

```
import cv
import cv2
import math
import numpy as np
import os

centroid = (0,0)
radius = 0
currentEye = 0
eyesList = []
# Returns a different image filename on each call. If there are
#no more elements in the list of images, the function return -1
def getNewEye(list):
    global currentEye
    newEye=list[currentEye]
    currentEye+=1
    if(currentEye>=len(list)):
        currentEye=-1
    return (newEye)

# Returns the cropped image with the isolated iris and black-
#painted pupil. It uses the getCircles function in order to look
#for the best value for each image (eye) and then obtaining the
#iris radius in order to create the mask and crop.
def getIris(frame):
    global radius
    iris = []
    copyImg = cv.CloneImage(frame)
    resImg = cv.CloneImage(frame)
    cv.Circle(resImg,centroid,int(radius),cv.CV_RGB(255,0,0),1)
    grayImg = cv.CreateImage(cv.GetSize(frame), 8, 1)
    mask = cv.CreateImage(cv.GetSize(frame), 8, 1)
    storage = cv.CreateMat(frame.width, 1, cv.CV_32FC3)
    cv.CvtColor(frame,grayImg,cv.CV_BGR2GRAY)
    cv.Canny(grayImg, grayImg, 5, 70, 3)
    cv.Smooth(grayImg,grayImg,cv.CV_GAUSSIAN, 7, 7)
    circles = getCircles(grayImg)
    iris.append(resImg)
    circle=circles[0]
    rad=circle[0][2]
    radius = rad
    cv.Circle(resImg,centroid,rad,cv.CV_RGB(255,0,0),1)
    return (resImg)
```

```
def segmentIris(img):
    h,w,d=img.shape
    circle_img=np.zeros((h,w),np.uint8)
    cv2.circle(circle_img,centroid,radius,255,thickness=-1)
    segmented_iris = cv2.bitwise_and(img,img,mask=circle_img)
    return segmented_iris


# Search middle to big circles using the Hough Transform function
# and loop for testing values in the range [80,150]. When a
#circle is found, it returns a list with the circles' data
#structure. Otherwise, returns an empty list.

def getCircles(image):
    i = radius
    while i < 151:
        storage = cv.CreateMat(image.width, 1, cv.CV_32FC3)
        cv.HoughCircles(image, storage, cv.CV_HOUGH_GRADIENT, 2,
100.0, 30, i, 100, 140)
        circles = np.asarray(storage)
        if (len(circles) == 1):
            return circles
        i +=1
    return ([])

# Returns the same images with the pupil masked black and set the
# global variable centroid according to calculations. It uses the
# FindContours  function for finding the pupil, given a range of
# black tones.
def getPupil(frame):
    pupilImg = cv.CreateImage(cv.GetSize(frame), 8, 1)
    cv.InRangeS(frame, (30,30,30), (100,100,100), pupilImg)
    contours = cv.FindContours(pupilImg, cv.CreateMemStorage(0),
mode = cv.CV_RETR_EXTERNAL)
    del pupilImg
    pupilImg = cv.CloneImage(frame)
    while contours:
        moments = cv.Moments(contours)
        area = cv.GetCentralMoment(moments,0,0)
        if (area > 50):
            pupilArea = area
            x = cv.GetSpatialMoment(moments,1,0)/area
            y = cv.GetSpatialMoment(moments,0,1)/area
            pupil = contours
            global centroid
            centroid = (int(x),int(y))
            global radius
            radius = math.sqrt(area/3.141)
            cv.DrawContours(pupilImg, pupil, (0,0,0), (0,0,0), 2,
cv.CV_FILLED)
            break
        contours = contours.h_next()
    return (pupilImg)

# Returns the image as a "tape" converting polar coord. to
#Cartesian coord.
```

```
def getPolar2CartImg(image, rad):
    imgSize = cv.GetSize(image)
    c=centroid
    rad=int(min(float(imgSize[0]/2.0), float(imgSize[1]/2.0)))
    imgRes = cv.CreateImage((rad*3, int(360)), 8, 3)
    cv.LogPolar(image,imgRes,c,60.0,
cv.CV_INTER_LINEAR+cv.CV_WARP_FILL_OUTLIERS)
    return (imgRes)

def autocrop(image, threshold=0):
    if len(image.shape) == 3:
        flatImage = np.max(image, 2)
    else:
        flatImage = image
    assert len(flatImage.shape) == 2
    rows = np.where(np.max(flatImage, 0) > threshold)[0]
    if rows.size:
        cols = np.where(np.max(flatImage, 1) > threshold)[0]
        image = image[cols[0]: cols[-1] + 1, rows[0]: rows[-1] +
1]
    else:
        image = image[:1, :1]
    return image

def processImg(path,savePath):
    frame = cv.LoadImage(path)
    iris = cv.CloneImage(frame)
    output = getPupil(frame)
    radius_pupil=radius
    iris_localized = getIris(output)
    cv.SaveImage(savePath,output)
    iris2=cv2.imread(savePath)
    segmented_iris = segmentIris(iris2)
    cv2.imwrite(savePath,segmented_iris)
    iris=cv.LoadImage(savePath)
    normImg = getPolar2CartImg(iris,radius)
    cv.SaveImage(savePath,normImg)

    img=cv2.imread(savePath)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    crop = autocrop(gray,30)
    crop=cv2.equalizeHist(crop)
    final=cv2.resize(crop,(60,180))
    cv2.imwrite(savePath,final)

    cv.ShowImage("input", frame)
    cv.ShowImage("localized", iris_localized)
    cv2.imshow("segmented iris",segmented_iris)
    cv.ShowImage("normalized", normImg)
    cv2.imshow("cropped",crop)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```
ip = "S1003L01.jpg"
processImg(ip,"temp/"+ip)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Iris_recognition.py

```
from PIL import Image
import numpy as np
import os
import matplotlib.pyplot as plt
#import requests
np.seterr(all = 'ignore')
class Neural_Net(object):
    def __init__(self):
        self.epochs=10
        self.instances=140
        self.input_layer_size=10800;
        self.output_layer_size=20;
        self.hidden_layer1_size=530;
        self.hidden_layer2_size=440;
        self.alpha=0.001
        self.w1=np.random.randn(self.input_layer_size,
self.hidden_layer1_size)
        self.w2=np.random.randn(self.hidden_layer1_size,
self.hidden_layer2_size)
        self.w3=np.random.randn(self.hidden_layer2_size,
self.output_layer_size)
        self.bias1=np.random.randn(1,self.hidden_layer1_size)
        self.bias2=np.random.randn(1,self.hidden_layer2_size)
        self.bias3=np.random.randn(1,self.output_layer_size)


    def forward(self,X):
        self.z1=np.dot(X,self.w1)+self.bias1
        self.a1=self.sigmoid(self.z1)
        self.z2=np.dot(self.a1,self.w2)+self.bias2
        self.a2=self.sigmoid(self.z2)
        self.z3=np.dot(self.a2,self.w3)+self.bias3
        y_predicted=self.softmax_func(self.z3)
        return y_predicted

    def sigmoid(self,z):
           return 1/(1+np.exp(-z))
    def sigmoid_prime(self,z):
        return self.sigmoid(z)*(1-self.sigmoid(z));

    def train(self,X,y):
        cost_plot=np.zeros([self.epochs])
        for epoch in range(0,self.epochs):
            print("epoch: "+ "  "+str(epoch))
            mse=0.0
            for i in range(0,self.instances):
                x1=np.zeros([1,self.input_layer_size])
                y1=np.zeros([1,self.output_layer_size])
                for j in range(0,self.input_layer_size):
                    x1[0][j]=X[i][j]
                for j in range(0,self.output_layer_size):
                    y1[0][j]=y[i][j]
                self.y_hat=self.forward(x1)
                cost=self.cost_func(y1)
                mse=mse+cost
                delta4=-(y1-self.y_hat)

delta3=np.dot(delta4,self.w3.T)*self.sigmoid_prime(self.a2)

delta2=np.dot(delta3,self.w2.T)*self.sigmoid_prime(self.a1)
                dw3=np.dot(self.a2.T,delta4)
                dw2=np.dot(self.a1.T,delta3)
                dw1=np.dot(x1.T,delta2)
                db3=delta4
                db2=delta3
                db1=delta2
                self.w3=self.w3-self.alpha*dw3
                self.w2=self.w2-self.alpha*dw2
                self.w1=self.w1-self.alpha*dw1
                self.bias1=self.bias1-((self.alpha)*(db1))
                self.bias2=self.bias2-((self.alpha)*(db2))
                self.bias3=self.bias3-((self.alpha)*(db3))
            mse=mse/self.instances
            print mse
            cost_plot[epoch]=mse
            if mse<=0.001:
                break
        return cost_plot

    def cost_func(self,y):
         y_tmp=(y-self.y_hat)**2
         return np.sum(y_tmp)/2

    def softmax_func(self,zz):
        """Compute softmax values for each sets of scores in zz."""
        e_x = np.exp(zz - np.max(zz))
        return e_x / e_x.sum()

x_values=np.zeros([200,10800])
y_values=np.zeros([200,1])
base_path="final/"
#this is converting each image into grayscale values from a folder
where the images are serially numbered
for i in range(1,21):
    eyeList=os.listdir(base_path+str(i)+"/")
    for j in range(0,10):
        #c=chr(ord('a')+j)
        full_path=base_path+str(i)+"/"+str(eyeList[j])
        img=Image.open(full_path).convert('L')
        new_img=img.resize((60,180))
        x_values[(i-1)*10+j]=np.array(new_img).reshape((1,10800))
        y_values[(i-1)*10+j][0]=i-1;
#this is converting the given output values into an output matrix
that could be used for
#neural net  i.e. m*1 type
y_values1=np.zeros([200,20])
for i in range(0,200):
    xxx=y_values[i][0]
    for j in range(0,20):
        if xxx==j:
            y_values1[i][j]=1
y_values=y_values1
x_values1=x_values.T
#this is normalisation of data by taking mean and standard deviation
mean1=np.zeros([1,10800])
std1=np.zeros([1,10800])
for i in range(0,10800):
    mean1[0][i]=np.mean(x_values1[i])
    std1[0][i]=np.std(x_values1[i])
    x_values1[i,:]-=mean1[0][i]
    x_values1[i,:]/=std1[0][i]
x_values=x_values1.T
x_train=np.zeros([140,10800]);
x_test=np.zeros([60,10800]);
y_test=np.zeros([60,20]);
y_train=np.zeros([140,20])
itrain=0;
itest=0;
#dividing my dataset into training and testing datasets and
#corresponding output values also
for i in range(0,20):
    for j in range(0,7):
        x_train[itrain]=x_values[10*i+j];
        y_train[itrain]=y_values[10*i+j];
        itrain+=1;
    for j in range(7,10):
        x_test[itest] =x_values[10*i+j];
        y_test[itest]=y_values[10*i+j];
        itest+=1;

#creating object of the class
obj=Neural_Net()
#cost_values will store the costfunction values for each iteration
cost_values=obj.train(x_train,y_train)
#plot the cost_values values
iterat=np.arange(obj.epochs)*obj.instances;
plt.plot(iterat,cost_values)
plt.show()
#counting total postives and hence predicting accuracy
tp=0;
for i in range(0,60):
    xx=x_test[i];
    xx=np.array([xx])
    yy=y_test[i]
    yy=np.array([yy])
    ypredict=obj.forward(xx)
    ypredict11=ypredict.argmax(axis=1)
    if( yy[0][ypredict11]==1 ):
        tp+=1
print("tp here is ",tp)
acc=tp/60.0
print("accuracy is:",acc)

xinput=np.zeros([1,10800])
iter=0
while iter<10:
    ip=raw_input("Enter image name: ")
    full_path="temp/"+str(ip)
    img=Image.open(full_path).convert('L')
    new_img=img.resize((60,180))
    xinput[0]=np.array(new_img).reshape((1,10800))
    #this is normalisation of data by taking mean and standard
deviation
    xinput1=xinput.T
    for i in range(0,10800):
        xinput1[i,:]-=mean1[0][i]
        xinput1[i,:]/=std1[0][i]
    xinput=xinput1.T
```

```
#END normalization
xt=xinput[0]
xt=np.array([xt])
yp=obj.forward(xt)
#print yp
yp11=yp.argmax(axis=1)
print "Recognized as: Person #"+str( yp11[0]+1)
iter=iter+1
```