Iris Recognition Using Discrete Cosine Transform and Kekre’s Fast Codebook Generation Algorithm

H B Kekre¹, T K Sarode², V A Bharadi³
¹Sr. Professor, ²³Ph.D. Scholar, MPSTME, NMIMS University, Vileparle (w), Mumbai-56, India
²Assistant Professor, TSEC, Mumbai-50, India.
³Assistant Professor, TCET, Mumbai-101, India.

A Agrawal, R Arora, M Nair
B.E. Computer Engineering, Thadomal Shahani Engg. College, Bandra (w), Mumbai 400-050, India

ABSTRACT
Iris recognition enjoys universality, high degree of uniqueness and moderate user co-operation. This makes Iris recognition systems unavoidable in emerging security & authentication mechanisms. We propose an iris recognition system based on vector quantization and its performance is compared with the Discrete Cosine Transform (DCT). The proposed VQ based system does not need any pre-processing and segmentation of the iris. For vector quantization we have used Kekre’s Fast Codebook Generation Algorithm (KFCG). Proposed VQ based method using KFCG requires 99.99% less computations as that of full 2-dimensional DCT. Further the KFCG method gives better performance with the accuracy of 89.10% outperforming DCT that gives accuracy around 66.10%. The system has a future scope for integration in existing mechanisms.

Categories and Subject Descriptors
1.4. Image Processing and Computer Vision
1.4.2 Compression (Coding):- Approximate methods

General Terms
Algorithms, Performance

Keywords
Biometrics, Iris recognition, DCT, Vector Quantization, KFCG.

1. INTRODUCTION

In today’s world, where terrorist attacks are on the rise, employment of infallible security systems is a must. The identification of a person or an individual on the basis of their biometric characteristics like fingerprint, face, speech, and iris is thus gaining importance. Iris recognition is one of the important techniques and it is rotation and aging invariant. Compared with other biometric features (such as face, voice, etc.), the iris is more stable and reliable for identification [1]. Iris is the central part of the eye surrounding the pupil. Iris Recognition is the analysis of the coloured ring that surrounds the pupil [1]. The iris has unique structure and these patterns are randomly distributed; which can be used for identification of human being. Typical iris is shown in the eye image in the Figure.1. With fast development of iris image acquisition technology, iris recognition is expected to become a fundamental component of modern society, with wide application areas in national ID card, banking, e-commerce, welfare distribution, biometric passport, and forensics, etc. Since 1990s, research on iris image processing and analysis has achieved great progress [2].

![Figure 1: Eye image showing Iris, Pupil & Sclera](image1.png)

Figure 1: Eye image showing Iris, Pupil & Sclera

Generally, iris recognition system consists of four major steps. They include image acquisition from iris scanner, iris image preprocessing, feature extraction and enrollment / recognition. Image acquisition is a very important process as iris image with bad quality will affect the entire iris recognition process. One such system developed by center of biometrics & security research (http://www.cbsr.ia.ac.cn) is shown in Figure 2.(a), another such system using an iris capture device by OKI is shown in Figure 2.(b). Thus, it is critical to be implemented through good hardware design as well as software interface. Equally important is the iris image preprocessing step for mobile applications as the iris images taken by the users are less controllable as in the controlled laboratory environment. Improper iris image preprocessing can also influence the subsequent processes like feature vector extraction and enrollment/recognition [3].

![Figure 2: (a) Iris Capture device developed by CBS (b) Iris Camera from OKI](image2.png)

Figure 2. (a) Iris Capture device developed by CBS (b) Iris Camera from OKI (http://www.cbsr.ia.ac.cn).
Consequently, the iris preprocessing step needs to be robust and performs iris localization accurately. Daugman [4] made use of integro-differential operators for iris localization. It searches the path circularly to detect the iris boundary. The system by Tisse et al. [2] implemented the integro-differential operators and Hough transform for iris localization. Wildes [5] implemented a gradient-based edge detector (a generalized Hough transform) to detect local boundaries of an iris. Ma et al. [6] proposed a new algorithm which locates the center of pupil and uses it to approximate iris region before executing edge detection and Hough transform. Cui et al. [7] made use of the low frequency information from wavelet transform for pupil segmentation and localized the iris with integro-differential operator. Moreover, the eyelids detection was also performed after the eyelashes detection. These methods are used to define the area of iris which is later segmented for the feature extraction. In the next section we discuss some of the feature extraction methods.

2. IRIS FEATURE EXTRACTION METHODS

Many approaches are available in the literature, the iris texture contains information which should be extracted and represented using selected feature vectors. S Attrachi & K Faiez [8] have used a complex mapping procedure and best-fitting line for the iris segmentation and 1D Gabor filter with 2DPCA for the recognition approach. In the recognition procedure, they used the real term of 1D Gabor filter. In order to reduce the dimensionality of the extracted features, the new introduced 2DPCA method was used. Another such system using Gabor filter, 2DPCA & Gabor Wavelet Neural Network (GWNN) was proposed by Z Zhou, H Wu and Q Lv in [9].

H Koh, W Lee et. al. have proposed a multimodal iris recognition system [10] using two iris recognitions and also the levels of fusion and the integration strategies to improve overall system accuracy. This technique first implements the Daugman’s iris system using the Gabor transform and Hamming distance. Second, they proposed an iris feature extraction method having a property of size invariant through the Fuzzy-LDA with five types of Contourlet transform. This gives a multimodal biometric system based on two iris recognition systems. To effectively aggregate two systems, they used statistical distribution models based on matching values for genuine and impostor, respectively.

Iris recognition based on LDA and LPCC was proposed by Chu & Ching [11]. In addition, a simple and fast training algorithm, particle swarm optimization (PSO), was also introduced for training the Probabilistic Neural Network (PNN). Paul & Monwar [12] proposed iris recognition system consists of an automatic segmentation system that is based on the Hough transform, and is able to localize the circular iris and pupil region, occluding eyelids and eyelashes, and reflections. The extracted iris region was then normalized into a rectangular block with constant dimensions to account for imaging inconsistencies. Finally, the phase data from 1D Log-Gabor filters was extracted and quantized to four levels to encode the unique pattern of the iris into a bit-wise biometric template. The Hamming distance was employed for classification of iris templates. Besides these approaches many other systems are proposed, we can see that the performance of the system greatly depends on pre-processing, localization & segmentation of the iris. In this paper we discuss a system which does not needs the preprocessing. We have used DCT and Vector Quantization with clustering using Kekre’s Fast Codebook Generation Algorithm (KFCG). We discuss these techniques in the next sections.

3. DCT USED FOR IRIS RCOCGNITION

The discrete cosine transform (DCT) represents an image as a sum of sinusoids of varying magnitudes and frequencies. The DCT function for 2-dimensional image is given by Equation 1 & 2.

\[ B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \left( \frac{p(2m+1)}{2M} \right) \cos \left( \frac{q(2n+1)}{2N} \right) \]

\[ \alpha_p = \frac{1}{\sqrt{M}} \quad \text{if} \quad p = 0 \]
\[ \alpha_q = \frac{1}{\sqrt{N}} \quad \text{if} \quad q = 0 \]
\[ = \frac{2}{\sqrt{M}} \quad \text{if} \quad 1 \leq p \leq M-1 \]
\[ = \frac{2}{\sqrt{N}} \quad \text{if} \quad 1 \leq q \leq N-1 \]

Where \( B_{pq} \) are called the DCT coefficients of \( A \) which can be an image data \( A \) (m, n). The DCT decomposes a signal into its elementary frequency components. When applied to an MXN image/matrix, the 2D-DCT compresses all the energy information of the image and concentrates it in a few coefficients located in the upper-left corner of the resulting real-valued MXN DCT/frequency matrix [13]. This is illustrated in Fig. 3.

![Original Image](image1.jpg)  ![After applying DCT to image](image2.jpg)

Figure 3. Iris Image from the phoenix Database & it’s DCT

These DCT Coefficients can be used as a feature vector to retrieve the iris images. The retrieval of images becomes feasible because of the DCT coefficients as the DC components of DCT coefficients reflect the average energy of pixel blocks whereas the AC components reflect the intensity [13]. However, the representation of the image would not be compact because of the number of DCT coefficients and the number of pixels being equal. The common problem with the technique is that it is difficult to represent an image in its entirety or to perceive it by relating it to the DCT coefficients although the transformation contains the relevant information. DCT is a lossy compression technique that separates an image into discrete blocks of pixels of differing...
importance with respect to the overall image [14]. We describe the formal steps to extract the feature vector using DCT. We consider color iris images scaled to a dimension 128X128x3 Pixels

3.1 DCT Feature Extraction
The DCT algorithm that we have used for our study on iris recognition is as shown below:

1) Read the database image (Size=128X128x3)
2) Extract the Red, Green and Blue component of that image such that each is of size 128X128.
3) Apply DCT to each component and append in a new column the result for each Red, Green and Blue in matrix form. So we get 128X384 entries. This is the Feature Vector (F.V) of that image.
4) Repeat steps 1 through 3 for every database image.
5) Read the Query image.
6) Repeat step 2 and 3 for the query image so as to obtain its Feature Vector.
7) For every Database image ‘i’ and a Query image ‘q’ Calculate the Mean Squared Error using the Equation 3.

\[
MSE(i) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [FV_i(m,n) - FV_q(m,n)]^2
\]  

Where

- M=128, N=384,
- \(FV_i\) = Feature vector of i\textsuperscript{th} query image,
- \(FV_q\) = Feature vector of q\textsuperscript{th} query image

Determine the minimum M.S.E and corresponding image is matching iris. We discuss the accuracy of this method in the results section, next we discuss the Vector Quantization based feature vector generation.

4. VECTOR QUANTIZATION
Vector Quantization (VQ) [15-23] is an efficient technique for data compression and has been successfully used in various applications such as index compression [24, 25]. VQ has been very popular in a variety of research fields such as speech recognition and face detection [26, 27]. VQ is also used in real time applications such as real time video-based event detection [28] and anomaly intrusion detection systems [29], image segmentation [30-33], speech data compression [34], content based image retrieval CBIR [35] and face recognition [36].

Vector Quantization (VQ) techniques employ the process of clustering. Various VQ algorithms differ from one another on the basis of the approach employed for cluster formations. VQ is a technique in which a codebook is generated for each image. A codebook is a representation of the entire image containing a definite pixel pattern [15] which is computed according to a specific VQ algorithm. The image is divided into fixed sized blocks [15] that form the training vector. The generation of the training vector is the first step to cluster formation. The method most commonly used to generate codebook is the Linde-Buzo-Gray (LBG) algorithm which is also called as Generalized Lloyd Algorithm (GLA)[16].

In this paper, we carry out Iris Recognition using KFCG.

4.1 Kekre’s Fast Codebook Generation Algorithm (KFCG).
Here the Kekre’s Fast Codebook Generation algorithm proposed in [18][19] for image data compression is used. This algorithm reduces the time for code book generation. The image is divided into blocks of 2x2 pixels as shown in Figure 4. and each block is then converted to vector, thus forming a training vector set. Initially we have one cluster with the entire training vector set and the codevector C1 which is centroid.

In the first iteration of the algorithm, the clusters are formed by comparing first element of training vector with first element of code vector C1. The vector Xi is grouped into the cluster 1 if x1 < c1 otherwise vector Xi is grouped into cluster 2 as shown in Figure 5.(a) (First Iteration). Where codevector dimension space is 2. In second iteration, the cluster 1 is split into two by comparing second element x12 of vector Xi belonging to cluster 1 with that of the second element of the codevector. Cluster 2 is split into two by comparing the second element x12 of vector Xi belonging to cluster 2 with that of the second element of the codevector as shown in Figure 5.(b) (Second Iteration).

\[
P1=[R1, G1, B1], P2=[R2, G2, B2], P3=[R3, G3, B3], P4=[R4, G4, B4].
\]

Figure 4. Dividing Image into 2*2 pixel blocks to generate Training vector set.
This procedure is repeated till the codebook size is reached to the size specified by user. We have specified codebook size 16. Thus the feature vector space has 16x12 number of elements. This is obtained using following steps of Kekre’s Fast Codebook Generation (KFCG) algorithm

1. Image is divided into the windows of size 2x2 pixels (each pixel consisting of red, green and blue components).
2. These are put in a row to get 12 values per vector. Collection of these vectors is a training set (initial cluster).
3. Compute centroid (codevector) of the cluster.
4. Compare the first element of the training vector with the first element of the codevector and split the above cluster into two.
5. Compute the centroids of both the clusters obtained in step 4.
6. Split both the clusters by comparing second element of training vectors with the second element of the codevectors.
7. Repeat the process till we obtain codebook of size 16.
8. The result is stored as the feature vector for the image.

Thus the feature vector database is generated. Here we have preferred Euclidian distance as a similarity measure. The direct Euclidian distance between an image P and query image Q can be given in Equation 4.

\[ ED = \sqrt{\sum_{i=1}^{n} (V_{pi} - V_{qi})^2} \]  

(4)

where, V_{pi} and V_{qi} be the feature vectors of image P and Query image Q respectively with size ‘n’.

5. RESULTS

In the above implemented methods, we have not done any preprocessing on the iris images in the database or the query images. Also the images don’t solely contain the iris but also the sclera surrounding it. We have used phoenix database [37] consisting of irises of 64 individuals such that 3 images correspond to the left and 3 images corresponding to the right eye. 6 iris images (RGB) in Portable Network Graphics (PNG) format of each individual were taken into consideration. Thus in all there were (64 X 6) 384 such images as a part of our database. We have resized each image to a 128 x 128 matrix. Thus, we have a 3-dimensional image sized 128 x 128 x 3.

We have discussed DCT which transform based & KFCG which are VQ based feature vectors. These algorithms are now applied to the compressed form of the database image and then to the query image and the apparent match are sent as the result. Each of these algorithms were implemented in MATLAB 7.0 on Intel Pentium Dual Core Processor (2.01 GHz), 2GB RAM on Windows XP Professional SP3.

Now for the purpose of illustration we have taken a query image (which could be the left or the right eye of any individual of our database) and applied it to our proposed methodology and have shown what output is provides us with in the Figure.6.
images. In few cases it has happened that the left query image has given best match with the right iris image of the same person, this is also treated as success.

The Graph shows the comparison between the accuracy of DCT, and KFCG. Here the results so obtained are such that if the query image was of left/right eye of an individual then the output is also the left/right eye of that same individual. Since we have 64 images of left and 64 images of right iris and 256 database images, the accuracy is calculated as,

\[
\text{Accuracy for Left Eye} = \frac{N_1}{N_L} \times 100
\]

(5)

\[
\text{Accuracy for Right Eye} = \frac{N_2}{N_R} \times 100
\]

(6)

Where,

\(N_1=\) No. of correct individual identified for Left iris query images.

\(N_L=\) Total no. of Left iris images from the database (64).

\(N_2=\) No. of correct individuals identified for Right iris images.

\(N_R=\) Total no. of Right iris images from the database (64).

\(N=\) Total Number of Images in the database (\(N_L+N_R=128\))

\[
\text{Overall Accuracy} = \frac{(N_1+N_2)}{N} \times 100
\]

(7)

\[5.2\] Complexity Analysis

Let \(M\) be the total number of training vectors,

\(k\) be the vector dimension,

\(N\) be the codebook size,

1 CPU unit is required for addition of 8 bit numbers

1 CPU unit for comparison

For multiplication of two 8 bits number 8 CPU units are required.

To compute one squared Euclidean distance (ED) of \(k\) dimensional vector we require \(k\) multiplications and \(2k-1\) additions hence \(8k + 2k - 1\) CPU units.

\[
\text{Total CPU Units} = M \log_2 N
\]

TABLE I. KFCG ALGORITHM WITH RESPECT TO TOTAL NUMBER OF COMPARISONS, TOTAL NUMBER OF ED COMPUTATIONS AND TOTAL CPU UNITS REQUIRED.

<table>
<thead>
<tr>
<th>Complexity Parameters</th>
<th>KFCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Comparisons</td>
<td>(M\log_2 N)</td>
</tr>
<tr>
<td>Total No. of ED</td>
<td>0</td>
</tr>
<tr>
<td>Total CPU units</td>
<td>(M\log_2 N)</td>
</tr>
</tbody>
</table>

For full 2-Dimensional DCT for an \(P \times Q \times 3\) image the number of multiplications required are \(3xPxQx(P+Q)\) and number of additions required are \(3xPxQx(P+Q-2)\). Total CPU units required for full 2-Dimensional DCT = \(8x3xPxQx(P+Q) + 3xPxQx(P+Q-2)\).

From the Table II it is observed that DCT method requires 6906 times more computations as that of KFCG. It should be noted that in our study we have not employed the first stage i.e. Image Segmentation, in spite of which we get a better accuracy. Table III gives the summary of performance of individual algorithm tested. This is illustrated in a graph given in Figure. 7.

\[
\text{TABLE II. TOTAL CPU UNITS REQUIRED FOR KFCG AND DCT FOR} \quad r,q=128, M=4096, N=16 \text{ AND } k = 12
\]

<table>
<thead>
<tr>
<th>Complexity Parameters</th>
<th>KFCG</th>
<th>DCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total CPU Units</td>
<td>16384</td>
<td>113147904</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

In this paper we have discussed Iris Recognition using DCT & VQ based KFCG methods. We have implemented these algorithms on iris image without any pre-processing or segmentation including iris localization in spite of which it has been possible for us to obtain such a high accuracy. KFCG has the better performance with the accuracy of 89.10%. DCT has low accuracy around 66.10%. KFCG requires 99.99% less computations as that of full 2-D DCT. If we combine these methods with iris pre-processing the results can still be improved, this can be future scope for the research.

7. REFERENCES


[11] C Ch, C Chen, " High Performance Iris Recognition Based on LDA and LPCC"17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’05), IEEE DOI-1082-3409/05


[37] http://phoenix.inf.upol.cz/iris/download/, (referred on 10-09-2009, 10:00 a.m.).