Iris Recognition System

Real Time System Project
for
Masters in High Integrity Systems

Group 13

Chinmaygopal Voolapalli
Matrikel No. 1272455

Mani Shankar Rajagopalan
Matrikel No. 1273894

Shashi Kamal Jaiswal
Matrikel No. 1273250

March 25, 2019

Abstract

A Biometric System allows the identification of a particular person based on their unique identifying feature. Iris detection is considered as the most dependable and accurate biometric system present. Most commercial iris detection systems use patented algorithms developed by Daugman, and these algorithms are able to produce perfect recognition rates. (Masek et al., 2003) The basic algorithm adopted in this paper required us to work with a certain database of images. We employ the steps of segmentation, normalisation and feature encoding and matching sequentially in order to successfully detect and validate the iris in our image. All the said steps of the algorithm were created using the image processing functions present in OpenCV library in C++. Due to the time constraints of the project, we were not able to realize the Feature Encoding and Matching section of the algorithm and instead utilized the Face Recognizer tool present in OpenCV. The entire application is a simulation-based application that has been developed to support greyscale images for now. We will be elaborating on the Intent/Scenario of the project, and Steps involved in the algorithm in separate chapters in the report, in order to provide a modular overview of our application.

Contents

List of Figures

1 Introduction

1.1 Biometric Technology

Biometric technology deals with recognizing the identity of individuals based on their unique physical or behavioral characteristics (Wildes, 1997). These characteristics, also referred to as biometric identifiers, include hand geometry, fingerprints, facial recognition, voice and many more. The most popular application of this technology is in the field of security. They are widely used for user authentication and authorization.

Generally the following steps are carried out in biometric systems:

1. A sample of the feature from the user is captured.
2. The sample is is then formatted via some mathematical function to obtain a biometric template. The template exposes the most discriminating properties of the sample.
3. The template is compared with other templates to validate the user

There are usually two parts to such a system:

1. User registration
   The user and a sample of his biometric identifier(post template generation) are enrolled into a user database.
2. Pattern Recognition
   The sample procured at real time is then matched with the templates in the user database to identify the user.

With a increasing number of countries adopting bio-metrically enabled identity (ID) card schemes and a large number of consumer appliances such as mobiles and laptops, also progressively depending on this technology, the importance for highly distinct, scalable and permanent biometric identifiers is considerable. The human iris is the best identifier in this scenario.

1.2 Iris Recognition

The iris is a thin circular diaphragm, which lies between the cornea and the lens of the human eye.

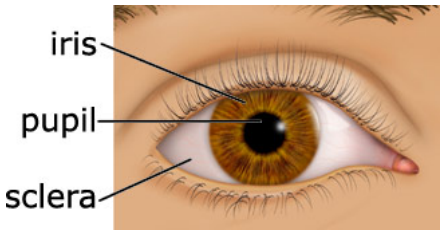

Figure 1.1: The Human Eye
(apsubiology.org, 2015)

As seen in Figure 1, the iris region is bounded between the pupil and sclera (more precisely, the limbus). The pupil is the black circular opening that controls the amount of light entering the eye by varying its size. the iris assists in controlling the diameter of the pupil, making the vision sharper.

Iris patterns are unique, stable and cannot be modified surgically. Iris recognition applies different pattern matching algorithms to the template obtained from the user iris (the biometric identifier). The transformations involved are segmentation - isolating the iris region in an image of the eye, normalization - development of a dimensionally appropriate iris image, and feature encoding and matching - template creation consisting of the highly unique attributes of the iris template matching to validate the user. These steps are essentially same as those for any biometric system, discussed in the previous section. The Iris Detection was first implemented in 1990 based on the algorithms provided by Dr. John Daugman. The techniques will be discussed in detail in the successive sections.

1.3 Open Source Computer Vision Library (OpenCV)

"OpenCV is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms" (opencv.org, 2019b) The functions and classes that we have utilized from OpenCV have also been described in detail in the forthcoming sections. The versatility and inbuilt packages encompassing the various required functionalities to perform machine vision related application development, makes OpenCV an extremely valuable programming library for the field of image processing. The presence of specific functions defined for filtering, contouring and for other tasks required in the application, enhances the overall productivity.

1.4 Intent/Scenario of the Project

The project is a simulation based project and we have performed the test and analysis for a set of NIR(Near-Infra-Red) greyscale images obtained from the CASIA(Chines Academy of Sciences Institute of Automation) database. We have assumed that the system is working in a scenario, in which a user would use his authentication ID, such as ID card, unique login data, and others, in order to prepare the system to compare the input data against only the user's data from the database. As this is a simulation based project, we are working with only a certain type of images, and the algorithm is tuned to detect only these type of images. We have used OpenCV library and created the application entirely in C++.

2 State of the Art

After having analyzed some papers on this particular topic, we were able to realize that the basic steps of Iris detection were: Segmentation, Normalization and Feature Encoding. There are a multitude of algorithms present for each step of the detection, but some of them are frequently preferred in tandem. In order to reach this conclusion, we had analyzed the papers: (Masek et al., 2003), (Daugman, 2009), (Mohammadi Arvacheh, 2006). Listed below are few of the approaches we found after surveying these papers:

Segmentation: Daugman's Integro-differential Operator, Hough Transform, Discrete Circular Active Contours, Discrete Circular Active Contour

Normalization: Daugman's Cartesian to Polar Transform, Wildes Image Registration, Non-linear Normalization Model

Feature Encoding and Matching: Gabor Filters, Log-Gabor Filters, Zero-crossings of the 1D wavelet, Haar Wavelet, Laplacian of Gaussian Filters, Hamming distance, Weighted Euclidean Distance, Normalised Correlation

From these, we selected and explored the Hough Transform approach for Segmentation, and Daugman's Cartesian to Polar Transform for Normalization. For the Feature Encoding and Matching phase we followed a different approach form the standard algorithms. We used the Local Binary Pattern Histogram (LBPH) approach for reasons elaborated in the corresponding section

3 Segmentation

3.1 Overview

Iris Segmentation is the first step of Iris Detection. The purpose of Iris Segmentation is to isolate the iris region. The isolated iris region is further used as an input to perform subsequent Iris Detection steps which are Image Normalization and Feature Matching to define uniqueness of the iris.
For grasping the algorithm better we will first discuss *contours*.
Contours are enclosed curves that possess a geometric shape. These geometrical shapes can be realized through their areas. An object in the image can be distinguished by it's geometrical shape. Since, these contours possess certain area, a software can realize various objects in an Image by calculating their respective area.

3.2 Hough Transform

It's a Feature extraction technique which is used in various fields like Digital Image Processing, Computer Vision and many more. A digital image is processed in the form of a matrix of size (Image Height x Image Width). The elements of this matrix are the pixel intensity values. These range from 0 to 255 for an RGB image.
Based on these pixel values, Hough Transform finds the parameters of geometric objects like circles and lines available in the image. An algorithm considering circular Hough Transform is used by (Ma et al., 2002) and (Tisse et al., 2002) Circular Hough Transform calculates the radius r and Centre coordinate x and y of the circles present in a circular geometric object. A circle can be defined by the equation:
$x^2 + y^2 = r$

3.3 Implementation and Results

We are using OpenCV open source library to perform image processing. It is widely used in the field of Computer Vision as a cross platform library. OpenCV is used as a wrapper between the application and various Image Processing algorithms. This library being native to C++ works efficiently in real time applications. Our application uses functions provided by OpenCV, which would be discussed further.
Iris Segmentation is subdivided into three parts of segmentation:

1. **Pupil Segmentation**:All the contours available in the image are identified. OpenCV method *findContours()* does the job for us by identifying them. Sometimes, these methods can provide noise present in the image as contours as well. When the noisy contour area goes beyond a specific value, these noisy contours are filtered out.

   OpenCV method *contourArea()* is used to calculate the contour area. It takes a particular contour as input parameter.After the filtration, though there are possibilities of multiple contours being present in the iris image. The contour with the center closest to the centre of

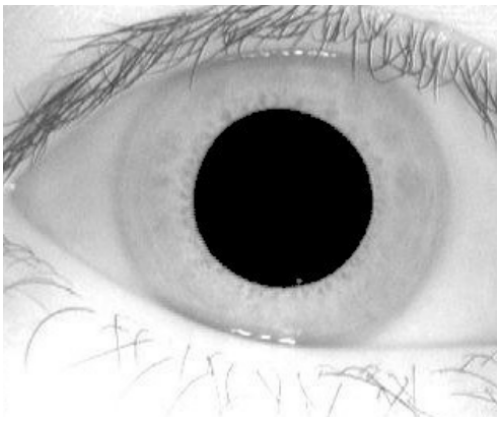

Figure 3.1: Contour in the form of pupil of the eye

the image is identified to be the desired contour. Pupil happens to be the contour (geometric shape) closest to the centre of iris image. To segment the pupil region, it is masked with zero pixels as can be seen in the image above. OpenCV method *drawContours()* is used.

2. **Outer Iris Segmentation:** The next step of segmentation algorithm deals with detecting the outer iris of the image. Canny edge filter is applied. It works such that whenever there is a sudden change of pixel intensity values, it considers it as an edge. These edges form the borders available in the image. OpenCV method *Canny()* performs the edge detection. It takes a threshold pixel value as an input parameter which is specific to CASIA database image. OpenCV method *GuassianBlur()* is used to further smoothen the image. Hough transform is applied through the OpenCV method *HoughCircles()* to find the outer Iris boundary of the image . Furthermore, through the OpenCV method *circle()*, a circle with red pixels distinguishes the iris boundary. This can be seen in the image below:

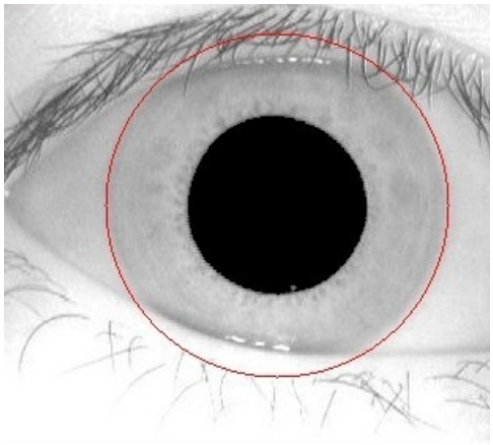

Figure 3.2: Outer Iris circle identified by Hough Transform

3. **Final Segmented Iris region:** The unwanted region is masked with black pixels. The

resulting image is the final segmented region, to be further used as the input for the Image Normalization stage.

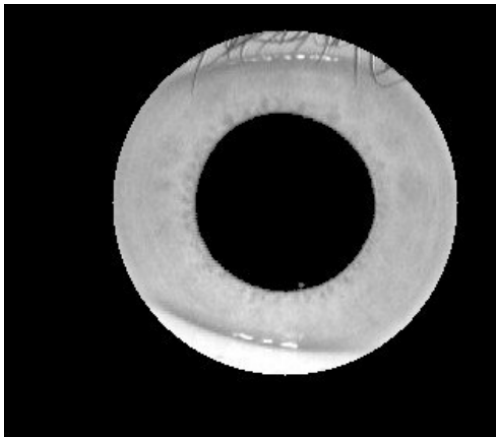

Figure 3.3: Segmented Iris region

For a detailed description of all above listed the functions please refer (opencv.org, 2016).

4 Normalization

Normalization involves the process of *unwrapping* the iris and converting it form the cartesian to the polar coordinates. The iris size is dependent on the pupil size which could be inconsistent due to several factors. Pupil sizes may vary to accommodate changes in the light intensities while capturing the eye image, changes in camera position, change in the working distance and others. to compensate these variation, normalization techniques are applied to the segmented image. It also takes care of non-concentric pupil modifications While there are many different algorithms available we have used the approach suggested by John Daugman in his paper (Daugman, 2009), the Daugman's Rubber Sheet model.

4.1 Daugman's Rubber Sheet Model

The technique transforms every pixel in the the isolated iris image, obtained post segmentation, from the cartesian coordinates to a corresponding polar coordinate (r, θ). Figure 3.1 provides an illustration for this. The mapping of the iris texture from the pupil border to the limbus border is captured by r which lies in the interval of [0,1]. θ denotes the angle with values in the range of [0,2π]. The remapping model is depicted as: $I(x(r,\theta), y(r,\theta)) \rightarrow I(r,\theta)$, with

$$x(r,\theta) = (1-r)x_p(\theta) + rx_l(\theta)$$
$$y(r,\theta) = (1-r)y_p(\theta) + ry_l(\theta)$$

where $I(x,y)$ represents the iris texture, (x,y) are the original cartesian coordinates, (r,θ) are the transformed polar coordinates, x_p, y_p represent the pupil border and x_l, y_l represent the limbus border (Masek et al., 2003).

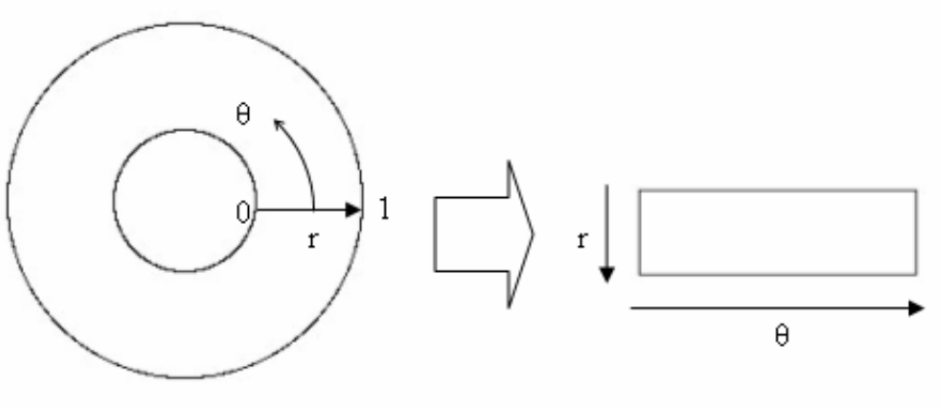

Figure 4.1: Daugman's Rubber Sheet Model
(Masek et al., 2003)

4.2 Implementation

Before unwrapping, we first find the centre of the isolated iris area in the image obtained in the previous step. Then we apply the *logPolar* function in the OpenCV library to perform the actual normalization.
The function prototype is given below:

void *logPolar*(***InputArray*** *src*, ***OutputArray*** *dst*, ***Point2f*** *center*, ***double*** *M*, ***int*** *flags*)

Parameters (opencv.org, 2019a):

src Source image

dst Destination image. It will have same size and type as src.

center The transformation center; where the output precision is maximal

M Magnitude scale parameter. It determines the radius of the bounding circle to transform too

flags A combination of interpolation methods

A detailed description of all the parameters can be found in the following reference (opencv.org, 2019a). Here we will discuss the important parameters with respect to the implementation.
We call the function as below:

logPolar(this->regionCropped, unwrappedData, centerValue, *40*, *INTER_LINEAR + WARP_FILL_OUTLIERS*);

The important parameters in this function are the flags:

WARP_FILL_OUTLIERS: This indicates that the destination image pixels will be padded. It also sets the outliers in the source image to zero.

INTER_LINEAR: : This indicates that bilinear interpolation must be performed for the purpose of image rescaling. Its aids in the padding of the image.

The combination of the two ensures commonality in the image domain, thereby accounting for the variations in pupil size.

4.3 Results

We were successful in obtaining a normalized output for our iris image. Figure 3.2 illustrates the result of this phase for a sample image.

Figure 4.2: The Normalized Iris Image using the Cartesian to Polar Transformation

5 Feature Encoding and Matching

5.1 Overview

The final step required to perform the detection of Iris is the Feature Encoding and Matching part of the algorithm. The step following the Normalization step, involves extracting the significant features of each iris, so that a template can be made to match it to the test images. The template that is generated in the feature encoding process will also need a corresponding matching metric, which gives a measure of similarity between two iris templates.(Masek et al., 2003)

5.2 Local Binary Pattern Histogram (LBPH)

The Gabor Filter algorithm is most preferably used to perform feature encoding. Gabor filters are able to provide optimum conjoint representation of a signal in space and spatial frequency. A Gabor filter is constructed by modulating a sine/cosine wave with a Gaussian.(Masek et al., 2003). Initially we tried to realize the final part of the application used Gabor Filters, but due to the time constraint, we were not able to finalize on the results. Hence, we went ahead with utilizing the Face Recognizer class in OpenCV to perform the result comparison between normalized images.
The algorithm that we utilized for this is called the Local Binary Pattern Histogram (LBPH) algorithm. The idea with LBPH is not to look at the image as a whole, but instead, try to find its local structure by comparing each pixel to the neighbouring pixels (SuperDataScience, 2017).
We take a 3x3 window and slide it across our image. At each position, we compare the intensity of the pixel at the centre against the intensity of the pixels that are present around it. We then specify these neighbouring pixels with 0, if their intensity is less than the centre, and 1 if otherwise. We proceed then to read these ones and zeroes in a clockwise manner and we then create a binary pattern, for example: 11100011 that corresponds to a certain area in the image. When we finish performing this for the entire image, we have a list of Local Binary Patterns.
The image below showcases the basic idea behind the steps described in the previous paragraph.
After the list of binary values are computed, we then calculate the corresponding decimal values of these binary values. Following this step, we create a histogram of these values which serves as the histogram for the image. We train our model(explained in next section) using multiple such histograms of the expected image, and this model is then utilized to make decisions whether a certain Iris image matches our template.

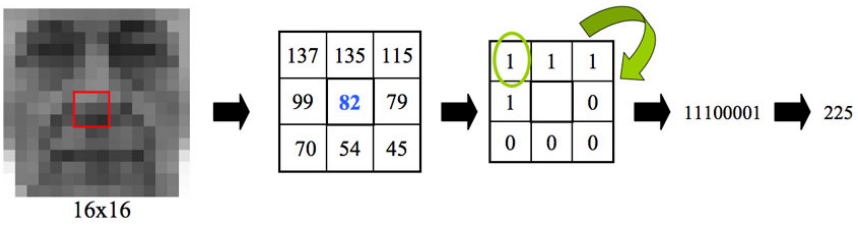

Figure 5.1: LBP conversion to binary
(SuperDataScience, 2017)

5.3 Implementation and Results

We utilize the above said algorithm in order to create templates of the images, which are to be used as the reference in our comparisons. As stated previously we will be utilizing the Face Recognizer class present inside OpenCV library in order to achieve this. We can describe the entire process using two important functions present in the Face Recognizer class: *train()* and *predict()*.

cv2.face.LBPHFisherFaceRecognizer(): We utilize this function in the beginning the generate a mode object which will be trained using the images which were obtained after the normalization phase.
train(): We utilize this function to feed our model with a set of reference images. This model then serves as the model which is utilized to predict the test images, and confirm whether they are the same as our templates.We used a total of 12 normalised iris images of the left and right irises for both the users User 1 and User 2.
predict(): After transforming our test image also accordingly, we then use our model created through the function to execute the predict function with the test image as an argument in it. If the test image is indeed matching with the set of template/reference images, then we provide a non-zero output, which will confirm that a match has been detected.

6 Flowcharts

In the following pages, we have the flowcharts for the overall application and for each phase.

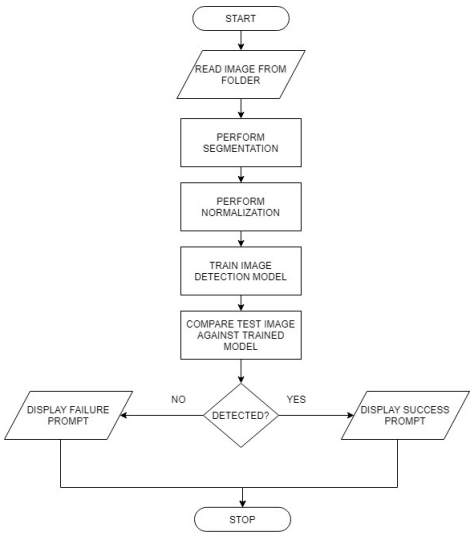

Figure 6.1: Overall Application Flowchart

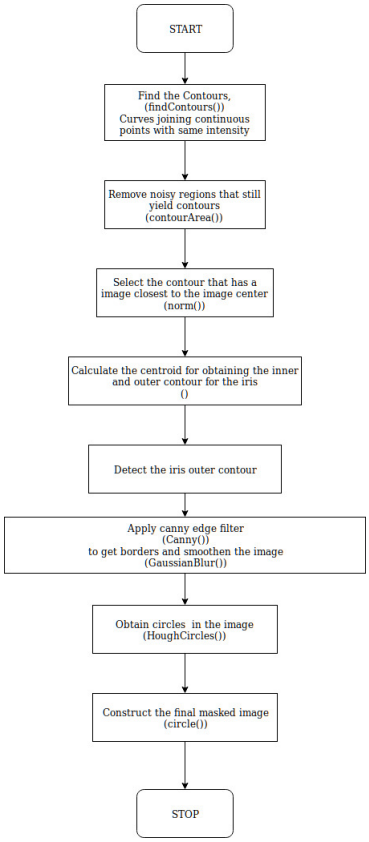

Figure 6.2: Segmentation Flowchart

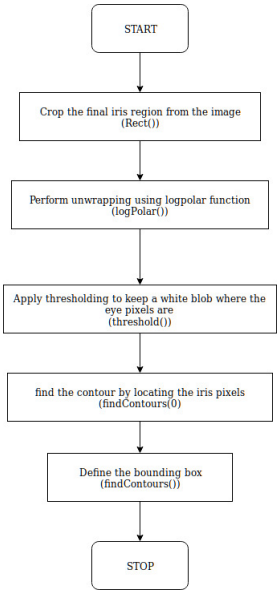

Figure 6.3: Normalization FLowchart

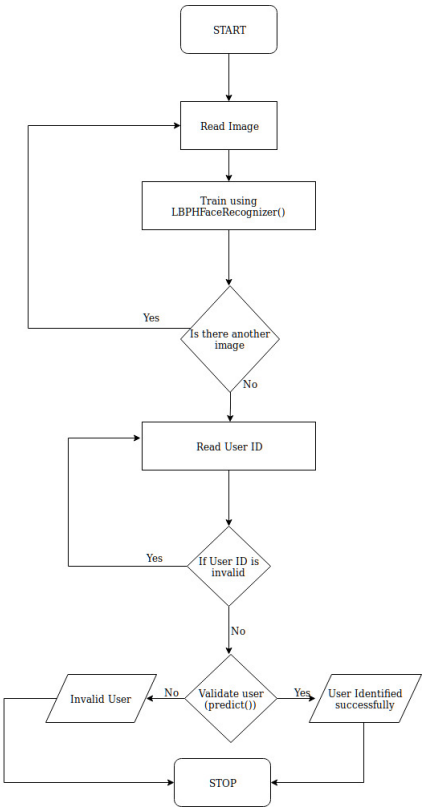

Figure 6.4: Feature Matching via LBPH Flowchart

7 Discussion and Result

The paper presents an iris detection system. First, implementation started with Java as the programming language but it could not handle OpenCV type mismatches like conversion of *MatOfPoint* to *MatOfPoint2f* class. This led us to move to Python. Here as well, the implementation was constrained by the ability of wrapper functions to not work as expected by the application. the *HoughCircles()* function inexplicably detected a different set of circles in the Python version as compared to the C++, with the latter giving the desired output. Hence, we changed our programming platform to C++. OpenCV is developed using C++. Also, it provides image processing algorithms with efficient wrapper functions to work on. It helped us to successfully complete the implementation of application using C++. The application is simulation based which has been tested and verified successfully for NIR images from CASIA database. Below are screenshots of our application for a user with User ID 1 and with the correct iris image captured for that user, and for a user with User ID 2 and with the incorrect iris image for that user.

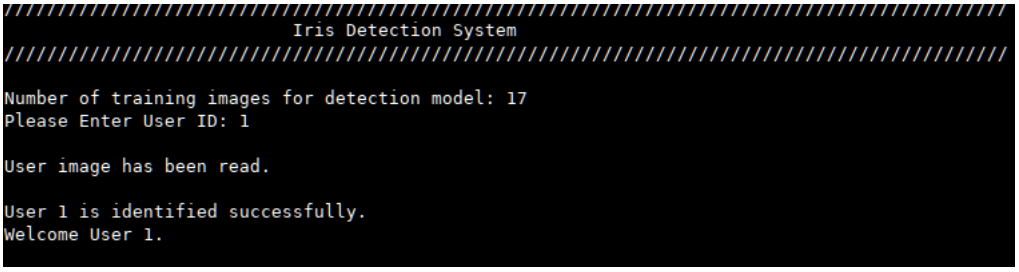

Figure 7.1: User with User ID 1 and with the Correct Iris Image

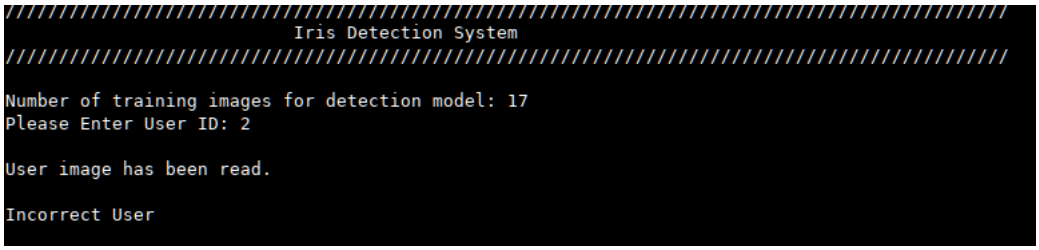

Figure 7.2: user with User ID 2 and with the Incorrect Iris Image

The application has been tested considering various test cases like adding impurities to image, altering the test brightness, altering the contrast of the image. For this we used the GNU Image Manipulation Program(GIMP). We found that adding impurities to the image, as long as the structure of the iris was not affected, still allowed for proper validation. Increasing or decreasing the brightness or the contrast by 30 units also didn't affect the performance or working. However decreasing the contrast by 30 units and altering the brightness by 50 units resulted in major loss of information, thereby causing the application to fail. However, false positives were also detected when decreasing the brightness by 30 and increasing the contrast by 50, thus not making our approach fool-proof.

8 Future Scope and Conclusion

The application is able to perform properly but there are few problems as seen in the previous section. The software has to be made flexible enough to work for different set of images. An improvement would be to write image processing algorithms instead of using OpenCV library which can lead the application to work faster. This also strengthens the real-time nature of the application. Frame grabber can be used to capture a set of images for processing instead of processing one static image at a time. The presence of false positives means our system is not suited for security based applications. This can be improved in the Feature Encoding and Matching portions, by implementing the original algorithm of Gabor filters, or other encoding techniques. Analysis of the encoded values will possibly allow the system to eliminate the false positives.

9 Project Download

You can follow the Google Drive link to download the project. Instructions for use are provided in the *Readme.md* file.
`https://drive.google.com/file/d/1waBJI37yLasO_L_HA-dYlBBactV211RY/view?usp=sharing`

References

apsubiology.org (2015). Iris image. http://www.apsubiology.org/anatomy/2010/2010_Exam_Reviews/Exam_4_Review/CH_15_Accessory_Structures_of_the_Eye.htm. Accessed: 2019-03-22.

Daugman, J. (2009). How iris recognition works. In *The essential guide to image processing*, pages 715–739. Elsevier.

Ma, L., Wang, Y., and Tan, T. (2002). Iris recognition using circular symmetric filters. In *Object recognition supported by user interaction for service robots*, volume 2, pages 414–417. IEEE.

Masek, L. et al. (2003). *Recognition of human iris patterns for biometric identification*. PhD thesis, Masters thesis, University of Western Australia.

Mohammadi Arvacheh, E. (2006). A study of segmentation and normalization for iris recognition systems. Master's thesis, University of Waterloo.

opencv.org (2016). Feature detection - image processing. https://docs.opencv.org/3.2.0/dd/d1a/group__imgproc__feature.html. Accessed: 2019-03-23.

opencv.org (2019a). Geometric image transformations. https://docs.opencv.org/3.4/da/d54/group__imgproc__transform.html#gaec3a0b126a85b5ca2c667b16e0ae022d. Accessed: 2019-03-22.

opencv.org (2019b). Opencv. https://opencv.org/. Accessed: 2019-03-25.

SuperDataScience (2017). Face recognition using opencv and python: A beginner's guide. https://www.superdatascience.com/blogs/opencv-face-recognition. Accessed: 2019-03-23.

Tisse, C.-l., Martin, L., Torres, L., Robert, M., et al. (2002). Person identification technique using human iris recognition. In *Proc. Vision Interface*, volume 294, pages 294–299. Citeseer.

Wildes, R. P. (1997). Iris recognition: an emerging biometric technology. *Proceedings of the IEEE*, 85(9):1348–1363.