An Iris Authentication System Based on Artificial Neural Networks

by

Brenden Velu

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2015

TABLE OF CONTENTS

D. **LIST OF TABLES AND FIGURES**

Tables

Figures

Abstract

An iris authentication system verifies the authenticity of a person based on their iris features. The iris features are extracted through wavelet transform of the isolated iris from modified iris images. A level 5 wavelet decomposition is performed on the images, and the resulting low-frequency wavelet coefficients represent the inputs to the artificial neural network. The artificial neural network reads these features as inputs, and classifies each set of inputs according to their target identity. This authentication system currently classifies up to 10 people. The irises used for classification represent ideal situations with minimum eyelash and eyelid interference.

Chapter I: Iris Authentication System Overview

The iris identification system authenticates a person's identity based on their iris features. Iris recognition is a greatly effective recognition method because the iris possesses highly unique characteristics pertaining to the individual [1]. Other common recognition methods include face, voice, and fingerprint analyses. Iris recognition is regarded as a more efficient method than others because its features are the direct result of environmental conditions in the development stage, as well as the iris's lower chance of sustaining permanent damage [2].

The context for this project originates from the importance of security systems in the modern age. Information security becomes more valuable as the rate of technology advancement increases in society. The motivation for this project originates from this awareness as there is a necessity to protect information with secure methods. Iris identification provides one form of security because the iris possesses many complexities.

This recognition system consists of three steps to perform authentication: iris segmentation, feature extraction, and classification. The segmentation procedure consists of pupil and iris boundary detection so that center coordinates and the radius of both circles are retrieved. For this detection, the circular Hough transform is applied. The goal for iris segmentation is to obtain a rectangular image representing the unwrapped isolated iris. With this image, iris features can be extracted. The iris images are obtained from the Chinese Academy of Sciences' Institute of Automation (CASIA) database, which consist of 480 x 640 resolution grayscale images. For each person in the CASIA database, there are 10 unique iris images. Half of the images are used for training the neural network, and the other half are used for simulation. The images used to train the neural network are shown in Figure 1. The images used to test the neural network are shown in Figure 2. In Figures 1 and 2, the 5 images per person mainly differ in lighting conditions and image quality, as seen in the differing pupil sizes and image blur respectively.

Figure 1: Iris Images used for Training (each row represents a different person) [16]

Figure 2: Iris Images used for Testing (each row represents a different person) [16]

To train the neural network, each of the 5 training images are rotated by a randomly generated angle 100 times. As a result, there are 5000 modified images used to train the neural network. For simulation, each of the 5 testing images are rotated by a randomly generated angle 10 times. There are 500 modified images used to test the neural network.

The feature extraction step receives the unwrapped iris image and applies a level 5 wavelet decomposition. The important results of the decomposition are the low-frequency wavelet coefficients which are normalized and then sent as inputs to the artificial neural network.

The artificial neural network performs classification using the coefficients obtained from feature extraction. The neural network is trained with different sets of coefficients corresponding to different identities, and stores the results in a database. Then, it is tested with new sets of coefficients by comparing them to information stored in the database to find the correct identity [3].

As seen in figure 3, the block diagram and function table consists of two subsystems: a feature extraction system and a classification system. The iris image is input into the feature extraction system. This system reduces and transforms the image with the Haar wavelet transform so there is data prepared for classification [8]. Table I summarizes the input and output of the feature extraction system. The feature extraction system's output is input

into the classification system. The classification system uses this input to compare them with features from the database [4]. The comparison process implements artificial neural networks. The extracted features are the low-frequency coefficients resulting from the wavelet transform, and they represent individual inputs to each node in the input layer. Each node sends an output to sets of nodes in each layer until a result is determined. Table II summarizes the input and output of the classification system.

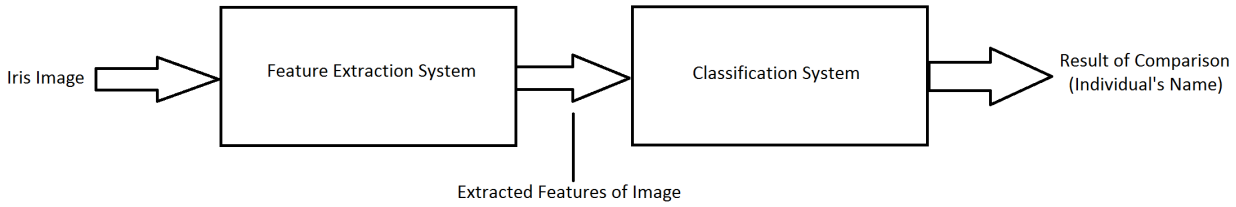

Figure 3: General System Block Diagram

TABLE I

FEATURE EXTRACTION SYSTEM BLOCK DIAGRAM FUNCTION TABLE

	Description
Input	• 480 x 640 resolution iris image. Images with minimum eyelash or eyelid interference are used.
Output	• The low-frequency wavelet coefficients per iris image. A level 5 decomposition is performed.
Function	• The feature extraction system receives an iris image to extract iris features using the Haar wavelet transform. These resulting wavelet coefficients become the input of the classification system.

TABLE II

CLASSIFICATION SYSTEM BLOCK DIAGRAM TABLE

	Description
Input	• The low-frequency wavelet coefficients.
Output	• The result of the classification process outputs a binary representation of the iris identity, and displays a message (“Success” or “Failure”) depending on accuracy of the identification.
Function	• The classification system receives the extracted iris features and compares them to the stored features in the database. The results of this comparison indicate success or failure by showing the binary representation of the iris identified by the neural network.

Chapter II: Literature Review

[1] L. W. Liam et al., "Iris Recognition Using Self-Organizing Neural Network," in Research and Development, 2002 © IEEE. doi: 10.1109/SCORED.2002.1033084

The source's iris recognition methods used relate heavily to my chosen project. The source can inform the project because both methods involve artificial neural network recognition systems dealing with iris identification. The artificial neural network analyzes constructed images following camera captures and detects patterns. The source possesses authority because it claims 9 citations and the IEEE database lists it as one of its publications.

[2] R. M. Farouk et al., "Iris Matching using Multi-Dimensional Artificial Neural Network," in Computer Vision, IET, 2011 © IEEE. doi: 10.1049/iet-cvi.2010.0133

This source employs multi-dimensional artificial neural networks so that the entire iris can input rather than a time-affected repurposed image. This source can inform my project since the multi-dimensional neural network method can significantly increase accuracy because its inherent recognition abilities prove accurate. The source claims reputability because it possesses 3 citations and claims IEEE publication.

[3] C. Gerhensen, "Artificial Neural Networks for Beginners," Cornell University Library. Ithaca, NY, Rep. arXiv:cs/0308031, Aug. 2003. Available: http://arxiv.org/pdf/cs/0308031.

This source contains beginner knowledge regarding artificial neural network workings. The source can inform my project through the initial working mechanics regarding how neural networks operate improving conceptual understanding. The source possesses 50 citations through other works, and the Cornell University Library published the document.

[4] A. Nait-Ali and R. Fournier, "Signal and Image Processing for Biometrics," Hoboken, NJ: John Wily & Sons, Inc., 2012.

This book teaches biometric signal processing methods. The project employs signal processing when the system receives the input following the iris scan. The source should inform the project using the signal processing principles. The source contains authority through the authors who possess instructor positions.

[5] Y. Min et al., "Applications of Generalized Learning in Image Recognition," in Neural Interface and Control, 2005 © IEEE. doi: 10.1109/ICNIC.2005.1499867

The source informs a learning type called generalized learning which claims new image recognition model introduction. This learning type fuses multiple learning types together inducing an adaptive learning capability which recognition projects can utilize. The source claims authority through the references which list reputable sources, such as IEEE publications. The source itself claims IEEE publication.

[6] H. Rashid et al., "Design Method of Video Based Iris Recognition System (V-IRS)," Universiti Kebangsan. Selangor, Malaysia, Nov. 2013.

The source provides information regarding iris recognition methods which prove inferior when compared against video based iris recognition systems. Although not directly artificial neural network related, it highlights various recognition methods and their positives and negatives. This information improves general iris recognition knowledge regarding the project. The document possesses authority through its reference section. The extensive reference section includes many reputable sources. The document also received a Malaysian university publication.

[7] K. Kim et al., "Biometric Identification Apparatus and Method using Bio Signals and Artificial Neural Network," U.S. Patent 7 630 521, December 8, 2009.

The patent involves bio signal and artificial network use which detects living body signals. The technology does not involve iris identification, however, the technological principles employed when detecting bodily signals can relate. The source possesses authority through official publication as a patent.

[8] P. Georgieva et al., "Advances in Intelligent Signal Processing and Data Mining," Heidelberg, Germany: Springer, 2013.

The source possesses multiple chapters which may aid artificial neural network implementation including object recognition and reinforcement learning. The object recognition section specializes scenarios where objects may prove difficult detection. This section informs the project giving clearer iris recognition capabilities. A key method artificial neural networks employ revolve around reinforcement learning so that correct patterns the system learned can implement itself. The source proves reputable because the authors possess instructing university positions residing the University of Aveiro, Lancaster University, and the University of South Australia. The references section also possesses a varying reputable reference set.

[9] S. H. Chagas et al., "An Approach to Localization Scheme of Wireless Sensor Networks based on Artificial Neural Networks and Genetic Algorithms," in New Circuits and Systems Conference, 2012 © IEEE. doi: 10.1109/NEWCAS.2012.6328975

The source proposes wireless sensor networks utilizing artificial neural networks. This idea could inform my project because it utilizes the same artificial neural network technology, but applies positional tracking through the sensor networks inducing greater capturing accuracy. The source claims one citation by IEEE and the IEEE database published the report.

[10] B. Kovoor et al., "Effectiveness of Feature Detection Operators on the Performance of Iris Biometric Recognition System," International Journal of Network Security & Its Applications (IJNSA). Vol. 5, No. 5, September 2013.

This paper discusses methods used to perform segmentation and feature extraction. The important information taken from this piece is the effectiveness of the Canny Operator for edge detection. The authors tested various edge detection operators and tabulated their results. The Canny Operator shows a significantly greater success ratio. This is relevant to my project because the Canny Operator is used to

identify how much information is being processed by the circular Hough transform function and the discrete wavelet transform steps.

[11] K. Lee et al., “Efficient Iris Recognition through Improvement of Feature Vector and Classifier,” ETRI Journal. Vol. 23, p. 61-70, 2001.

The authors of this paper employed the Haar wavelet transform for their feature extraction process. The paper also includes descriptions about segmentation and classification steps. To extract the low-frequency wavelet coefficients from their normalized iris, they used the Haar Mother Wavelet transform. This is relevant to my project within the feature extraction step where I use the Haar wavelet transformation to extract the wavelet coefficients.

[12] U. Dias et al., “A Neural Network Based Iris Recognition System for Personal Identification,” ICTACT Journal on Soft Computing, Issue: 02, October 2010.

The writers of this paper performed a series of tests using the different backpropagation algorithms and reported their results. From their data, the Traincgb function is one of the more effective algorithms to use since they received high accuracy results at 94.24% with a relatively fast training time at ~2:30 hr. This is relevant to my project because the backpropagation algorithm is used to train the neural network. As a result, the Traincgb algorithm is used for this project.

[13] B. Kovoor et al., “Iris Biometric Recognition System Employing Canny Operator,” Computer Science & Information Technology. pp. 65-74, 2013. @CS & IT-CSCP 2013. DOI: 10.5121/csit.2013.3307.

This paper introduces the process of applying histogram equalization to the unwrapped iris image to enhance the iris textures. Using the Canny Operator on this image will show how the edges are more pronounced after histogram equalization. The method is relevant because it is used in this project after unwrapping the iris image. The histogram equalization is performed before performing discrete wavelet transformation.

[14] M. Elgamal and N. Al-Biqami, “An Efficient Feature Extraction Method for Iris Recognition Based on Wavelet Transform,” International Journal of Computer and Information Technology. Vol. 2, Issue 03. May, 2013. ISSN: 2279-0764.

This paper discusses how to use the discrete wavelet transformation to find the low-frequency coefficients of a normalized iris image. It also mentions methods which are used to isolate the iris and prepare it to be sent to the feature extraction step. The methods used in the paper are methods similar to those used in this project, albeit with some changes.

[15] Homepages.inf.ed.ac.uk, 'Feature Detectors - Canny Edge Detector', 2015. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm. [Accessed: 26- May- 2015].

This website discusses how the Canny Edge detection method is achieved through a step-by-step procedure. This is relevant to my project because the Canny Edge detection method is used to identify how distinct the iris features are before they are sent to further feature extraction steps.

[16] Biometrics.idealtest.org, 'Biometrics Ideal Test', 2015. [Online]. Available: http://biometrics.idealtest.org/dbDetailForUser.do?id=4. [Accessed: 01- Jun- 2015].

This reference is the CASIA database from which all the iris images used to train and test the classification system were taken from. The images used were from the Iris-Syn set of irises which the sources encourages to use for iris recognition research.

Chapter III: Iris Authentication System Design

Pupil Boundary Detection

Figure 4 shows the block diagram of the segmentation/feature extraction algorithm. There are multiple processes applied to the iris image before it is ready for feature extraction. In order to obtain enough training and testing data for the classification process, each iris image is rotated within a range of -10° to +10°. The angle of rotation is determined by using Matlab's random number generator function (rand()). Figure 5 shows a sample image rotated by 7°. For each iris image input, there are 100 resulting modified iris images which are sent for feature extraction so that the neural network can be trained.

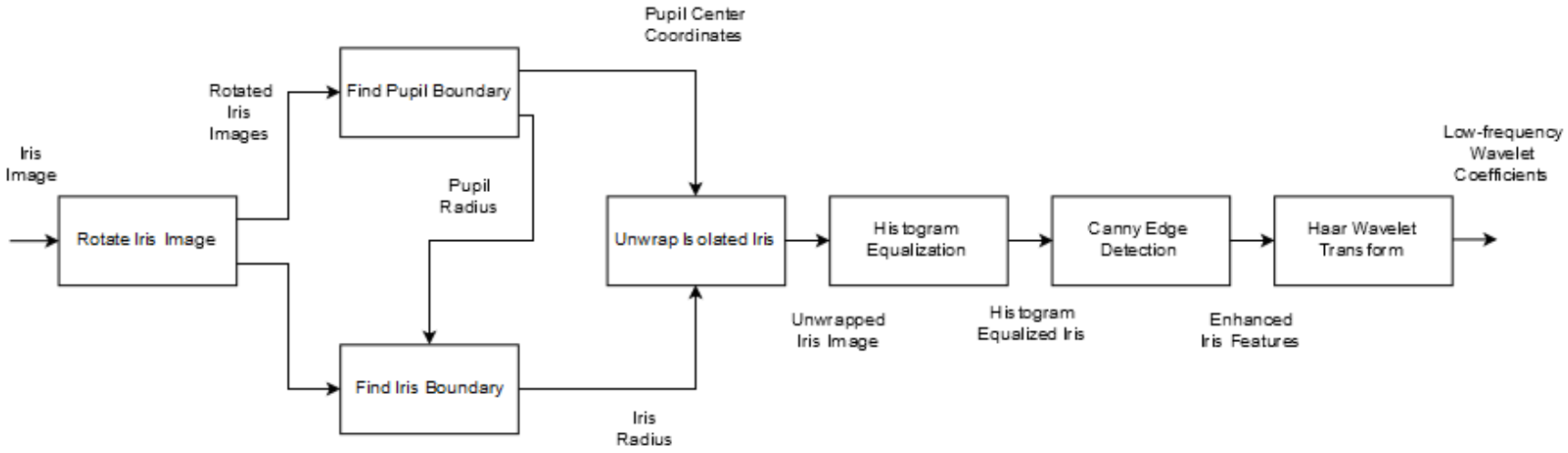

Figure 4: Segmentation/Feature Extraction Block Diagram

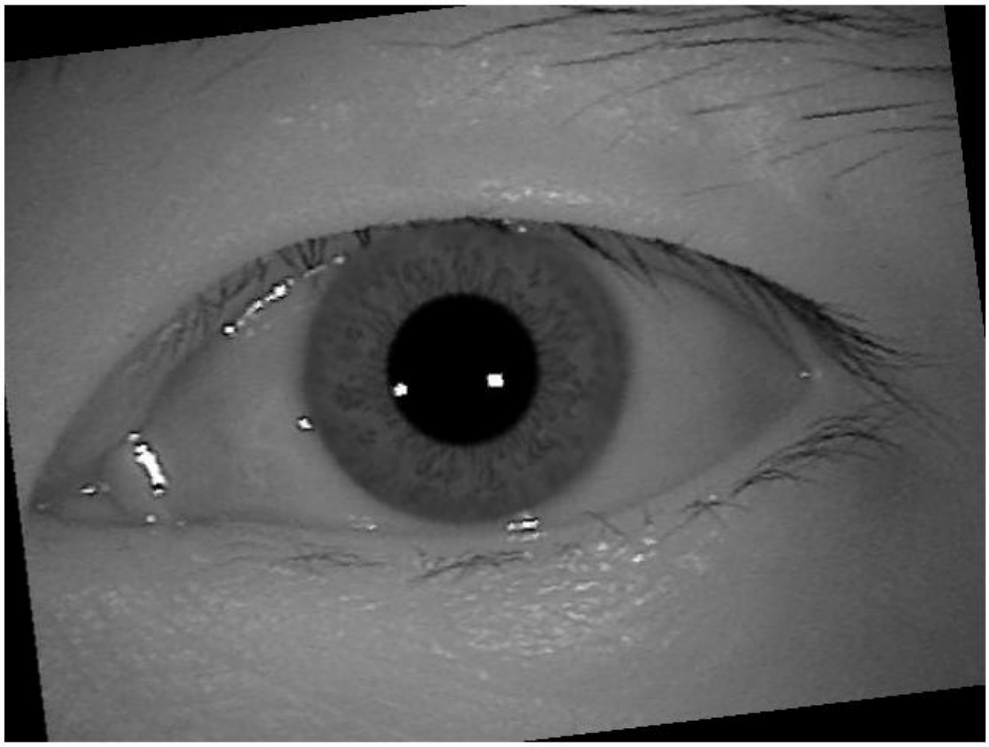

Figure 5: Rotated Iris Image

Matlab's circular Hough transform function is applied to each modified image twice: once for finding the pupil boundary, and once for finding the iris boundary. However, when detecting the iris boundary, multiple circles are often detected because of noise or unwanted objects in the image. To circumvent this issue, the pupil radius is sent to the iris boundary detection algorithm so that a difference between the iris radius and pupil radius is calculated. This distance represents a threshold the iris radius must exceed so that the correct circle information is output from the circular Hough transform. To prevent errors during feature extraction, this threshold is set to 32 pixels.

The threshold value was determined through trial-and-error, and it ensures that the low-frequency coefficient matrices have the same sizes. When the distance between the iris radius and the pupil radius was less than 32, the number of coefficients resulting from feature extraction was less than 24 in some cases. This caused errors because the feature extraction is designed to take 24 coefficients from a portion of the image after 5 decompositions. Every iris image had to produce 24 coefficients otherwise there would be a mismatch in matrix size to send as inputs to the neural network.

The reason for there being less than 24 coefficients at times is because the circular Hough transform was detecting multiple iris and pupil circles in the image, and the circles which incorrectly bounded the iris and pupil were being chosen. The value was set to 32 because the correct iris and pupil circular

boundaries had a difference in radius of at least 32 pixels which meant that the code would choose these circles to output to the feature extraction step.

The circular Hough transform detects circles by finding the intersection points of many circles drawn on the image. After a specific radius range or exact radius is specified, a circle is drawn at each (x,y) point on the original image with the radius specification. An accumulator matrix is created for the image which contains only zeros, and for each pixel containing an intersecting circle, the corresponding pixel's zero value is incremented by 1. When the algorithm has finished processing each pixel and incrementing the accumulator matrix, the pixels with the highest values in the matrix represent the center coordinates of the circle for the original image. The following equation is used in this algorithm, the (a,b) point represents the center coordinate of a circle in the original image:

$$(x - a)^2 + (y - b)^2 = r^2$$

The purpose of segmentation is to remove all regions besides the iris within the image so that the iris can be transformed into information suitable for identification. The pupil segmentation process begins with identifying the pupil's center coordinates and radius so that a circle can be drawn along the borders of the pupil-iris boundary. Figure 6 shows a sample iris image used to test the boundary detection algorithm's accuracy.

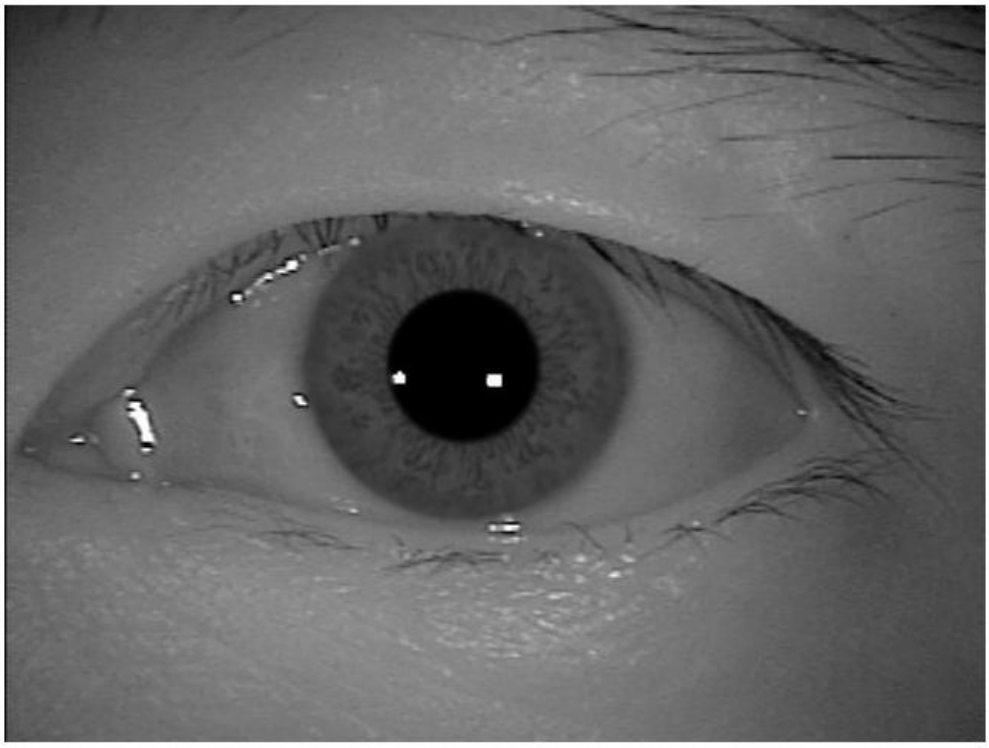

Figure 6: Iris Image Analyzed

To determine the pupil-iris boundary, Matlab's circular Hough transform function (imfindcircles()) is used with user-defined values for the radius range, sensitivity, and edge threshold. The function gives

output values for the pupil center coordinates, and the pupil radius. Given that the images tested are of 480 x 640 resolution, it is assumed that the pupil's radius is within the range of 35 and 75 pixels. The sensitivity parameter dictates how many circles the function will detect, and it is bound by a [0, 1] range. The edge threshold parameter sets the threshold for finding edge pixels. A lower value means that circles with fainter edges are more likely to be detected. This parameter is also bound by a [0, 1] range. The sensitivity parameter was set to 0.95, and the edge threshold parameter was set to 0.05. The sensitivity parameter is set to 0.95 so that the function can detect more edges. The 0.05 edge threshold value allows enough sensitivity to smooth edge detection so that the pupil boundary can be found. The parameter values were determined through trial-and-error; the majority of the other values tested did not accurately determine the pupil boundary.

Using Matlab's canny edge detection method (edge()), Figure 7 shows the edges detected within the iris image. As seen in the image, the pupil's edge is clearly defined, while the iris's edge is not as defined. This observation is important to note when determining appropriate parameter values, specifically for setting the edge threshold.

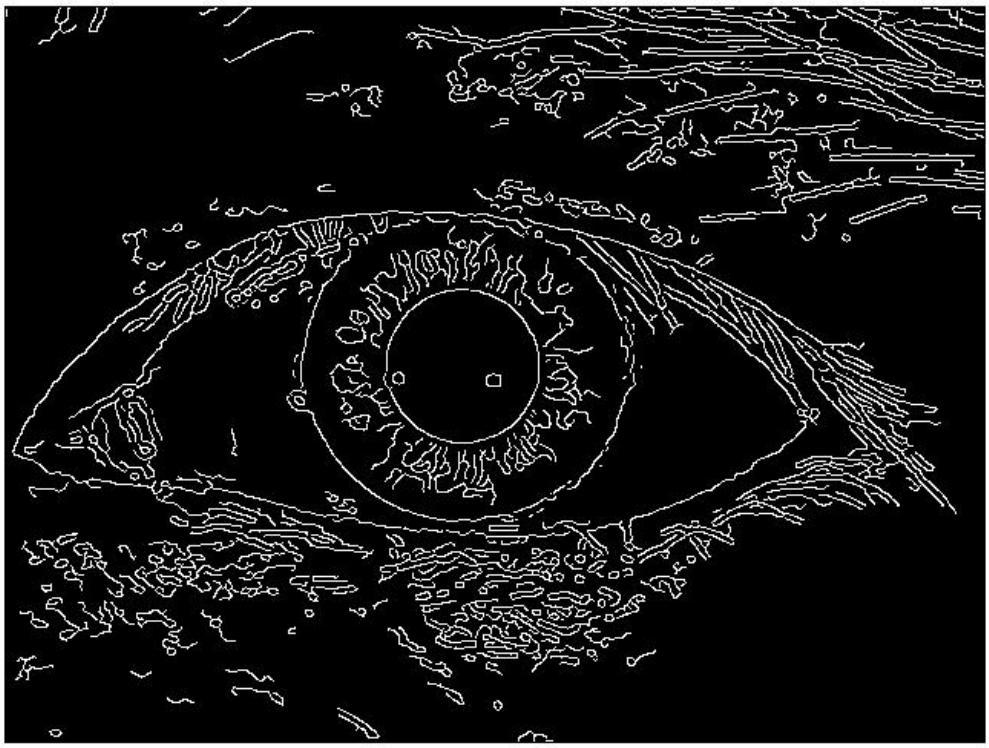

Figure 7: Canny Edge Detection applied to Iris Image

The pupil boundary circle determined for this iris image is shown in Figure 8.

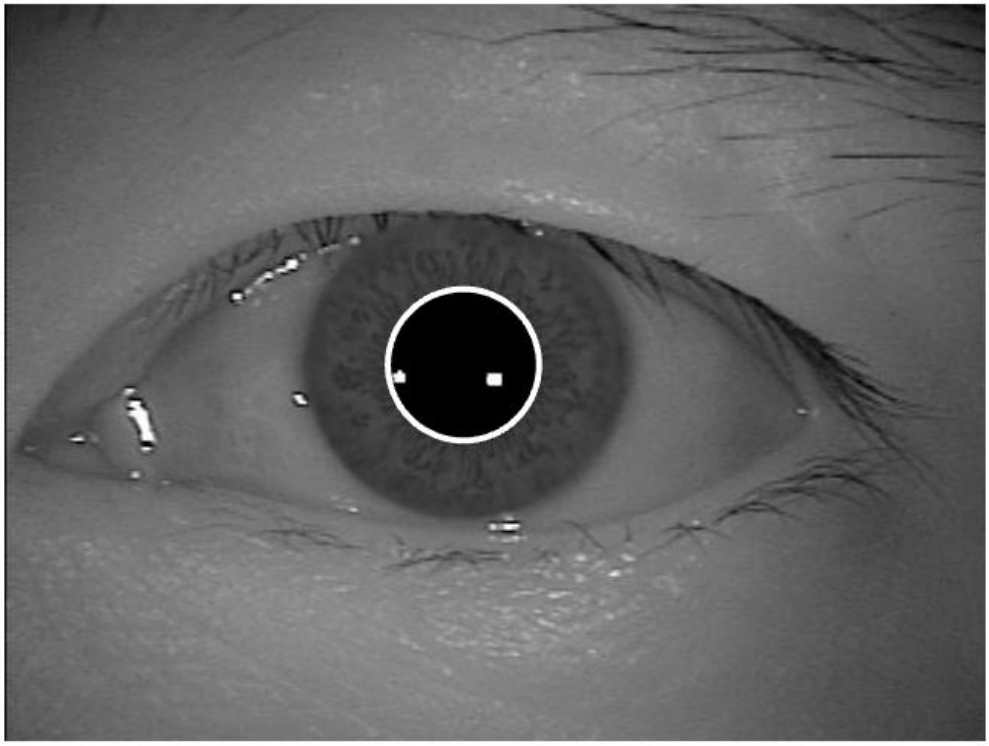

Figure 8: Circle bordering pupil-iris boundary

Iris Boundary Detection

The current iris segmentation process is similar to the processes used for pupil segmentation. This step will have the Hough transform function output the iris center coordinates, and the iris radius. As noted in the edge image, the boundary bordering the iris and sclera is less distinct than the pupil boundary so the circular Hough transform parameters must be adjusted. The radius range is adjusted to 70 to 130 pixels. The sensitivity parameter is adjusted to 0.97, and the edge threshold parameter is adjusted to 0.005. The sensitivity parameter is adjusted to 0.97 so that the function can detect more circles, and the edge threshold parameter is set to 0.005 so that the function can detect very smooth edges. These parameter values were also determined by trial-and-error, and they provided the most accurate iris boundaries out of the values tested.

The iris boundary is more difficult to detect because the edges are not as defined as with the pupil, so the parameters are adjusted so that circles detection is more aggressive. These parameter values indicate greater aggression in finding circles in the image which is necessary given that the iris edge often blends slightly with the sclera. The circle bounding the iris is shown in Figure 9.

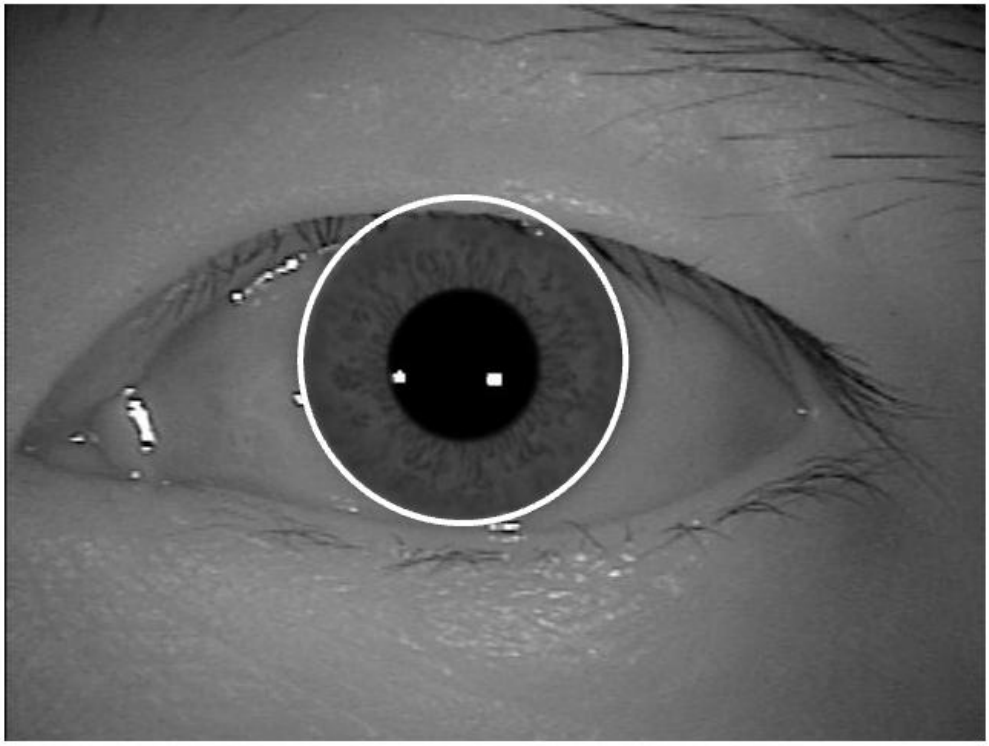

Figure 9: Circle bordering iris-sclera boundary

As seen in Figure 9, the iris boundary is captured accurately. However, there is eyelid interference. This interference can be removed through further additions to the algorithm, but for this project, images with minimal interference are used. Figure 10 shows the circles determined for both pupil and iris segmentation in one image.

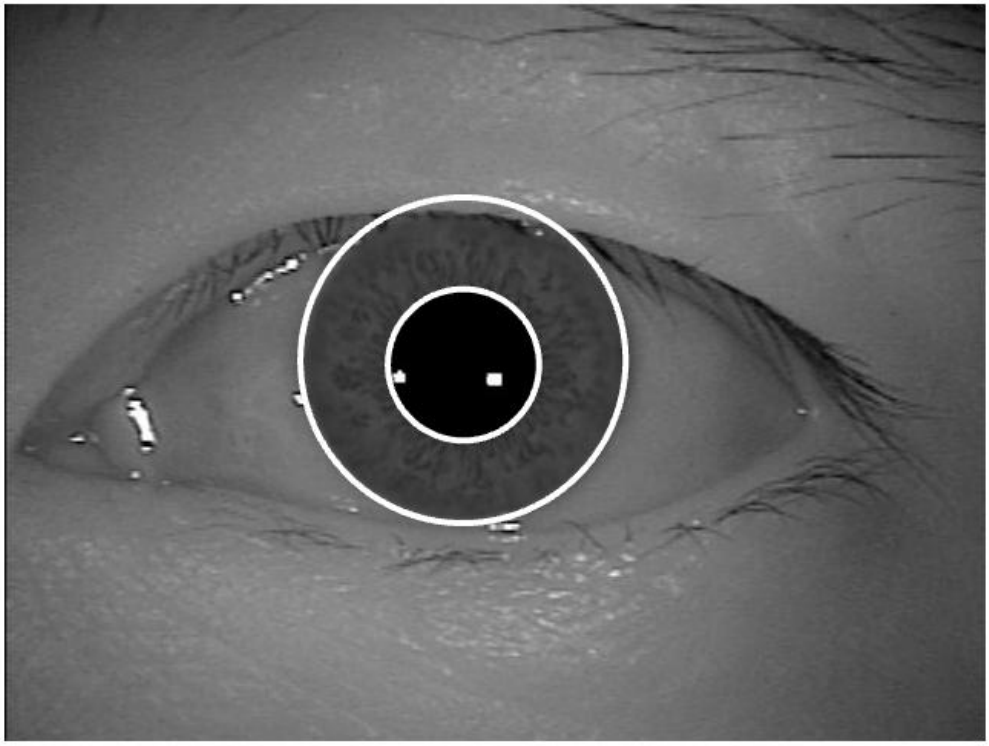

Figure 10: Iris Image with Pupil and Iris Boundary detected

Unwrap Iris

After the boundary detection procedure is finished, the pupil center coordinates, pupil radius, and iris radius are used to unwrap the iris into a rectangular shape [13]. The rectangular shape is obtained by finding Cartesian coordinates from the iris. First, pixel length of the iris is found by finding the difference between the iris radius and pupil radius. Then, that many circles are drawn starting at the pupil boundary, and ending at the outer boundary of the iris. The iris Cartesian coordinates are found by using the pupil's center coordinates and trigonometric identities to find the x and y values along each circle incrementing θ by 1° from 1° – 360°. The radius value (r) in the following equation is incremented in steps of 1 starting at the pupil radius value and ending at the iris radius value. This algorithm can be expressed with the following equations:

$$x_{iris} = x_{center} \pm r\cos(\frac{\theta\pi}{180})$$

$$y_{iris} = y_{center} \pm r\sin(\frac{\theta\pi}{180})$$

The addition and subtraction operators are used depending on which quadrant of the iris the coordinate is located in. For the x_{iris} coordinate, the addition operator is used when the coordinate resides in the first or fourth quadrants. For the y_{iris} coordinate, the addition operator is used when the coordinate resides in the third or fourth quadrants.

A visual aid for why the addition and subtraction operators are used in certain quadrants is shown in Figure 11. Note that the center of the iris shown is approximately at (300, 240). When the x_{iris} coordinate is in the first and fourth quadrants, the coordinate value exceeds 300 so x_{center} must be added to $rcos(\frac{\theta\pi}{180})$. The value resulting from $rcos(\frac{\theta\pi}{180})$ represents x_{iris}'s distance from x_{center}. The same logic applies when determining y_{iris}. When y_{iris} is in the third and fourth quadrants, it exceeds 240 so the addition operator is used.

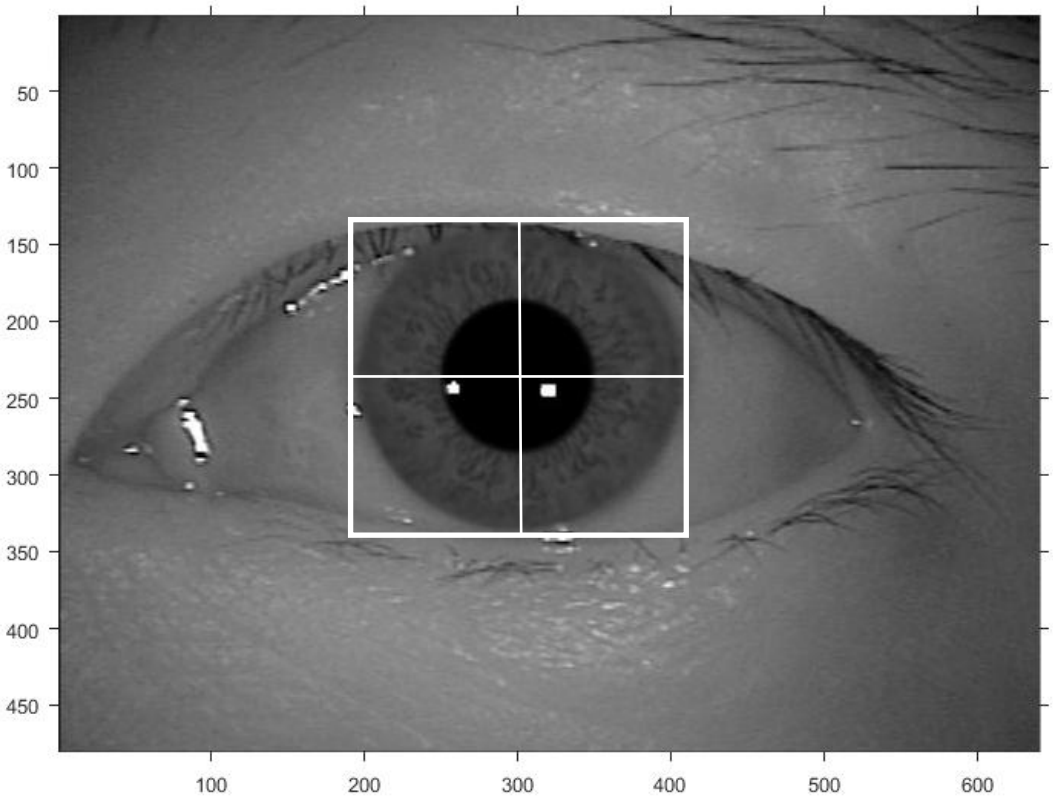

Figure 11: Iris Image with Axes displayed

These coordinate values are then used to find the corresponding pixel intensity values on the iris image. The unwrapped iris is obtained using these intensity values and assigning them to a new axis with angle value representing the horizontal axis, and the distance from the pupil boundary representing the vertical axis. Figure 12 shows the unwrapped image obtained with this algorithm.

Figure 12: Unwrapped Iris

Feature Extraction

The feature extraction portion of the project involves extracting useful information from the isolated iris to send to the artificial neural network. This useful information is the low-frequency coefficients of the wavelet transform.

The unwrapped image first goes through a histogram equalization process and canny edge detection process [10] so that the image contrast is improved. To perform histogram equalization, Matlab's histeq() function was used to produce the image shown in Figure 13.

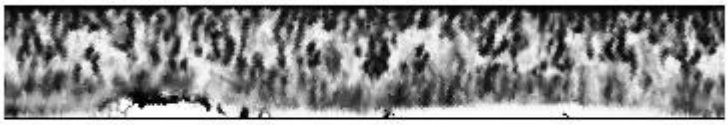

Figure 13: Histogram Equalized Unwrapped Iris

The input to the function is the unwrapped iris image from the previous steps. The histogram equalization process allows the intensity values associated with each pixel to be more pronounced. This effect is achieved by taking the lowest intensity value of an image and setting it to 0, and taking the highest intensity value of an image and setting it to 255. As a result, the intensities in the image used are more broadly distributed so that portions of the image with lower contrast now have higher contrast. To convert each pixel intensity to the equalized value, the image's cumulative distributive function (CDF) must be retrieved. The CDF can be described as the probability of a certain value to be equal to or less than itself in a population. The following equation shows how each pixel intensity is converted to the equalized value:

$$Equalized\ Value = round\left(\frac{CDF\ for\ pixel - \min CDF}{(rows\ x\ columns) - \min CDF}\ x\ (number\ of\ gray\ levels - 1)\right)$$

The canny edge detection process makes the image texture resulting from the equalization more distinct. This edge detection method is used because it has shown success in other iris recognition systems [13]. This method utilizes a several steps to show an image's edge features. With canny edge detection, a Gaussian filter is applied to the image to remove any noise. Next, the intensity gradient of the image is derived so that the orientation of the edges is found. These edges are then thinned by comparing them to surrounding intensity values in their corresponding gradient directions. If the value is larger than any

surrounding values, then that value is stored and the surrounding values are set to 0. Next, a low and high threshold value are defined. If any remaining intensity values are below the low threshold, then they are set to 0. If the value is higher than the low threshold, but lower than the high threshold then they are set as a weak edge pixel. If the value is higher than the high threshold, then it is set as a strong edge pixel [15].

To observe the edges using the canny edge detection method, Matlab's edge() function was used. The input to the function is the histogram equalized image, and the edge detection method is set to 'canny'. A comparison with the canny edge detection applied to the unwrapped iris before and after histogram equalization is shown in Figure 14. The wavelet transform is performed on this enhanced image.

Figure 14: Canny Edge Detection applied to Iris before Histogram Equalization (top) and after (bottom)

The 2-dimensional Haar wavelet transform method is used to extract the low-frequency coefficients. The discrete wavelet transform operates by passing a signal through a low-pass and high-pass filter to obtain approximation and detail coefficients respectively. This operation is observed in the following equation:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]g[n-k]$$

The coefficients are a result of the convolution of the signal and the filter. For iris recognition, the low-frequency coefficients are used as inputs for the classification process, so the approximation coefficients are retrieved after performing the transform using a low-pass filter. After each level of the transform, the approximation coefficients are sent as inputs to another decomposition level until 5 decompositions are performed. This number of wavelet decompositions was chosen because it provides enough coefficients for accurate classification without providing too many which would increase neural network training and testing time. If less decompositions were performed, the classification time would increase; if more decompositions were performed, there would be too few wavelet coefficients so there would not be enough data to accurately classify each iris.

A level 5 decomposition is performed on the unwrapped iris image, and the resulting low-frequency values of the 5th transform are used as inputs to the neural network. The unwrapped iris image's y-axis was typically in the range of 50 to 60 pixels. This value varies because the distance between the pupil boundary and iris boundary varies per image. With this range, the matrix resulting from 5 wavelet decompositions produces a low-frequency matrix of size 2x12. The resulting matrix used for the neural network inputs is taken from the upper left corner of the image after 5 decompositions. Matlab's dwt2() function is called 5 times to perform the level 5 decomposition. The input to the first function call is the enhanced unwrapped iris image, and the output is the low-frequency portion after the transform. For each successive call, the low-frequency portions are used as inputs. With an image dimension of 480 x 640, the number of inputs to the neural network is 24. Figure 15 provides a visual representation of how each

wavelet transform operation divides the image into low-frequency and high-frequency sections. The image shows a level 4 decomposition.

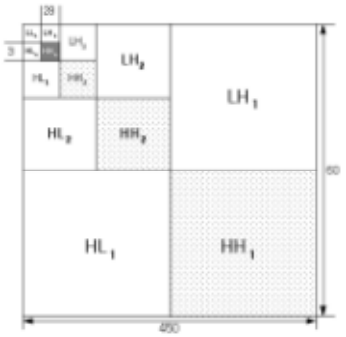

Figure 15: Decomposition of Image after Successive Haar Wavelet Transforms [11]

After obtaining the low-frequency coefficients, they are normalized so that each coefficient has some value between 0 and 1. This normalization is performed by dividing each coefficient by the maximum valued coefficient found as seen in the following equation:

$$Normalized\ Coefficient = \frac{Coefficient\ Value}{Max\ Coefficient\ Value}$$

Before normalization, the typical range of the coefficients is between approximately 2.75 and 8.15.

Neural Network Training

In this step, the normalized wavelet transform coefficients determined from the previous step are used to train the artificial neural network. The coefficients retrieved from all images are combined into one large matrix, and then used to train the neural network. The order of input/output pairs for the neural network inputs and targets were randomized using the randperm() function in Matlab. Figure 16 shows the block diagram for the classification system. The neural network is trained using the backpropagation method. In this method, the network determines the error rate between the actual output and desired output. This error rate is sent backwards into the hidden layer to re-calculate the weights which should result in an output with a lower error rate. To create the neural network, Matlab's feedforwardnet() function is used. The function is set to use the traincgb backpropagation method, and it is configured to use 10 nodes in the hidden layer. The traincgb backpropagation training method is used because it provides the greatest accuracy for a relatively low training duration [12].

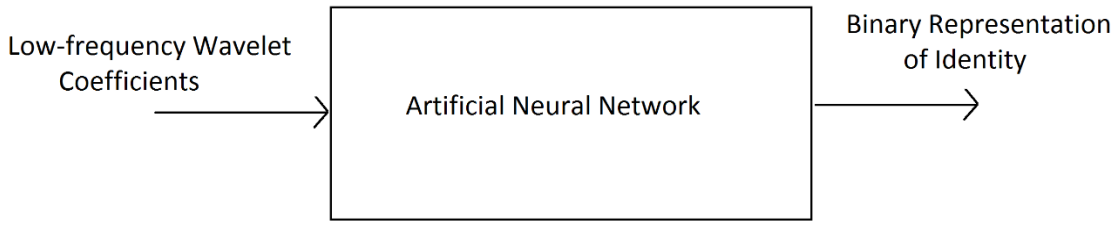

Figure 16: Classification System Block Diagram

The neural network is currently configured as shown in Figure 17.

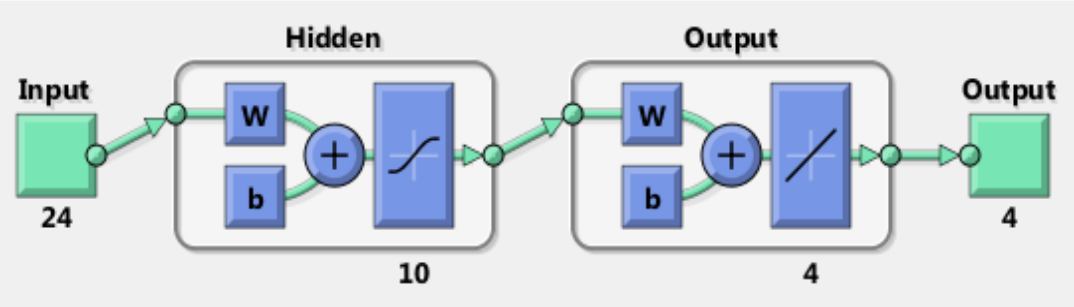

Figure 17: Neural Network Block Diagram

As shown in Figure 17, the input layer has 24 input nodes, and the output layer has 4 output nodes. The 4 output nodes are expected to be either 0 or 1. Together, they form a binary representation of the person identified by the neural network. The network is configured with a hidden layer containing 10 nodes.

The neural network was trained with coefficients from 5 iris images each from 10 different people. As outlined in the segmentation algorithm, each iris image input results in 100 modified iris images. As a result, the training set for each individual is a 24 x 500 matrix of low-frequency wavelet coefficients. In total, the neural network receives a 24 x 5000 input matrix for training for 10 people.

Chapter IV: Simulation Results

The neural network was tested with 5 images per person which differ from the 5 images used to train the neural network. Each image was rotated from a range of -10° to +10°. The angle of rotation is randomly generated using Matlab's rand() function. Each image is randomly rotated 10 times, which results in 50 modified images for testing per person. The coefficients from all 50 images were retrieved, and formed into a large matrix to test the neural network. Table III shows the target binary output by the neural network for each person. If the neural network output is not representative of any person's binary code, then the result is filed under unidentified as seen in Table IV. Table IV shows the results obtained with the testing data.

Table III: Binary Representation for each Person

Person	Binary Representation
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010

Table IV: Neural Network Classification Results (24 inputs)

		Output of Artificial Neural Network										
True Output		Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Person 7	Person 8	Person 9	Person 10	Unidentified
	Person 1	38	0	8	0	1	0	0	2	1	0	0
	Person 2	1	28	1	5	2	5	0	5	0	1	2
	Person 3	0	2	48	0	0	0	0	0	0	0	0
	Person 4	1	0	0	36	9	0	0	1	1	0	2
	Person 5	3	1	0	1	42	2	0	1	0	0	0
	Person 6	0	0	0	0	1	45	0	1	0	1	2
	Person 7	1	0	0	0	1	0	48	0	0	0	0
	Person 8	0	0	0	0	0	1	0	47	0	1	1
	Person 9	0	0	0	0	0	0	0	1	47	1	1
	Pers	0	0	0	0	1	0	0	1	0	48	0

	on 10											

For persons 1 through 10, their respective accuracies are 76%, 56%, 96%, 72%, 84%, 90%, 96%, 94%, 94%, and 96% with a total of 500 tested images. Total accuracy of this testing procedure is 85.4%. The authentication system was also tested using a matrix containing 16 low-frequency coefficients resulting from feature extraction. This results are shown in Table V below.

Table V: Neural Network Classification Results (16 inputs)

		Output of Artificial Neural Network										
True Output		Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Person 7	Person 8	Person 9	Person 10	Unidentified
	Person 1	41	2	2	0	2	0	0	2	0	0	1
	Person 2	0	33	2	5	1	0	0	0	0	9	0
	Person 3	1	1	48	0	0	0	0	0	0	0	0
	Person 4	1	3	1	33	4	0	1	2	2	3	0
	Person 5	4	1	1	3	34	3	2	1	0	0	1
	Person 6	0	0	0	2	2	44	1	0	0	1	0
	Person 7	0	0	2	0	1	0	47	0	0	0	0
	Person 8	4	1	1	1	0	1	0	32	1	5	4
	Person 9	3	0	0	0	0	1	0	3	42	0	1
	Person 10	0	4	1	1	3	1	2	0	0	37	1

For persons 1 through 10, their respective accuracies are 82%, 66%, 96%, 66%, 68%, 88%, 94%, 64%, 84%, and 74%. The total accuracy is 78.2% so testing with 16 coefficients showed lower accuracy when compared to 24 coefficients.

The neural network was also tested using a matrix containing 36 low-frequency coefficients. These coefficients were obtained by performing a level 4 wavelet decomposition on the unwrapped iris image because a level 5 decomposition only provided a maximum of 24 coefficients. From the resulting low-frequency matrix, 36 coefficients were used to train and test the neural network producing the results shown in Table VI.

Table VI: Neural Network Classification Results (36 inputs)

		Output of Artificial Neural Network										
True Output		Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Person 7	Person 8	Person 9	Person 10	Unidentified
	Person 1	33	1	4	1	0	0	0	3	8	0	0
	Person 2	3	32	1	0	1	2	1	4	0	5	1
	Person 3	10	0	30	0	5	0	2	1	0	0	2
	Person 4	1	0	1	43	5	0	0	0	0	0	0
	Person 5	11	0	8	0	27	0	3	1	0	0	0
	Person 6	3	0	1	3	1	39	1	1	1	0	0
	Person 7	0	0	3	0	0	7	32	0	0	1	7
	Person 8	5	2	2	0	1	2	0	32	0	3	3
	Person 9	0	2	1	0	0	5	0	0	39	0	3
	Person 10	0	0	1	0	1	0	2	0	0	30	16

For persons 1 through 10, their respective accuracies are 66%, 64%, 60%, 86%, 54%, 78%, 64%, 64%, 78%, and 60%. The total accuracy for 36 input testing is 67.4%. This is a noticeable drop in accuracy from both the 24 and 16 input test case, however, the 36 input test case should show an increase in accuracy because more iris information being used for classification. This drop in accuracy can be attributed to a situation which was noted when retrieving inputs for the neural network. When performing a level 4 wavelet decomposition, the resulting low-frequency matrix retrieved for each image were not of similar sizes. As a result, the 36 coefficients retrieved for each image were not consistently retrieved from the same locations of each matrix. The number of elements within the matrices would sometimes go above 90 coefficients.

As seen in the results for 24 and 16 input testing, there is a noticeable drop in accuracy for persons 2, 4, and 5. This can be attributed to several factors. The first is eyelid interference. For many of the iris images analyzed, there is visible interference within the images which is used as inputs to the neural network along with the vital iris coefficients. This will affect accuracy because coefficients obtained from eyelid regions do not contain relevant information for iris identification. This interference can be seen in Figure 18.

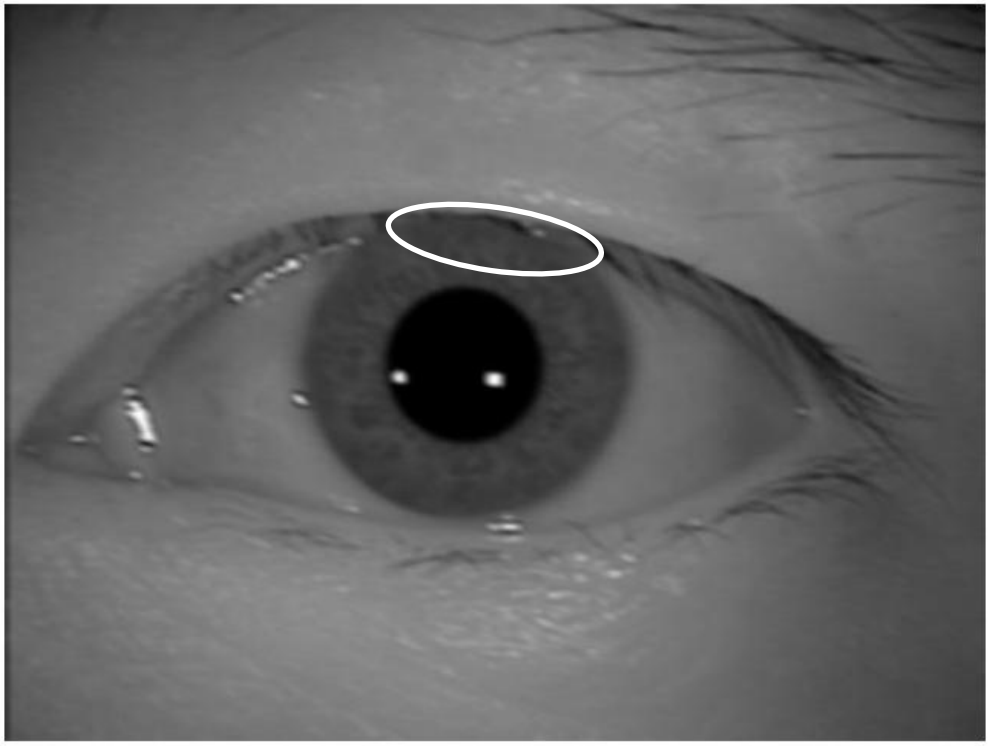

Figure 18: Iris Image showing Eyelid Interference

Another source of error may be within image compression. For the three persons, particularly person 2, there is a large variance in image quality used to train and test the neural network. As a result, some of the images have a blurred effect which makes defined iris features more difficult to extract. Figure 19 shows a comparison between two iris images which have different levels of image clarity.

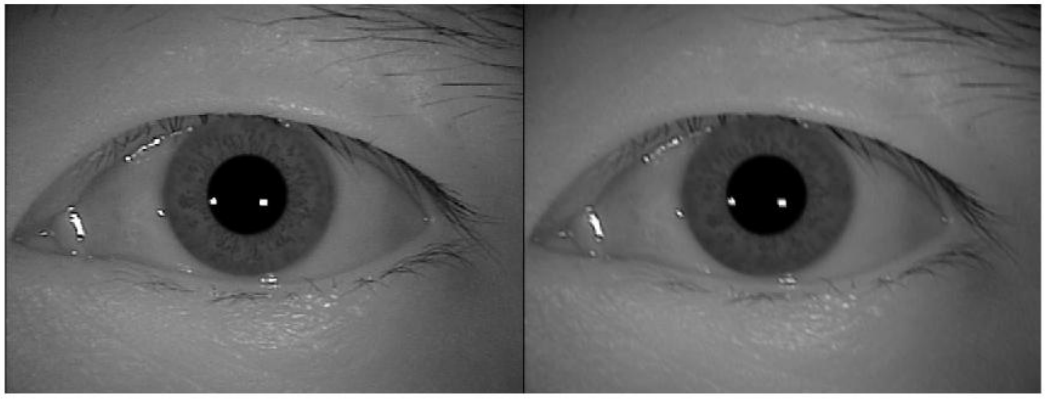

Figure 19: Comparison between Sharp (left) and Blurry (right) Iris Images

Chapter V: Conclusions and Future Work

The iris authentication system shows decent results. For 500 tested images, 406 were successfully verified. The authentication system is configured to display “Success” or “Failure” based on verification of the tested images.

The system consists of 3 sections: segmentation, feature extraction, and classification. Of the three sections which could receive improvement is segmentation. Currently, the segmentation algorithm assumes that an ideal iris image is used for authentication, however, many of the irises trained and tested had some interference by eyelash or eyelid. This interference shows in the unwrapping result provided, and it may have affected the coefficient values obtained from feature extraction. Using images from the CASIA database, finding irises with no object interference is very difficult since most of them have a little object overlap with the iris.

However, the iris authentication system shows that it is indeed an effective recognition system with the accuracy observed. There is some more work to be done to improve the accuracy, and expand the database. In conclusion, performing classification using iris wavelet coefficients as inputs shows that iris verification is successful, and will experience greater success when improvements are added.

There are several more tasks to complete which can improve the project. As of now, the system is trained and tested with 10 people, but a goal to work towards is to build a larger database of identifiable irises. For identification of more people, the binary representation at the output of the neural network may need to be increased in bit size. The pupil and iris boundary detection algorithm could be adjusted as well because performing a level 4 wavelet decomposition on a unwrapped iris image provided significantly lower classification accuracies.

Lastly, another step which is to be implemented is adding noise to the iris images, and testing the neural network with them. Noise can be added to an image using Matlab functions, and verifying that the authentication system stills works with high accuracy indicates a successful iris authentication system.

A. References

[1] L. W. Liam et al., “Iris Recognition Using Self-Organizing Neural Network,” in Research and Development, 2002 © IEEE. doi: 10.1109/SCORED.2002.1033084

[2] R. M. Farouk et al., “Iris Matching using Multi-Dimensional Artificial Neural Network,” in Computer Vision, IET, 2011 © IEEE. doi: 10.1049/iet-cvi.2010.0133

[3] C. Gerhensen, “Artificial Neural Networks for Beginners,” Cornell University Library. Ithaca, NY, Rep. arXiv:cs/0308031, Aug. 2003. Available: http://arxiv.org/pdf/cs/0308031.

[4] A. Nait-Ali and R. Fournier, “Signal and Image Processing for Biometrics,” Hoboken, NJ: John Wily & Sons, Inc., 2012.

[5] Y. Min et al., “Applications of Generalized Learning in Image Recognition,” in Neural Interface and Control, 2005 © IEEE. doi: 10.1109/ICNIC.2005.1499867

[6] H. Rashid et al., “Design Method of Video Based Iris Recognition System (V-IRS),” Universiti Kebangsan. Selangor, Malaysia, Nov. 2013.

[7] K. Kim et al., “Biometric Identification Apparatus and Method using Bio Signals and Artificial Neural Network,” U.S. Patent 7 630 521, December 8, 2009.

[8] P. Georgieva et al., “Advances in Intelligent Signal Processing and Data Mining,” Heidelberg, Germany: Springer, 2013.

[9] S. H. Chagas et al., “An Approach to Localization Scheme of Wireless Sensor Networks based on Artificial Neural Networks and Genetic Algorithms,” in New Circuits and Systems Conference, 2012 © IEEE. doi: 10.1109/NEWCAS.2012.6328975

[10] B. Kovoor et al., “Effectiveness of Feature Detection Operators on the Performance of Iris Biometric Recognition System,” International Journal of Network Security & Its Applications (IJNSA). Vol. 5, No. 5, September 2013.

[11] K. Lee et al., “Efficient Iris Recognition through Improvement of Feature Vector and Classifier,” ETRI Journal. Vol. 23, p. 61-70, 2001.

[12] U. Dias et al., “A Neural Network Based Iris Recognition System for Personal Identification,” ICTACT Journal on Soft Computing, Issue: 02, October 2010.

[13] B. Kovoor et al., “Iris Biometric Recognition System Employing Canny Operator,” Computer Science & Information Technology. pp. 65-74, 2013. @CS & IT-CSCP 2013. DOI: 10.5121/csit.2013.3307.

[14] M. Elgamal and N. Al-Biqami, “An Efficient Feature Extraction Method for Iris Recognition Based on Wavelet Transform,” International Journal of Computer and Information Technology. Vol. 2, Issue 03. May, 2013. ISSN: 2279-0764.

[15] Homepages.inf.ed.ac.uk, 'Feature Detectors - Canny Edge Detector', 2015. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm. [Accessed: 26- May- 2015].

[16] Biometrics.idealtest.org, 'Biometrics Ideal Test', 2015. [Online]. Available: http://biometrics.idealtest.org/dbDetailForUser.do?id=4. [Accessed: 01- Jun- 2015].

B. Senior Project Analysis

- The iris identification system receives an inputted image which it identifies and extracts key features from. The system processes the image's features through a classification process where it compares common and uncommon aspects to images used to train the neural network. This final comparison determines whether or not a match in irises exists.

- An associated difficulty involves accounting external variables regarding the eye such as eyelashes, eyelids, and lighting variances. These variables affect the identification process so they need accounting for within the code so they do not impact accuracy.

- The project involves various economic impacts. For human capital, the knowledge gathered regarding artificial neural networks and iris identification processes should help build the device. The knowledge gained from this project can pass between individuals. For financial capital, necessary labor costs and material costs arise. Regarding manufactured capital, the project needs certain tools and technology ensuring completion, such as MATLAB.

  Costs and benefits accrue during various parts of the project's lifecycle. Costs accrue during development because of labor and acquiring software to develop the project. A more in depth explanation of the costs and their quantification receive explanation below and in Table VII. Benefits accrue after the software development completion, and the project performs its functional requirements. A benefit accrued includes the ability to manufacture the system. This naturally benefits the laborers because it returns the invested hours and component costs required to complete the project. Another benefit accrued includes the fact that the goal reached completion. This indicates the project functions successfully.

  The project requires few inputs: the inputted image and computer power. The powered computer should analyze the image so a comparison results between it and an ideal image [2]. The total estimated cost of parts resides around $150. This includes a $99 MATLAB and additional toolboxes which may include an additional $50. MATLAB approximates to $99 because the software provider provides the $99 student version. The computing processes the system performs receive instructions from the created code in MATLAB. Table VII includes these cost estimations, and their justifications, in an easily readable format.

Table VII: Cost Estimations Table

Part	Cost	Justification
MATLAB	$99 + ~$50 additional toolboxes	MATLAB provides the method which the programmed system functions. The code enables the image comparison functionality. The student MATLAB version costs $99, so $99 proves a reasonable estimate.
Labor	$30/hr	The labor cost should estimate around $30/hr because this wage proves somewhat standard regarding various engineering professions. The estimated time spent working on this project approximates to 120 hours. This would indicate a total $3600 cost for labor.

Products emerge once software development finishes and the technology fully implements itself into the product. The product comes to fruition providing the technology produces accurate

results. Development time approximates around 5 months, providing an estimated 100 hour work time. Figure 20 shows the project task schedule over the course of 10 months. The sum of these tasks falls in accordance with the 100 hour labor estimation.

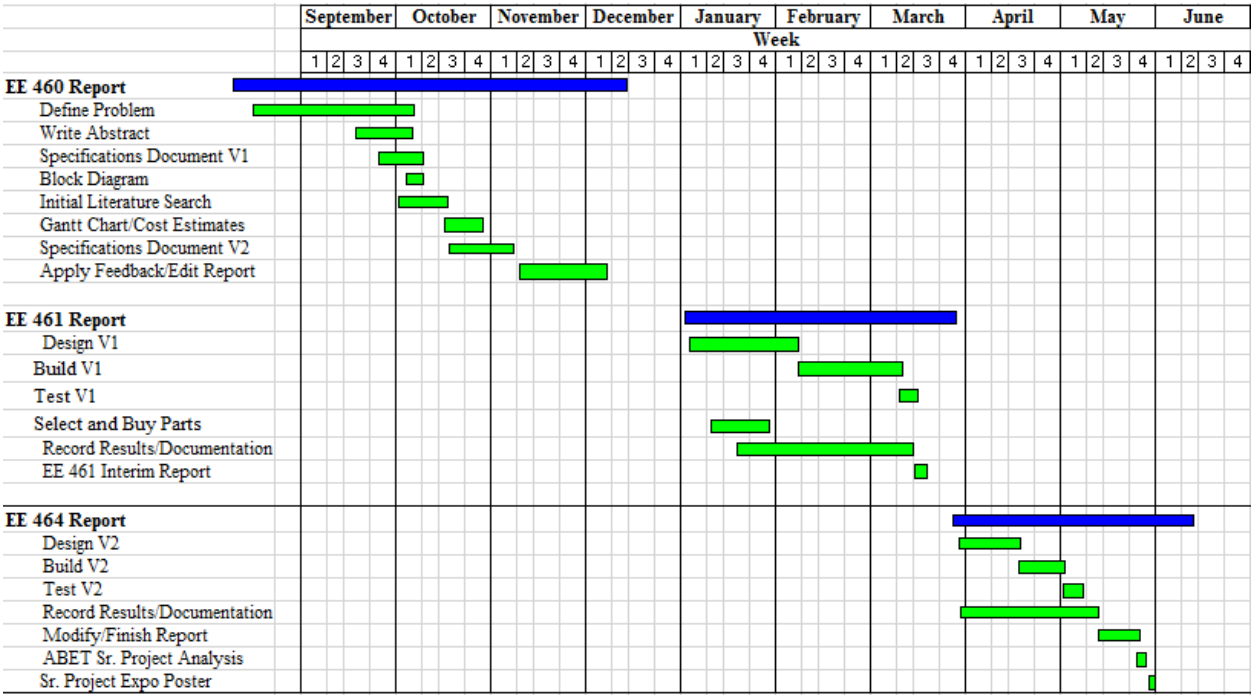

Figure 20: Gantt Chart

- The project does not necessarily improve or affect any natural resources because the sole source of power required is for powering a computer to run the software. The amount of energy used should not be substantial enough to have any significant effects on the environment.

- A manufacturing challenge associated involves building the full device using all the components, and ensuring bugs do not exist. Another manufacturing challenge includes software installation so the full device works. With device mass production, there exist issues which arise which require testing so user experience remains easy. These issues include incorrect software installation within the device.

- A system maintenance issue involves a potential camera's capturing capabilities. Over a long time period, the camera may no longer suit the image capturing process since it becomes outdated and perhaps vulnerable to malfunction. Another system maintenance issue involves the power source. The project needs a power source during operation so power must always exist. The project does not completely adhere to sustainable resource use because of material deterioration with camera usage. However, since the project remains software based, this portion should not consume resources. This reveals a possible upgrade as well.

- An iris identification technology application involves securing prized belongings so there exists no risk of theft. Under this use, the project implies psychological egoism, so every individual acts in their own self-interest. Typically, entities which use these technologies have something they need to keep secure. This framework is ethical because there exists an interest in keeping property safe, and away from malicious deeds.

  Another similar technological application involves security services such as keeping unwanted people out of certain areas. In these situations, the ethical framework, Utilitarianism, applies. This framework is relevant because the technology's implementation allows the safety for large groups of people which complies with Utilitarianism's "greatest good for the greatest number". There exists a common interest in treating everyone equally, and making decisions so the public's safety receives attention.

- A product safety concern involves the iris image capturing process. Concern arises through the image capturing method, such as implementation of extraneous lighting and how the lighting affects their eyes. Lighting sources prove necessary because darker images can be more difficult to classify.

- A social and political issue associated with project usage involves the sentiment that other security measures are adequate enough, such as fingerprint scanning. Some people may feel unwilling having such a vulnerable body part, their eye, used for identification processes.

  The project directly impacts those who desire iris identification technology for protection purposes. The indirect stakeholders identify as other teams who research iris identification technology. The project benefits stakeholders through proper iris identification to enhance security measures. The project harms stakeholders through providing inaccurate results. The stakeholders should all benefit equally if the project proves successful.

- A technique useful for the project involves implementing artificial neural networks through MATLAB. This serves as the project's basis and enables the system to perform the necessary calculations required for iris recognition.

Segmentation Code

```
function [radius_iris,radius_pupil,x_center,y_center,rotate1,rotate2,...
    rotate3,rotate4,rotate5,rotate6,rotate7,rotate8,rotate9,rotate10,...
  rotate11,rotate12,rotate13,rotate14,rotate15,rotate16,rotate17,rotate18,...
  rotate19,rotate20,rotate21,rotate22,rotate23,rotate24,rotate25,rotate26,...
  rotate27,rotate28,rotate29,rotate30,rotate31,rotate32,rotate33,rotate34,...
  rotate35,rotate36,rotate37,rotate38,rotate39,rotate40,rotate41,rotate42,...
  rotate43,rotate44,rotate45,rotate46,rotate47,rotate48,rotate49,rotate50,...
  rotate51,rotate52,rotate53,rotate54,rotate55,rotate56,rotate57,rotate58,...
  rotate59,rotate60,rotate61,rotate62,rotate63,rotate64,rotate65,rotate66,...
  rotate67,rotate68,rotate69,rotate70,rotate71,rotate72,rotate73,rotate74,...
  rotate75,rotate76,rotate77,rotate78,rotate79,rotate80,rotate81,rotate82,...
  rotate83,rotate84,rotate85,rotate86,rotate87,rotate88,rotate89,rotate90,...
  rotate91,rotate92,rotate93,rotate94,rotate95,rotate96,rotate97,rotate98,...
    rotate99,rotate100] = seg(iris_image)

%convert image to grayscale
dims = ndims(iris_image);

if dims > 2

    iris_image = rgb2gray(iris_image);

end

%find image dimensions
[M,N] = size(iris_image);

for i = 1:100
    angle_rot(i,1) = 10 .* rand(1);
    if i > 50
        angle_rot(i,1) = -1 * angle_rot(i,1);
    end
end

%find pupil and iris boundaries
for i = 1:100
    rotate = imrotate(iris_image,angle_rot(i,1),'crop');
    [radius_pupil(i,:),x_center(i,:),y_center(i,:),pupil_center(i,:)] =
pupil_boundary(rotate);
    [radius_iris(i,:),iris_center(i,:)] =
iris_boundary(rotate,radius_pupil(i,:));
end

rotate1 = imrotate(iris_image,angle_rot(1,1),'crop');
rotate2 = imrotate(iris_image,angle_rot(2,1),'crop');
rotate3 = imrotate(iris_image,angle_rot(3,1),'crop');
rotate4 = imrotate(iris_image,angle_rot(4,1),'crop');
rotate5 = imrotate(iris_image,angle_rot(5,1),'crop');
```

```
rotate6 = imrotate(iris_image,angle_rot(6,1),'crop');
rotate7 = imrotate(iris_image,angle_rot(7,1),'crop');
rotate8 = imrotate(iris_image,angle_rot(8,1),'crop');
rotate9 = imrotate(iris_image,angle_rot(9,1),'crop');
rotate10 = imrotate(iris_image,angle_rot(10,1),'crop');
rotate11 = imrotate(iris_image,angle_rot(11,1),'crop');
rotate12 = imrotate(iris_image,angle_rot(12,1),'crop');
rotate13 = imrotate(iris_image,angle_rot(13,1),'crop');
rotate14 = imrotate(iris_image,angle_rot(14,1),'crop');
rotate15 = imrotate(iris_image,angle_rot(15,1),'crop');
rotate16 = imrotate(iris_image,angle_rot(16,1),'crop');
rotate17 = imrotate(iris_image,angle_rot(17,1),'crop');
rotate18 = imrotate(iris_image,angle_rot(18,1),'crop');
rotate19 = imrotate(iris_image,angle_rot(19,1),'crop');
rotate20 = imrotate(iris_image,angle_rot(20,1),'crop');
rotate21 = imrotate(iris_image,angle_rot(21,1),'crop');
rotate22 = imrotate(iris_image,angle_rot(22,1),'crop');
rotate23 = imrotate(iris_image,angle_rot(23,1),'crop');
rotate24 = imrotate(iris_image,angle_rot(24,1),'crop');
rotate25 = imrotate(iris_image,angle_rot(25,1),'crop');
rotate26 = imrotate(iris_image,angle_rot(26,1),'crop');
rotate27 = imrotate(iris_image,angle_rot(27,1),'crop');
rotate28 = imrotate(iris_image,angle_rot(28,1),'crop');
rotate29 = imrotate(iris_image,angle_rot(29,1),'crop');
rotate30 = imrotate(iris_image,angle_rot(30,1),'crop');
rotate31 = imrotate(iris_image,angle_rot(31,1),'crop');
rotate32 = imrotate(iris_image,angle_rot(32,1),'crop');
rotate33 = imrotate(iris_image,angle_rot(33,1),'crop');
rotate34 = imrotate(iris_image,angle_rot(34,1),'crop');
rotate35 = imrotate(iris_image,angle_rot(35,1),'crop');
rotate36 = imrotate(iris_image,angle_rot(36,1),'crop');
rotate37 = imrotate(iris_image,angle_rot(37,1),'crop');
rotate38 = imrotate(iris_image,angle_rot(38,1),'crop');
rotate39 = imrotate(iris_image,angle_rot(39,1),'crop');
rotate40 = imrotate(iris_image,angle_rot(40,1),'crop');
rotate41 = imrotate(iris_image,angle_rot(41,1),'crop');
rotate42 = imrotate(iris_image,angle_rot(42,1),'crop');
rotate43 = imrotate(iris_image,angle_rot(43,1),'crop');
rotate44 = imrotate(iris_image,angle_rot(44,1),'crop');
rotate45 = imrotate(iris_image,angle_rot(45,1),'crop');
rotate46 = imrotate(iris_image,angle_rot(46,1),'crop');
rotate47 = imrotate(iris_image,angle_rot(47,1),'crop');
rotate48 = imrotate(iris_image,angle_rot(48,1),'crop');
rotate49 = imrotate(iris_image,angle_rot(49,1),'crop');
rotate50 = imrotate(iris_image,angle_rot(50,1),'crop');
rotate51 = imrotate(iris_image,angle_rot(51,1),'crop');
rotate52 = imrotate(iris_image,angle_rot(52,1),'crop');
rotate53 = imrotate(iris_image,angle_rot(53,1),'crop');
rotate54 = imrotate(iris_image,angle_rot(54,1),'crop');
rotate55 = imrotate(iris_image,angle_rot(55,1),'crop');
rotate56 = imrotate(iris_image,angle_rot(56,1),'crop');
rotate57 = imrotate(iris_image,angle_rot(57,1),'crop');
rotate58 = imrotate(iris_image,angle_rot(58,1),'crop');
rotate59 = imrotate(iris_image,angle_rot(59,1),'crop');
rotate60 = imrotate(iris_image,angle_rot(60,1),'crop');
rotate61 = imrotate(iris_image,angle_rot(61,1),'crop');
```

```
rotate62 = imrotate(iris_image,angle_rot(62,1),'crop');
rotate63 = imrotate(iris_image,angle_rot(63,1),'crop');
rotate64 = imrotate(iris_image,angle_rot(64,1),'crop');
rotate65 = imrotate(iris_image,angle_rot(65,1),'crop');
rotate66 = imrotate(iris_image,angle_rot(66,1),'crop');
rotate67 = imrotate(iris_image,angle_rot(67,1),'crop');
rotate68 = imrotate(iris_image,angle_rot(68,1),'crop');
rotate69 = imrotate(iris_image,angle_rot(69,1),'crop');
rotate70 = imrotate(iris_image,angle_rot(70,1),'crop');
rotate71 = imrotate(iris_image,angle_rot(71,1),'crop');
rotate72 = imrotate(iris_image,angle_rot(72,1),'crop');
rotate73 = imrotate(iris_image,angle_rot(73,1),'crop');
rotate74 = imrotate(iris_image,angle_rot(74,1),'crop');
rotate75 = imrotate(iris_image,angle_rot(75,1),'crop');
rotate76 = imrotate(iris_image,angle_rot(76,1),'crop');
rotate77 = imrotate(iris_image,angle_rot(77,1),'crop');
rotate78 = imrotate(iris_image,angle_rot(78,1),'crop');
rotate79 = imrotate(iris_image,angle_rot(79,1),'crop');
rotate80 = imrotate(iris_image,angle_rot(80,1),'crop');
rotate81 = imrotate(iris_image,angle_rot(81,1),'crop');
rotate82 = imrotate(iris_image,angle_rot(82,1),'crop');
rotate83 = imrotate(iris_image,angle_rot(83,1),'crop');
rotate84 = imrotate(iris_image,angle_rot(84,1),'crop');
rotate85 = imrotate(iris_image,angle_rot(85,1),'crop');
rotate86 = imrotate(iris_image,angle_rot(86,1),'crop');
rotate87 = imrotate(iris_image,angle_rot(87,1),'crop');
rotate88 = imrotate(iris_image,angle_rot(88,1),'crop');
rotate89 = imrotate(iris_image,angle_rot(89,1),'crop');
rotate90 = imrotate(iris_image,angle_rot(90,1),'crop');
rotate91 = imrotate(iris_image,angle_rot(91,1),'crop');
rotate92 = imrotate(iris_image,angle_rot(92,1),'crop');
rotate93 = imrotate(iris_image,angle_rot(93,1),'crop');
rotate94 = imrotate(iris_image,angle_rot(94,1),'crop');
rotate95 = imrotate(iris_image,angle_rot(95,1),'crop');
rotate96 = imrotate(iris_image,angle_rot(96,1),'crop');
rotate97 = imrotate(iris_image,angle_rot(97,1),'crop');
rotate98 = imrotate(iris_image,angle_rot(98,1),'crop');
rotate99 = imrotate(iris_image,angle_rot(99,1),'crop');
rotate100 = imrotate(iris_image,angle_rot(100,1),'crop');

return
```

Find Pupil Boundary Code

```
function [radius_pupil,x_center,y_center,pupil_center] = pupil_boundary(iris_image)

[pupil_center,radius_pupil] = imfindcircles(iris_image,[35 75],...
   'ObjectPolarity','dark','Sensitivity',0.95,'EdgeThreshold',0.05);
pupil_center = pupil_center(1,:);
x_center = pupil_center(1,1);
y_center = pupil_center(1,2);
radius_pupil = radius_pupil(1);
```

```
end
```

Find Iris Boundary Code

```
function [radius_iris,iris_center] = iris_boundary(iris_image,radius_pupil)

[center,radius] = imfindcircles(iris_image,[70 130],...
   'ObjectPolarity','dark','Sensitivity',0.97,'EdgeThreshold',0.005);
[M,N] = size(radius);
j = zeros(M,N);
num = numel(radius);
radius_iris = radius(1);
iris_center = center(1,:);

r = radius_iris - radius_pupil;
n = 1;

if r < 33
   for i = 1:num
      k = radius(i) - radius_pupil;
      if k > 32
         j(n) = n + 1;
         n = n + 1;
      end
   end
   radius_iris = radius(j(1));
   iris_center = center(j(1),:);
end

end
```

Unwrap Isolated Iris Code

```
function [intensity] = unwrap(x_center,y_center,radius_pupil,radius_iris,iris_image)

iris_length = radius_iris - radius_pupil;

i = 1;
%smallest circle values
for circle_radius = ceil(radius_pupil):floor(radius_iris)
   for angle = 1:360
      x(i,angle) = abs(circle_radius.*cos(angle*pi/180));
      y(i,angle) = abs(circle_radius.*sin(angle*pi/180));
   end
   i = i + 1;
end

for angle = 1:360
   if angle < 90 || angle > 270
      x(:,angle) = round(x_center + x(:,angle));
   else
```

```
        x(:,angle) = round(x_center - x(:,angle));
    end
end

for angle = 1:360
    if angle > 180
        y(:,angle) = round(y_center + y(:,angle));
    else
        y(:,angle) = round(y_center - y(:,angle));
    end
end

[M,N] = size(y);
intensity = zeros(M,N);

for j = 1:iris_length
    for k = 1:360
        intensity(j,k) = iris_image(y(j,k),x(j,k));
    end
end

intensity = uint8(intensity);
%figure(2)
%imshow(intensity)

end
```

Feature Extraction Code

```
function [coefficients] = feature_extraction(intensity)

%enhance image
contrast = histeq(intensity);
texture = edge(contrast,'canny');

%perform Haar wavelet transform
[LL1,~,~,~] = dwt2(texture,'Haar');
[LL2,~,~,~] = dwt2(LL1,'Haar');
[LL3,~,~,~] = dwt2(LL2,'Haar');
[LL4,~,~,~] = dwt2(LL3,'Haar');
[LL5,~,~,~] = dwt2(LL4,'Haar');

rows = numel(LL5);
arr = zeros(rows,1);

%organize low frequency values into vector array
[M,N] = size(LL5);
k = 1;
for i = 1:M
    for j = 1:N
        arr(k) = LL5(i,j);
        k = k +1;
```

```
   end
end

%low frequency values set between [0:1]
maximum = max(arr);
coefficients = arr./maximum;

end
```

Neural Network Training Code

```
function [net,tr] = ann_train(x_train,target)

%prepare inputs/target
%x_data = x_data.';
%target_data = target_data.';

%create network
%net = newff(x_data,target_data,10,{},'traincgb');
net = feedforwardnet(10,'traincgb');
%view(net)

%set error goal for feedback
error = 0.01;
net.trainParam.goal = error;
net.trainParam.max_fail = 100;
%net.trainParam.epochs = 10000;
shuffle = randperm(size(x_train,2));
x_train_shuffle = x_train(:,[shuffle]);
target_shuffle = target(:,[shuffle]);

[net,tr] = train(net,x_train_shuffle,target_shuffle);
nntraintool

%check training outputs
%train_out = net(x_train);
%target_length = length(target);
%num_correct = 0;
%for i = 1:target_length
%    check(i) = train_out(i) - target(i);
%    if check(i) == 0
%        num_correct = num_correct + 1;
%    end
%end

%percentage of correct training outputs
%percent_correct = num_correct ./ target_length;

%extra calcs
%outputs = net(x_train);
%error = outputs - target;
%perf = perform(net,outputs,target);
```

Neural Network Testing Code

```
function [data,one,two,three,four,five,six,...
    seven,eight,nine,ten] = ann_test(net,test,target)

shuffle = randperm(size(test,2));
test_shuffle = test(:,[shuffle]);
target_shuffle = target(:,[shuffle]);

j = 0;

results = round(net(test_shuffle));

for i = 1:2000
    if results(i) < 0
        results(i) = 0;
    end
    if results(i) > 1
        results(i) = 1;
    end
end

for i = 1:500
    if results(1,i) ~= target_shuffle(1,i) || ...
            results(2,i) ~= target_shuffle(2,i) || ...
            results(3,i) ~= target_shuffle(3,i) || ...
            results(4,i) ~= target_shuffle(4,i)
        j = j + 1;
        incorrect(:,j) = results(:,i);
        real_target(:,j) = target_shuffle(:,i);
    end
end

one = 0;
two = 0;
three = 0;
four = 0;
five = 0;
six = 0;
seven = 0;
eight = 0;
nine = 0;
ten = 0;

for i = 1:j
    if incorrect(:,i) == [0;0;0;1]
        misidentified_as(:,i) = 1;
    elseif incorrect(:,i) == [0;0;1;0]
        misidentified_as(:,i) = 2;
    elseif incorrect(:,i) == [0;0;1;1]
        misidentified_as(:,i) = 3;
    elseif incorrect(:,i) == [0;1;0;0]
        misidentified_as(:,i) = 4;
```

```
    elseif incorrect(:,i) == [0;1;0;1]
        misidentified_as(:,i) = 5;
    elseif incorrect(:,i) == [0;1;1;0]
        misidentified_as(:,i) = 6;
    elseif incorrect(:,i) == [0;1;1;1]
        misidentified_as(:,i) = 7;
    elseif incorrect(:,i) == [1;0;0;0]
        misidentified_as(:,i) = 8;
    elseif incorrect(:,i) == [1;0;0;1]
        misidentified_as(:,i) = 9;
    elseif incorrect(:,i) == [1;0;1;0]
        misidentified_as(:,i) = 10;
    else
        misidentified_as(:,i) = 0;
    end
end

for i = 1:j
    if real_target(:,i) == [0;0;0;1]
        should_be(:,i) = 1;
        one = one + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [0;0;1;0]
        should_be(:,i) = 2;
        two = two + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [0;0;1;1]
        should_be(:,i) = 3;
        three = three + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [0;1;0;0]
        should_be(:,i) = 4;
        four = four + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [0;1;0;1]
        should_be(:,i) = 5;
        five = five + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [0;1;1;0]
        should_be(:,i) = 6;
        six = six + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [0;1;1;1]
        should_be(:,i) = 7;
        seven = seven + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [1;0;0;0]
        should_be(:,i) = 8;
        eight = eight + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [1;0;0;1]
        should_be(:,i) = 9;
        nine = nine + 1;
        wrong(:,i) = misidentified_as(:,i);
    elseif real_target(:,i) == [1;0;1;0]
        should_be(:,i) = 10;
```

```
        ten = ten + 1;
        wrong(:,i) = misidentified_as(:,i);
    end
end

data = [should_be; wrong];

end
```