**Iris Recognition Using Gray Level Co-occurrence Matrix Based Features and Classifier Fusion**

An Undergraduate Thesis

Presented to

The Faculty of the

School of Electrical Engineering and Applied Science

University of Virginia

In Partial Fulfillment

Of the Requirements for the Degree

Bachelor of Science in Electrical Engineering

By

**Sattyik Kundu**

**April 9, 2012**

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis Related Assignments

Signed Date

Approved \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Scott T. Acton, Thesis Advisor

**Executive Summary**

Finding robust feature extraction techniques capable of handling degraded iris images is an important research area of iris recognition. In this thesis, the application of GLCM (**Gray Level** **Co-occurrence Matrix)** toiris recognition has been described. Iris images considered have little or no occlusion but can be degraded due to lack of focus, poor contrast or other image impairment. Using 51 classes, less than 0.5% incorrect recognition is achieved in the best case scenario which is achieved using a multi-classifier scheme. This result is better than all other results reported in the literature using GLCM based features. Thus, multi-classifier recognition scheme using GLCM based features holds promise for iris recognition. Since many images in the experiments especially from UBIRIS v.1 databases are of poor quality iris images, experimental performance indicates that GLCM based features are robust for iris recognition.

**Acknowledgement**

I would like to sincerely acknowledge the help, support and understanding that I received from my thesis advisor, Prof. Scott T. Acton. I would also like to acknowledge the help and support I have received from Suvodip Mukherjee. I would also like to thank my family for their continuous support.

**Table of Contents**

1. Introduction 7
2. Proposed Iris Recognition Scheme 12
3. Experimental Results 28
4. Conclusions 35
5. References 37
6. Appendix: MATLAB and GLCM Features 42

**List of Figures**

Figure 1: Different Biometric Authentication Techniques 7

Figure 2: An eye image with marked iris area 8

Figure 3: Iris vs. pupil 9

Figure 4: Iris Recognition System 11

Figure 5**:** Four directions of adjacency for calculation of Haralick texture 14

Figure 6: Multi-Layer perceptron (MLP) neural network with 3 layers 18

Figure 7(a): Support Vector Machine 19

Figure 7(b): A Support Vector Machine example 20

Figure 7(c): Support vectors of a simple Support Vector Machine example 21

Figure 8: Schematic diagram of proposed iris recognition scheme 23

Figure 9: Left – An iris image, Right – Segmented circular iris outlined 24

Figure 10: Failure of iris segmentation algorithm 24

Figure 11: A perfectly segmented iris 25

Figure 12: Conversion of annular iris image to a rectangular iris image 25

Figure 13: Rectangular iris image of original iris image given by Fig. 11 26

**List of Tables**

Table 1: Selected classes from two different iris databases – UBIRIS.V1 and CASIS IRIS LAMP 29

Table 2: Experimental classification result using 28 classes from UBIRIS.v1 iris database

30

Table 3: Experimental classification result using 23 classes from CASIA IRIS LAMP

Database 31

Table 4: Experimental classification result using 28 classes from UBIRIS V.1 iris

database, and 23 classes from CASIA IRIS LAMP Database 32

Table 5: Results of three trials using multi-classifier fusion 34

Table 6: Consolidated results of three trials using multi-classifier fusion 34

1. INTRODUCTION

THE automated use of physiological or behavioral characteristics to determine or verify identity is called the biometric authentication technique.



**Fingerprint**



**Face**



**Signature**



**Iris**



**Voice**



**Hand geometry**

Figure 1: Different Biometric Authentication Techniques

There are a number of biometric authentication techniques as listed below.

1. Fingerprint identification that identifies a person by his/her fingerprints.
2. Face recognition that identifies a person by his/her 2-D or 3-D face image.
3. Iris recognition that identifies a person by his/her left or right iris image.
4. Signature identification that identifies a person by his/her signature.
5. Hand geometry identification that identifies a person by his/her hand geometry.
6. Voice recognition that identifies a person by his/her voice.
7. Other biometrics that include ear biometric, handwriting biometric etc.

The focus of this thesis is iris recognition. The iris is an internal organ of the eye that is located just behind the cornea and in front of the lens. Its function is to control the size of the pupil, which in turn regulates the amount of light entering the pupil and impinging the retina. Flom and Safir [1] have postulated that “the basic, significant features of the iris remain extremely stable and do not change over a period of many years” (this claim has been challenged in the recent literature). They state that every iris is unique and no two individuals have the same iris compositions. Indeed, the two irises of an individual have been observed to be different in their intricate texture structure. Hence, the iris is considered to be a robust and unique biometric with a very low False Acceptance Rate (FAR). Large-scale authentication experiments have confirmed this notion further underscoring the relevance of this biometric trait in distinguishing individuals [2], [3].

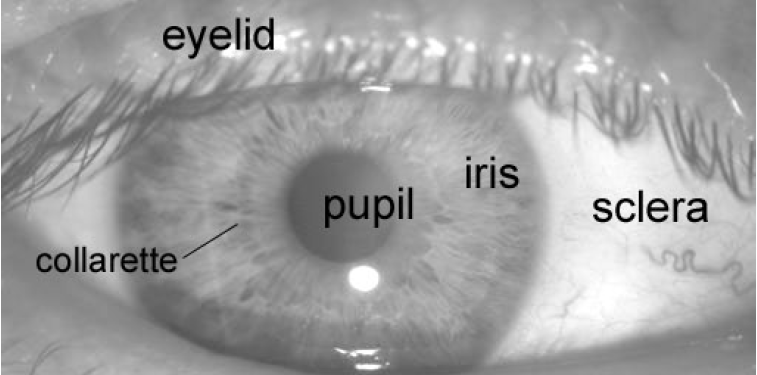


Figure 2: An eye image with marked iris area

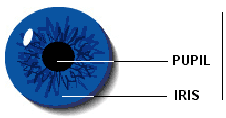


Figure 3: Iris vs. pupil

The function of an iris recognition system is to extract, represent and compare the textural intricacy present on the surface of the iris. Such a system comprises of modules for iris pre-processing that does segmentation and enhancement, feature extraction (encoding) and feature matching (Fig. 4). The first and, perhaps, the most important step in an iris recognition system is iris segmentation or localization. Segmentation involves detecting and isolating the iris structure from an image of the eye. As seen in Fig. 2, the iris projected onto a 2-D plane appears to be located in the vicinity of the sclera, pupil, and eyelids. Thus, the segmentation process has to accurately detect boundaries separating the iris from these components. Apart from estimating the actual shape of the iris, the segmentation routine should detect occlusions due to eyelashes that can confound the extracted features. Errors in segmentation may result in inferior recognition performance due to inaccurate encoding of the textural content of the iris. Several iris recognition algorithms have been proposed in the literature. Daugman [4], [5] uses a texture-based method to encode irises. Multi-scale 2-D Gabor-Wavelet transform is used to generate a 256-byte iriscode. Hamming distance is then used as a measure to determine the proximity of two iriscodes. The integro-differential operator, which acts as a circular edge detector, is employed for determining the inner and outer boundaries of the iris as well as the upper and lower eyelids. Wildes [6] uses Laplacian-of-a-Gaussian (LOG) filter to extract features from the iris image. A Hough transform-based method is used to segment the iris. Also, the upper and lower boundaries of the eyelid are approximated using parabolic curves. Matching is done using the normalized correlation between the test and training images. Masek and Kovesi [7] employ weighted gradients using a combination of Kovesi’s modified canny edge detector and the circular Hough-transform to segment the iris. Several other segmentation schemes proposed in the literature are also based on the Hough-transform (see, for example, [8]–[14]). Huang *et al.* [15] first coarsely segment the iris using edge detection filters and Hough transform before normalizing it. The noise due to eyelids is then localized by the edge information based on phase congruency. It should be noted that most segmentation models in the literature assume that the pupillary, the limbic, and the eyelid boundaries are circular or elliptical in shape. Hence, they focus on determining model parameters that best fit these hypotheses ([4], [6], [10]). So far, very few algorithms that do not assume circular or elliptical boundaries have been reported in the literature (e.g., see Abhyankar and Schuckers [16] and Daugman [3]). In this context, the application of geodesic active contour (GAC)-based scheme is considered [16] to accurately determine the boundary of the iris thereby overcoming any assumption about the shape of iris boundary.

It has been shown in [19-25] that the matching performance of iris recognition systems can be improved when quality is considered. Daugman applied the 2-D Focus Assessment to eliminate the out-of-focus iris images [19]. Ma *et al*. [20] used the frequency energy distribution to classify the images into two categories and get rid of the bad images. Kalka *et al.* [21] applied Dempster-Shafer theory to develop a comprehensive quality measure by combining several quality measures. Chen *et al*. [22] presented a Mexican hat wavelet method for iris quality evaluation. The quality of iris is a major consideration of most iris recognition. Thus, commercial iris recognition systems usually reject images of less-than-acceptable quality.

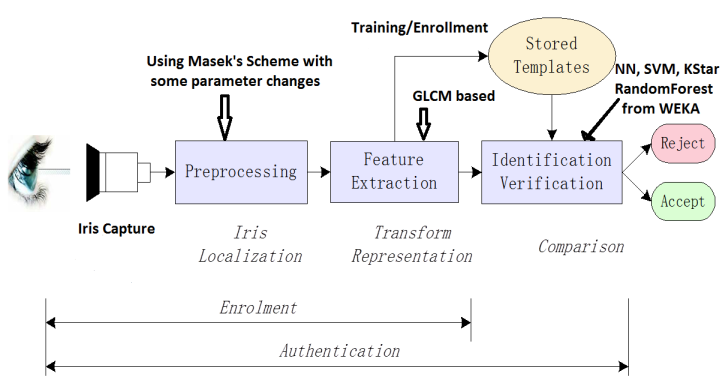


Figure 4: Iris Recognition System

In this thesis, we explore the application of GLCM (**Gray Level** **Co-occurrence Matrix)** iris recognition. So far, there have been few attempts to use GLCM for iris recognition. The technique described in [27] uses 190 different classes from UBIRIS database, a variant of neural network classifier, and 7 GLCM features. It uses only 5 images per class presumably the better quality images. The paper reports 97% accuracy, and 3% mis-recognition. The technique described in [28] uses 30 different classes (no database mentioned), a variant of nearest neighbor network, and some innovative GLCM based features. The best result reported in the paper is 83%. In this thesis, we have clarified a number of issues related to application of GLCM based feature to iris recognition. Finally, we have shown that a multi-classifier recognition scheme using GLCM based features holds promise for iris recognition. Since many images in our experiment especially from UBIRIS v.1 databases are of poor quality iris images, our performance indicate that GLCM based are robust for iris recognition. Finding robust feature extraction techniques capable of handling degraded iris images is an open research area of iris recognition.

This thesis is organized as follows. In next section, we described our proposed scheme in detail. In this regard, we discuss our database selection, feature selection as well as classifier selection. We elaborate the role of segmentation in our scheme. Section III describes our experimental results. Section IV describes the conclusions and scope for further research.

1. Proposed Iris Recognition Scheme

In this section, we describe the problem. We also describe the selection of iris corpuses, classifiers, features and classifier fusion.

**Database and Class Selection**

We have used iris images from two databases. These two databases are: (1) **UBIRIS.v1** [29] and **CASIA IRIS LAMP DATABASE** [30]. However, we have to be selective as we need irises without much occlusion from eyelashes and eyelids. Moreover, the segmentation algorithm used in our experiment [7] fails to properly segment many irises. Thus, many classes with enough good irises could not be used. It should be appreciated that this is not a shortcoming of our proposed scheme. Since we only have the segmentation scheme as described in [7] available, we have selected only those classes that have more than 7 good irises with little occlusion and segmented properly by the algorithm described in [7]. In total, we selected 51 classes, 28 from UBIRIS.V1 and 23 from CASIA IRIS LAMP DATABASE as described below.

UBIRIS.V1 Database is composed of 1877 images collected from 241 subject over 2 sessions; 40 subjects are initially selected as they have iris images with little occlusion. Finally, 28 subjects are selected as irises from 12 subjects could not be segmented adequately.CASIA IRIS LAMP DATABASE is composed of 819 different classes (411 subjects)**;** 51 classes are initially selected as they have iris images with little occlusion. Finally, 23 subjects are selected as irises from 28 subjects could not be segmented adequately.

**GLCM Based Texture Features**

An **image texture** is a set of metrics calculated in image processing designed to quantify the perceived texture of an image. Image Texture gives us information about the spatial arrangement of color or intensities in an image or selected region of an image. Image textures can be artificially created or found in natural scenes captured in an image. Image textures are one way that can be used for classification of images.

The basis of Haralick's texture features [27-28] is the gray-level co-occurrence matrix (**G** in Equation (2.1)). This matrix is square with dimension *Ng*, where *Ng* is the number of gray levels in the image. Element [*i*,*j*] of the matrix is generated by counting the number of times a pixel with value *i* is adjacent to a pixel with value *j* and then dividing the entire matrix by the total number of such comparisons made. Each entry is therefore considered to be the probability that a pixel with value *i* will be found adjacent to a pixel of value *j*.

|  |  |
| --- | --- |
|  | (2.1) |

Since adjacency can be defined to occur in each of four directions in a 2D, square pixel image (horizontal, vertical, left and right diagonals - see Figure 5), four such matrices can be calculated. The adjacency parameter is defined by d[dx,dy] where dx stands for distance between X-coordinates of two pixels, and dy stands for distance between Y-coordinates of two pixels. Here, the usual raster scan of the image is assumed.

|  |
| --- |
|  |

Figure 5: Four directions of adjacency as defined for calculation of Haralick texture features; these directions are termed as d=[1,0]; d=[0, 1]; d=[1, 1] and d=[[-1, 1].

As noted before, the Haralick statistics are calculated from co-occurrence matrices generated using each of the directions of adjacency shown in Figure 5.

Co-occurrence matrices can capture properties of a texture, though they are not directly useful for further analysis. Numeric features computed from co-occurrence matrices can be used to represent and compare textures. Fourteen such features are described in the literature.The following 4 features computed from co-occurrence matrices are widely used, and have been used in our work.



|  |  |  |  |
| --- | --- | --- | --- |
| *Entropy* = − | ∑ | ∑ | *Nd*(*i*,*j*)*log*2*Nd*(*i*,*j*) |
|  | *i* | *j* |  |

|  |  |  |  |
| --- | --- | --- | --- |
| *Contrast* = | ∑ | ∑ | (*i* − *j*)2*Nd*(*i*,*j*) |
|  | *i* | *j* |  |



**Classifier Selection**

In this subsection, we will describe a number of commonly used classifiers.

**Nearest Neighbor Classifier**

Nearest Neighbor Classifier is the most flexible classifier possible. To classify an observation, all that is needed is to find the most similar example in the training set and return the class of that example. This is called 1-nearest-neighbor classification, or 1-nn More generally, we can take the k most similar examples and return the majority vote. This is k-nearest-neighbor classification. The relative frequency of classes among the neighbors can be used to get a crude measure of class probability.

Nearest-neighbor classifiers are very simple to design (all that is needed is to get a database of examples), and often equal or exceed in accuracy much more complicated classification methods. A necessary part of nearest neighbor classification is **nearest neighbor retrieval**, i.e., the task of actually finding the nearest neighbors of the query. This can be tricky to do efficiently, especially when the database is very large.

Nearest-neighbor retrieval has many uses in addition to being a part of nearest-neighbor classification. For example, when biologists identify a new protein, they typically use a computer program to identify, in a large database of known proteins, the proteins that are the most similar to the new protein.

Saying that a database object is the "nearest neighbor" of the query implies that we have a way to measure distances between the query and database objects. The way we choose to measure distances can drastically affect the accuracy of the system. At the same time, defining a good distance measure can be a challenging task. For example, what is the right way to measure similarity between two Web pages? This is a perennial research problem. Usually, un-weighted Euclidean distance measure is used to find the distance. This un-weighted Euclidean distance based nearest neighbor classifier is termed as IB1 or Instance Based 1 in WEKA terminology; and is found to be very successful in our scheme. Incidentally, Weka’s IB1 classifier implements standard nearest-neighbor algorithm using Euclidean distance with all attributes being normalized into a [0; 1] range [31]. On the other hand**,** *WEKA KStar algorithm also implements nearest-neighbor algorithm using data derived weighted Euclidean distance with all attributes being again normalized into a [0; 1] range [31].*

Examples of other measures include the edit distance for strings, the chamfer distance, Hausdorff matching for edge images, the Kullback-Leibler distance and the Earth Mover's Distance for probability distributions, and bipartite matching for sets of features.

**Randomforest Classifier**

Randomforest classifier is an advanced tree classifier. While using multiple decision trees, the focus is on splitting criterion and the tree sizes. The dilemma between over-fitting and achieving maximum accuracy is seldom resolved. In RandomForest classifier, subsets of the feature set are randomly selected and trees are constructed based on these subsets. Weka implements Breiman's RandomForest algorithm [31]

**Multi-Layer Perceptron Neural Net**

Multi-Layer perceptron (MLP) is a feed-forward neural network with one or more layers between input and output layer. Feed-forward means that data flows in one direction from input to output layer (forward). This type of network is trained with the back propagation learning algorithm. MLPs are widely used for pattern classification, recognition, prediction and approximation. Multi-Layer Perceptron can solve problems which are not linearly separable. Details about MLP-NN can be found in [32].

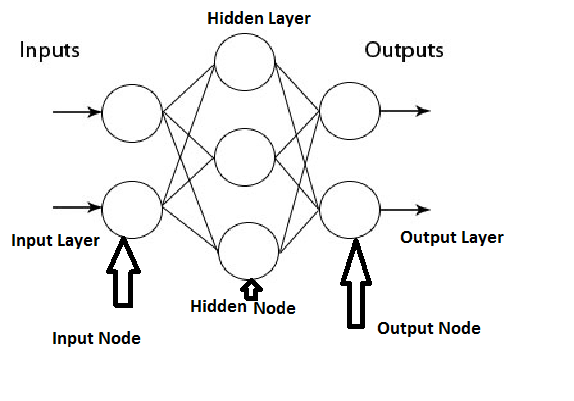


Figure 6: Multi-Layer perceptron (MLP) neural network with 3 layers, left-most layer is the input layer, middle layer is the hidden layer; right-most layer is the output layer.

**Support Vector Machine (SVM) Models**

A Support Vector Machine (SVM) performs classification by constructing an *N*-dimensional hyper-plane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron network described above. Thus, Support Vector Machine (SVM) models are a close cousin to classical multilayer perceptron neural network. Using a kernel function, SVM’s are an alternative training method for polynomial, radial basis function and multi-layer perceptron classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training.

In the parlance of SVM literature, a predictor variable is called an *attribute*, and a transformed attribute that is used to define the hyper-plane is called a *feature*. The task of choosing the most suitable representation is known as *feature selection*. A set of features that describes one case (i.e., a row of predictor values) is called a *vector*. So the goal of SVM modeling is to find the optimal hyper-plane that separates clusters of vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other size of the plane. The vectors near the hyper-plane are the *support vectors*. The figure below presents an overview of the SVM process.

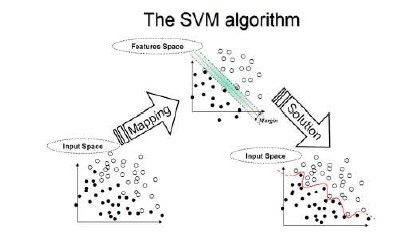


Figure 7(a): Support Vector Machine

**A Two-Dimensional Example**

Before considering *N*-dimensional hyper-planes, let’s look at a simple 2-dimensional example. Assume we wish to perform a classification, and our data has a categorical target variable with two categories. Also we assume that there are two predictor variables with continuous values. If we plot the data points using the value of one predictor on the X axis and the other on the Y axis we might end up with an image such as shown below. One category of the target variable is represented by rectangles while the other category is represented by ovals.

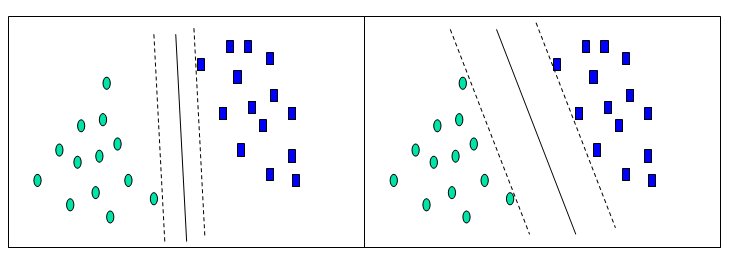


Figure 7(b): A Support Vector Machine example

In this idealized example, the cases with one category are in the lower left corner and the cases with the other category are in the upper right corner; the cases are completely separated. The SVM analysis attempts to find a 1-dimensional hyper-plane (i.e. a line) that separates the cases based on their target categories. There are an infinite number of possible lines; two candidate lines are shown above. The question is which line is better, and how do we define the optimal line.

The dashed lines drawn parallel to the separating line mark the distance between the dividing line and the closest vectors to the line. The distance between the dashed lines is called the *margin*. The vectors (points) that constrain the width of the margin are the *support vectors*. The following figure illustrates this.

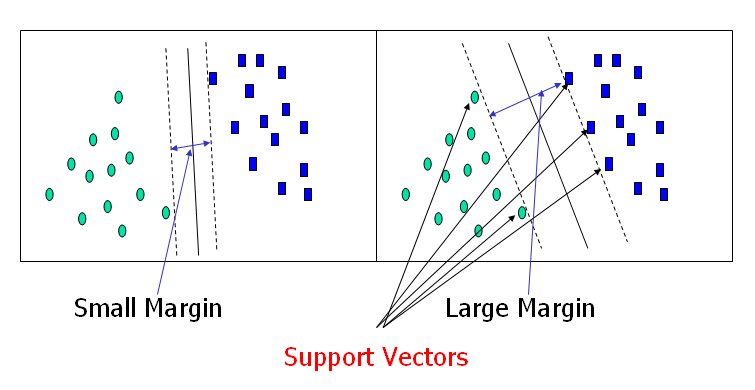


Figure 7(c): Support vectors of a simple Support Vector Machine example

An SVM analysis finds the line (or, in general, hyper-plane) that is oriented so that the margin between the support vectors is maximized. In the figure above, the line in the right panel is superior to the line in the left panel.

If all analyses consisted of two-category target variables with two predictor variables, and the cluster of points could be divided by a straight line, life would be easy. Unfortunately, this is not generally the case, so SVM must deal with (a) more than two predictor variables, (b) separating the points with non-linear curves, (c) handling the cases where clusters cannot be completely separated, and (d) handling classifications with more than two categories.

In the previous example, we had only two predictor variables, and we were able to plot the points on a 2-dimensional plane. If we add a third predictor variable, then we can use its value for a third dimension and plot the points in a 3-dimensional cube. Points on a 2-dimensional plane can be separated by a 1-dimensional line. Similarly, points in a 3-dimensional cube can be separated by a 2-dimensional plane. The logic is inductive, and generalizes SVM to higher dimension.

**Implementation Details**

In the following, we show the schematic diagram of the proposed iris recognition scheme using the techniques and databases described above.

Iris Images from Iris Databases

Select Classes that Contain Iris Images with Little Occlusion

Reject Classes If iris Segmentation Fails for a Number of Images

Segment Iris Image; Convert Annular Iris Image into Rectangular Image

Multi-Classifier Based Iris Recognition

Iris Recognition

Extract GLCM Based Features from Iris Image After Histogram Equalization

Figure 8: Schematic diagram of proposed iris recognition scheme

**Iris Segmentation**

We have used the scheme described in [7] for iris segmentation as shown below.



Figure 9: Left – An iris image, Right – Segmented circular iris outlined

The scheme described in [7] uses a circular model for iris. This is a simplification. A better is elliptical shape as used in [6]. The main reason for using the segmentation technique of [7] is that the software implementation is available publicly. However, this technique does not work well for all images, and extensive parameter tweaking is needed to make it work for different type of iris images. Still, the algorithm fails in many situations as shown below in Fig. 10.

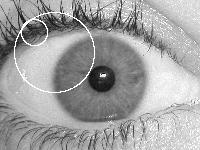


Figure 10: Failure of iris segmentation algorithm described in [7]

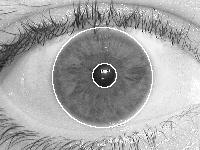


Figure 11: A perfectly segmented iris using iris segmentation algorithm of [7]

**Creation of Rectangular Iris Image**

Once again, we use the implementation described in [7] to convert annular/circular iris image which is obtained by iris segmentation to a rectangular image. The scheme is shown below using Fig. 12.

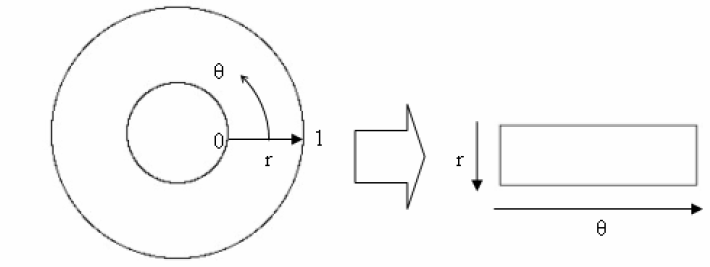


Figure 12: Conversion of annular iris image to a rectangular iris image



Figure 13: Rectangular iris image of original iris image given by Fig. 11

**Feature Extraction from Rectangular Iris**

GLCM based features are extracted from rectangular iris images. All rectangular iris images are of size 60x240. Before features are computed, the rectangular iris images are contrast enhanced using an adaptive histogram equalization program. For this task, adapthisteq function of Matlab is used using all default parameters except the distribution parameter that is set to ‘Rayleigh’. Experimentally, we have found that adaptive histogram equalization performs better than regular histogram equalization as regular histogram equalization tends to increase the contrast too much. After the image is histogram equalized, GLCM features are computed for four different ‘d’ values as shown in Fig. 5. From Fig. 5, one can see that there are four ways d can have one zero; d=[0, 1]; [0, -1], d[1,0] and [-1, 0]. The co-occurrence matrices belonging to all four ‘d’ values are averaged to form one co-occurrence matrix and 4 features as described before are calculated. Similarly, we define d=[1, 1], d=[1, -1], d= [-1, 1] and d=[-1, -1]. Once again, these co-occurrence matrices belonging to all four ‘d’ values are averaged to form one co-occurrence matrix and 4 features as described before are calculated. Finally, d=[2, 0], d[0,2], d[-2,0] and d=[0, -2] are used to define 4 more co-occurrences matrices. Once again, these co-occurrence matrices belonging to all four ‘d’ values are averaged to form one co-occurrence matrix and 4 features as described before are calculated. Finally, for d=[0,0] co-occurrence matrix, one feature (energy) is computed. Thus, from each image segment, 13 features are computed. Since we break our rectangular 60x240 iris images into 60x60 overlapping images with 50% overlap, we get 7 image segments. Thus, each image is represented by 91 dimensional feature vector. The algorithm is summarized below

* Original iris image is segmented into an annular iris image
* Annular iris image is converted rectangular iris image of size 60x240 (fixed)
* Use adaptive histogram equalization to enhance image contrast
* Break 60x240 image into seven 60x60 overlapping windows (50% overlap)
* From each window, compute 13 GLCM based features:
* Compute 91 features per image

**Classifiers Used**

For our classification scheme, we have used: (1) Nearest Neighbor (IB1 in WEKA), (2) Support Vector Machine (SMO in WEKA), (3) RandomForest Tree Classifier, (4) KStar Classifier, and (5) Multi-Layer Perceptron (NN in WEKA). For all these classifiers, **default WEKA parameters used**; only number of trees for RandomForest is set to 40.

WEKA:( <http://www.cs.waikato.ac.nz/ml/weka/>) is free software implemented in JAVA. WEKA implements all well-known classifiers and needs the data to be formatted into WEKA’s data format (ARFF format). One can use 10-fold cross validation to test the classification performance, or one can use traditional training/test sets to evaluate a classifier. WEKA also provides detailed output Thus, WEKA is one stop shop for most classifiers, and is widely used by researchers.

1. **Experimental Results**

**Class and Classifier Selection**

We have selected 51 classes from two databases (UBIRIS. v1 and CASIA LAMP Database) as described below.

|  |  |  |  |
| --- | --- | --- | --- |
| **UBIRIS.v1 SUBJECT ID** | **No. of Images** | **CASIA IRIS LAMP SUBJECT ID** | **No. of Images** |
| SUB01 | 10 | 5L | 28 |
| SUB02 | 10 | 9R | 30 |
| SUB08 | 8 | 23L | 15 |
| SUB11 | 10 | 23R | 15 |
| SUB12 | 10 | 24L | 10 |
| SUB15 | 9 | 34R | 11 |
| SUB16 | 10 | 90R | 14 |
| SUB19 | 9 | 110R | 15 |
| SUB44 | 9 | 160L | 13 |
| SUB52 | 10 | 160R | 16 |
| SUB55 | 10 | 167L | 11 |
| SUB59 | 10 | 167R | 11 |
| SUB61 | 10 | 188L | 11 |
| SUB71 | 8 | 188R | 14 |
| SUB76 | 10 | 216L | 20 |
| SUB81 | 9 | 216R | 20 |
| SUB85 | 10 | 239L | 11 |
| SUB94 | 10 | 269R | 10 |
| SUB105 | 9 | 333R | 14 |
| SUB115 | 7 | 406L | 10 |
| SUB116 | 9 | 406R | 16 |
| SUB130 | 9 | 407R | 13 |
| SUB131 | 10 | 135L | 10 |
| SUB132 | 10 |  |  |
| SUB183 | 10 |  |  |
| SUB218 | 9 |  |  |
| SUB228 | 8 |  |  |
| SUB233 | 10 |  |  |

Table 1: Selected classes from two different iris databases – UBIRIS.V1 and CASIS IRIS LAMP

For our classification scheme, as stated before, we have used: (1) Nearest Neighbor (IB1 in WEKA), (2) Support Vector Machine (SMO in WEKA), (3) RandomForest Tree Classifier, (4) KStar Classifier, and (5) Multi-Layer Perceptron (NN in WEKA). For all these classifiers, **default WEKA parameters used**; only number of trees for RandomForest is set to 40. The table below shows the result using 28 classes from UBIRIS database. Here, we have 233 exemplars (average 9.4 per class); ten-fold classification mode is used.

|  |  |
| --- | --- |
| **Classifier** | **Accuracy** |
| Nearest Neighbor | 92% |
| KStar | 93% |
| SVM | 84% |
| Multilayer Perceptron | 77% |
| RandomForest (40) | 87.5% |

Table 2: Experimental classification result using 28 classes from UBIRIS.v1 iris database

For our next experiment with 23 classes from CASIA database, the same classifiers and settings are used as in previous experiment. For this experiment, 336 exemplars (average 14.7 per class) are used with ten-fold cross-validation (classification).

|  |  |
| --- | --- |
| **Classifier** | **Accuracy** |
| Nearest Neighbor | 96% |
| KStar | 94% |
| SVM | 96.5% |
| Multilayer Perceptron | 97% |
| RandomForest (40) | 93.5% |

Table 3: Experimental classification result using 23 classes from CASIA IRIS LAMP Database

The same classifiers and settings are used in the next experiment. For combined 51 classes (28 classes from UBIRIS and 23 from CASIA IRIS LAMP Database); ten-fold cross-validation (classification) is used.

|  |  |
| --- | --- |
| **Classifier** | **Accuracy** |
| Nearest Neighbor | 93% |
| KStar | 92% |
| SVM | 82% |
| Multilayer Perceptron | 87% |
| RandomForest (40) | 87.5% |

Table 4: Experimental classification result using 28 classes from UBIRIS V.1 iris database, and 23 classes from CASIA IRIS LAMP Database

We observe that 93% accuracy obtained without much fine-tuning of classifiers is good given less than 9 images (exemplars) per class for many classes. For UBIRIS.V1, 8% images are bad, and many are of average quality. Images from UBIRIS contribute more to errors. The performance of both SVM and NN can be improved by changing parameters. These results are in agreement with those reported by others using GLCM based features [27-28]. Thus, GLCM features hold promise for iris classification

In iris recognition, incorrect recognition is much more costly than no recognition. Because of incorrect recognition, a dangerous person could get unauthorized access or privileges. On the other hand, no decision only adds some more complexity to the recognition process, that is, acquire the iris image all over again and go through the recognition process again. Almost all commercial iris recognition systems carry out detailed quality check of iris, and generously use ‘no decision’ as an option. The GLCM based features and multi-classifier framework could be used to reduce the number of incorrect recognition. In this scheme, we pick top-3 classifiers in terms of performance: Nearest Neighbor, KStar and MLP-NN. During recognition, an exemplar is run through all 3 classifiers; and if two or more classifiers vote in favor of a particular class, that class is declared as the recognized class for the exemplar. Otherwise, no decision is made for the input exemplar.

We carry out three trials. In each case, one or two exemplars are randomly selected from 51 classes. For classes with less than 10 exemplars, only one exemplar is selected. After this process, 78 exemplars in total are selected. These are now used as test exemplars while the rest are used as training exemplars. The results are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Decision | **Trial 1** | **Trial 2** | **Trial 3** |
| Correct | 76 | 74 | 74 |
| Incorrect | 0 | 1 | 0 |
| No Decision | 2 | 3 | 4 |

Table 5: Results of three trials using multi-classifier fusion

Next, we report the results of Table 5 above in terms overall recognition rate.

|  |  |  |
| --- | --- | --- |
| **Correct Recognition** | **Incorrect Recognition** | **No Decision** |
| 95.8% | 0.4% | 3.8% |

Table 6: Consolidated results of three trials using multi-classifier fusion

The most important result is that incorrect recognition rate is now down to 0.4% while the best result using individual classifier is 7% incorrect recognition rate (for KStar classifier). In this case, we have 3.8% no recognition rate which is a common state of affair in iris recognition.

1. **Conclusions**

We conclude that 93% accuracy obtained for 51 classes without much fine-tuning of classifiers is good given less than 9 images (exemplars) per class for many classes. For UBIRIS.V1, 8% images are bad, and many are of average quality. Images from UBIRIS contribute more to errors. The performance of both SVM and NN can be improved by changing parameters. These results are in agreement with those reported by others using GLCM based features [27-28].

The most important result using multi-classifier fusion is that incorrect recognition rate is now down to 0.4% while the best result using individual classifier is 7% incorrect recognition rate (for KStar classifier). In this case, we have 3.8% no recognition rate which is a common state of affair in iris recognition. This result is very promising and similar results have not been reported in the literature so far.

In conclusion, the results reported here are better than all other results reported in the literature using GLCM based features. Thus, multi-classifier recognition scheme using GLCM based features holds promise for iris recognition. Since many images in the experiments especially from UBIRIS v.1 databases are of poor quality iris images, experimental performance indicates that GLCM based features are robust for iris recognition.

Looking forward, a major limitation of this scheme is the requirement of irises with no or little occlusion. Extending this work to irises with occlusion is a challenge. However, it should be noted that other schemes [1-12] reject irises that has more than 30% occlusion. Other future research direction is periocular recognition. In periocular recognition, iris is not segmented, and the entire eye image is recognized. Since no iris segmentation is required, GLCM based features could be suitable for such recognition tasks.

1. **References**

[1] L. Flom and A. Safir, “Iris recognition system,” *U.S. Patent* *4641349*, Feb. 03, 1987.

[2] K. Bowyer, K. Hollingsworth, and P. Flynn, “Image understanding for iris biometrics: A survey,” *Computer Vision and Image Understanding*, Vol. 110, no. 2, pp. 281–307, 2008.

[3] J. Daugman, “New methods in iris recognition,” *IEEE Transaction on Systems Man and Cybernetics.*, Vol. 37, no. 5, pt. B, pp. 1167–1175, 2007.

[4] J. G. Daugman, “High confidence visual recognition of persons by a test of statistical independence,” *IEEE Transaction on Pattern Analysis and Machine* *Intelligence*, Vol. 15, no. 11, pp. 1148–1160, Nov. 1993.

[5] J. G. Daugman, “Demodulation by complex-valued wavelets for stochastic pattern recognition,” *International Journal on Wavelets, Multi-Resolution and Information Processing*, Vol. 1, no. 1, pp. 1–17, Jan. 2003.

[6] R. P. Wildes, “Iris recognition: An emerging biometric technology,” *Proceedings of IEEE*, Vol. 85, no. 9, pp. 1348–1363, Sep. 1997.

[7] L. Masek, “*Recognition of human iris patterns for biometric identification*,” B.S. dissertation, The School of Computer Science and Software Engineering, The University of Western Australia, Crawley, WA, Perth, Australia, 2003.

[8] L. Ma, Y. Wang, and T. Tan, “Iris recognition using circular symmetric filters,” *Proc. 16th International Conference on Pattern Recognition. (ICPR)*, Quebec City, Canada, Vol. 2, pp. 805–808, Aug. 2002.

[9] S. Lim, K. Lee, O. Byeon, and T. Kim, “Efficient iris recognition through improvement of feature vector and classifier,” *Journal of Electronic* *Telecommunications Research Institute*, Vol. 33, no. 2, pp. 61–70, Jun. 2001.

[10] L. Ma, T. Tan, and Y. Wang, “Efficient iris recognition by characterizing key local variations,” *IEEE Transaction on Image Processing*, Vol. 13, no. 6, pp. 739–750, Jun. 2004.

[11] J. Huang, L. Ma, Y. Wang, and T. Tan, “Iris model based on local orientation description,” *Proc. Asian Conference on Computer Vision*, Korea, pp. 954–959, Apr. 2004.

[12] X. Yuan and P. Shi, “Iris feature extraction using 2-D phase congruency,” *Proc. Third International Conference on Information Technology Applications. (ICITA)*, Sydney, Australia, Vol. 33, pp. 437–441, Jul. 2005.

[13] V. Dorairaj and N. Schmid, “Performance evaluation of iris based recognition system implementing PCA and ICA techniques,” *Proc.* *SPIE Conference on Biometric Technology and Human Identification*, pp. 51–58, Mar. 2005.

[14] J. Thornton, M. Savvides, and B. V. Kumar, “Robust iris recognition using advanced correlation techniques,” *Proc. Second International Conference on Image Analysis and Recognition (ICIAR)*, Toronto, Canada, Vol. 3656, pp. 1098–1105, Sep. 2005.

[15] J. Huang, Y. Wang, T. Tan, and J. Cui, “A new iris segmentation method for recognition,” *Proc. 17th International Conference on Pattern Recognition.* *(ICPR)*, Cambridge, U.K., Vol. 3, pp. 23–26, Aug. 2004.

[16] A. Abhyankar and S. Schuckers, “Active shape models for effective iris segmentation,” *Proc. SPIE Conference on Biometric Technology and Human Identification .*Orlando, FL, pp. 62020H.1–62020H.10, Apr. 2006.

[17] Samir Shah and Arun Ross, Iris Segmentation Using Geodesic Active Contours, *IEEE Transaction on Information Forensics and Security*, Vol. 4, No. 4, December 2009.

.

[18] W. Boles and B. Boashash, “A human identification technique using images of the iris and wavelet transform,” *IEEE Transaction on Signal Processing*, Vol. 46, no. 4, pp. 1185–1188, Apr. 1998.

[19] J. Daugman, “How Iris Recognition Works,” *IEEE Transaction on Circuits and Systems for Video Technology*, Vol. 14, No. 1, pp. 21-30,2004.

[20] L. Ma, T. Tan, Y. Wang, and D. Zhang, "Personal identification based on iris texture analysis," *IEEE Transaction on Pattern Analysis and Machine* *Intelligence*, Vol. 25, pp. 1519 - 1533, 2003.

[21] N. D. Kalka, V. Dorairaj, Y. N. Shah, N. A. Schmid, and B. Cukic, "Image Quality Assessment for Iris Biometric," *The Biometric* *Consortium Conference*, Arlington, USA, 2005.

[22] Y. Chen, S. C. Dass, and A. K. Jain, "Localized Iris Image Quality Using 2-D Wavelets," *Lecture Notes in Computer Science*, Vol. 3832, pp. 373 -381, 2005.

[23] C. Belcher, and Y. Du, “A Selective Feature Information Approach for Iris Image Quality Measure,” *IEEE Transactions on Information* *Forensics and Security*, Vol. 3, pp. 572-577, 2008.

[24] G. Zhang and M. Salganicoff, “Method of Measuring the Focus of Close-Up Images of Eyes,” *United States Patent*, No. 5953440, 1999.

[25] X.-D. Zhu, Y.-N. Liu, X. Ming, and Q.-l. Cui, "A Quality Evaluation Method of Iris Images Sequence Based on Wavelet Coefficients in "Region of Interest"," *Proc. Fourth International Conference on* *Computer and Information Technology (CIT'04)*, pp. 24 – 27, 2004

[26] R. P. Lippmann, “An Introduction to Computing with Neural Net,” *ASSP Magazine*, Vol. 4, No. 2, pages: 4-22, 1987.

[27] R. M. Sundaram and B. C. Dhara, **“**Neural network based Iris recognition system using Haralick features” *Proc.* *3rd International Conference*on*Electronics Computer Technology (ICECT), 2011* Vol. 3, pp. 19 – 23, 2011.

[28] A. Zaim, A. Sawalha, M. Quweider, J. Iglesias, and R. Tang, “A New Method for Iris Recognition using Gray-Level Co-occurrence Matrix**,”** *Proc.**IEEE International Conference on Electro/information Technology,* pp. 350 – 353, 2006.

[29] UBIRIS.v1 **(**[**http://iris.di.ubi.pt/**](http://iris.di.ubi.pt/)**),** January, 2012

[30]CASIA IRIS LAMP DATABASE [**http://biometrics.idealtest.org/dbDetailForUser.do?id=3**](http://biometrics.idealtest.org/dbDetailForUser.do?id=3)**,** January, 2012.

[31**]** I. H. Witten and E. Frank, *Data Mining: Practical machine Learning Tools and Techniques,* Elsevier, Amsterdam, 2005. Also, *Weka Machine Learning Project*, The University of Waikato, New Zealand, <http://www.scms.waikato.ac.nz/ml/>, December, 2012.

[32]R. M.Haralick, K. Shanmugan, and I. Dinstein, "Textural Features for Image Classification", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-3, 1973, pp. 610-621.

[33] R. M. Haralick and L.G. Shapiro, *Computer and Robot Vision*: Vol. 1, Addison-Wesley, 1992, p. 459.

[34] Leo Breiman, “Random Forests,” *Machine Learning*, Vol. 45, No. 1, pp. 5-32, 2001.

**Appendix: MATLAB and GLCM Features**

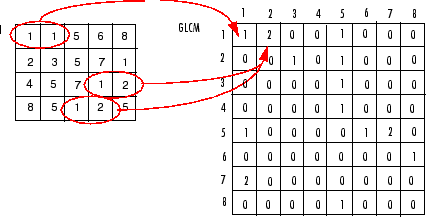
GLCM based features are extracted from rectangular iris images using MATLAB codes. In  **MATLAB graycomatrix** creates gray-level co-occurrence matrix from image using the syntax given below.

**Syntax for graycomatrix (from Matlab)**

* glcm = graycomatrix(I)  
  glcms = graycomatrix(I, param1, val1, param2, val2,...)  
  [glcm, SI] = graycomatrix(...)

glcm = graycomatrix(I) creates a gray-level co-occurrence matrix (GLCM) from image I. graycomatrix creates the GLCM by calculating how often a pixel with gray-level (grayscale intensity) value *i* occurs horizontally adjacent to a pixel with the value *j* (One can specify other pixel spatial relationships using the 'Offsets' parameter -- see Parameters using MATLAB ‘help’ function.). Each element (*i,j*) in glcm specifies the number of times that the pixel with value *i* occurred horizontally adjacent to a pixel with value *j*.

* The following figure shows how graycomatrix calculates several values in the GLCM of the 4-by-5 image I. Element (1,1) in the GLCM contains the value 1 because there is only one instance in the image where two, horizontally adjacent pixels have the values 1 and 1. Element (1,2) in the GLCM contains the value 2 because there are two instances in the image where two, horizontally adjacent pixels have the values 1 and 2. graycomatrix continues this processing to fill in all the values in the GLCM.



**Syntax for** GRAYCOPROPS **(from Matlab)**

GRAYCOPROPS Properties of gray-level co-occurrence matrix.

STATS = GRAYCOPROPS(GLCM,PROPERTIES) normalizes the gray-level

co-occurrence matrix (GLCM) so that the sum of its elements is one. Each

element in the normalized GLCM, (r,c), is the joint probability occurrence

of pixel pairs with a defined spatial relationship having gray level

values r and c in the image. GRAYCOPROPS uses the normalized GLCM to

calculate PROPERTIES, i.e., *feature vectors.*

* GLCM can be an m x n x p array of valid gray-level co-occurrence matrices. Each gray-level co-occurrence matrix is normalized so that its sum is one.
* PROPERTIES can be a comma-separated list of strings, a cell array containing strings, the string 'all', or a space separated string can be abbreviated, and case does not matter.

Properties include:

* 'Contrast' -- the intensity contrast between a pixel and its neighbor over the whole image. Range = [0 (size(GLCM,1)-1)^2]. Contrast is 0 for a constant image.
* 'Correlation' -- statistical measure of how correlated a pixel is to its neighbor over the whole image. Range = [-1 1]. Correlation is 1 or -1 for a perfectly positively or negatively correlated image. Correlation is NaN for a constant image.
* 'Energy' -- summation of squared elements in the GLCM. Range = [0 1]. Energy is 1 for a constant image.
* 'Homogeneity' -- closeness of the distribution of elements in the GLCM to the GLCM diagonal. Range = [0 1]. Homogeneity is 1 for a diagonal GLCM.