Turkish Journal of Computer and Mathematics Education Vol. 9 No. 1 (2018),550-588
Research Article

EFFECTIVE ONLINE IRIS IMAGE REDUCTION AND RECOGNITION METHOD BASED ON EIGEN VALUES

Anjali A Bhadre [1], Harshvardhan P Ghongade [2], Dr. Ravindra N Katiyar [3]

[1,3] Chhatrapati Shahu Ji Maharaj University, Kanpur, India
[2] North East Frontier Technical University, Arunachal Pradesh, India

[1] anjalibhadre38@gmail.com, [2] ghongade@gmail.com, [3] rnkatiyar@gmail.com

Article Received: 08 January 2018 **Article Accepted:** 15 March 2018 **Online First:** 02 June 2018

Abstract: The efficient Eigen values-based approach for online iris image compression and human identification, including the situation of identical twins, is introduced in this study. The iris image is retrieved after eliminating the pupil, eyebrow, skin, and other noise disturbances from an accurate picture. The retrieved iris picture is partitioned into several blocks, each 16 by 16 pixels. Eigenvalues are now generated for each block to identify each block better and save it in the smart card memory.

Therefore, all that is required to determine whether two iris pictures are the same is to compare the stored Eigenvalues with the online-calculated Eigenvalues. The identical Eigenvalues between two iris scans indicate they belong to the same individual. According to our study, various people's iris images—including those of identical twins—have distinct Eigenvalues. We tested our Eigen Values Based Iris Image Identification Technique using datasets of iris images from CASIA and Multimedia University, and we discovered that it provides 99.99% accuracy for matching identical twin and individual photos. According to the implementation, our strategy seems to give the most extraordinary matching results for identical twins and people. It is a practical, cost-effective, and effective method for online personal identification.

Keywords: identical twins; Eigen value; iris matching: iris image compression and identification.

1. Introduction

The facial structure, fingerprints, or DNA sequence are often employed for personal identification. However, in the case of identical twins, these techniques fall short of identifying a person. The DNA sequence comprises the letters A, C, G, T, and N, where N stands for unidentified nucleotides (A, C, G, or T). We need a significant amount of memory to store the DNA sequence. The Sort Tandem Repeat (STR) sequence of DNA may, however, be stored in compressed format and utilised going forward in criminal investigations.

Furthermore, since there is no accepted technique for gathering online DNA samples, we cannot utilise DNA sequences to identify people online [Mishra, 2010].

Therefore, we need a method to detect identical twins quickly and with the least amount of data stored. An individual, even identical twins, may be identified using the iris image-based identification approach. Everyone, even identical twins, has a distinct iris pattern. As a result, one of

the most valuable primary biometric traits that may be utilised to identify persons is the human iris picture. A human iris picture may provide an additional level of distinctive information for systems that are built on high-quality image data. Individuals may always be identified using features taken from the human iris picture, even genetically identical twins. The iris image of a person is wholly developed six months after birth, and it is unaffected by sickness or pregnancy. It is difficult to alter or change one's iris image via cosmetic surgery. However, it will likely modify or change one's facial structure and fingerprints (including the thumbprint). A person's voice may be altered or modified after having throat surgery if it changes due to sickness. Therefore, unlike fingerprint or facial structure storage systems, iris picture-based authentication approaches cannot be utilised to fraudulently mimic other persons.

The following applications of the iris image-based personal identification technique are possible: Confidential Database Access, Automated Banking, Credit Cards, Government Benefit Distribution, National ID Cards, Passports, Smartcard, Personal Identification, and Home Security Systems [Jain, 1999; Shih, 2010]. A person's iris, however, requires a lot of memory (about 450KB), much like fingerprinting and human DNA sequences. There are certain obstacles, such as the need for precision, the time needed for identification, and the storage capacity for iris images. False Rejection Rate (FRR) and False Acceptance Rate (FAR) issues are a part of the accuracy [Doeoteo, 2006]. Time requirements indicate that the method should identify the correct individual quickly and reject situations of incorrect identification. We must solve these issues before using the iris image-based approach for identification. As a result, the iris image-based identification method must not use an excessive amount of memory or take an excessive amount of time to identify. Cases from FAR and FRR should be included. Additionally, a respectable compression speed must be attained.

Due to the complexity and memory requirements of these methods, intelligent card-based systems cannot employ dynamic and adaptive variable-length encoding. Therefore, run-length encoding and static/variable-length encoding techniques are the real possibilities for using them in intelligent cards. In this study, we propose the Iris Image Compression and Identification System (IICIS), which is an iris image compression and identification system. This method entails cropping the iris out of an image of the eye.

The recovered iris image will consist of a preset number of pixels. Each pixel will consist of three hues red (R), green (G), and blue (B). An integer number will represent these pixels. After iris image extraction, we divide the iris ring into numerous blocks of size 16. The final few blocks of the iris picture may not be precisely 16 by 16. We must thus convert them into 16 16 matrices by multiplying them by an identity matrix.

The Eigenvalues for these matrices, which are adequate to identify a person, must now be determined. Each 16 by 16 matrix will have 16 Eigenvalues. The Eigenvalues will be expressed as floating-point numbers. These floating-point numbers, or Eigenvalues, will be stored in the server database for further identification and comparison. To reduce an iris image online and identify a person, comparing previously held Eigenvalues with recently calculated Eigenvalues of an iris picture will be employed. This online Eigen values-based compression and identification

methodology will take the least amount of time compared to earlier techniques. The Eigenvalues produced from an image's iris ring serve as the foundation for compressing and recognizing iris pictures. A multitude of difficulties arise while creating distinguishing irises for ideas. If we must have all images as pixels, coloured images may be represented in many colour spaces.

2. Related Work

The DNA sequences were compressed using the shape DNA-based approach. The Laplace-Beltrami operator's spectrum is called "Shape DNA" since it includes inherent shape information [Reuter, 2009]. We have demonstrated that this shape-DNA may be used to identify things in real-world applications, much like a DNA test. There are identical twins with varied shapes yet the same shape-DNA. As a result, an approach based only on condition DNA cannot be utilised to identify an individual human. Mesh partitioning and compression have both been accomplished using a Laplace operator variation. Spectral compression is the name of the technique in question. Kami [2000] and Ma et al. [2002] provide details. High dimensional data space is also dimensionally reduced using the Laplace operator [Belkin, 2003]. The Eigenvalues of a manifold defined by points in a particular featured area were utilised by Belkin and Niyogi. If a collection of traits is present that will be diminished for various causes, the Belkin and Niyogi technique might be helpful. The Multi-Dimensional Scaling approach includes the Belkin and Niyogi methods. Its performance is comparable to conventional techniques, even though it is implemented differently [Wildes, 1997]. Many biometric systems have chosen to use the iris picture as one of its features. In 1987, Flom and Safir presented the fundamental concept of the iris and its compression [Folm, 1987]. Since then, many methods and software programs have been created. Daugman created one of the most cutting-edge and widely used iris systems [Daugman, 1999, 2007]. In numerical trials, the usage of 2D Gabor wavelets has shown a low likelihood of false positives at the expense of a very high frequency of false negatives. Similar technologies have been created by combining the Hough transform for iris localisation with pattern matching using a Gabor filter bank [Tisse, 2003; Ma, 2002]. Last but not least, many systems have been developed that use wavelets and winner selection techniques. However, due to the delayed release of the CASIA benchmark database, it isn't easy to assess the effectiveness of current strategies [CASIA, 2010]. Based on the Fourier-Miller Transform, Iris compression was recently created for cryptographically secure personal identification [Daniel, 2004]. Suppose we are using iris image-based techniques for online identification of a person for a highly secured zone, including the case of a nuclear reactor. In that case, the person who wants to access the nuclear reactor of an atomic research centre will insert his smart card. Then he will have to stand in front of the Iris Image Compression and Identification System (IICIS) camera.

The camera will take the iris image of the person. Now the skin, eyebrow, pupil, and other noise disturbances will be separated from the iris image. After separating these noise distur- bances, we will get the iris in its purest and actual format. This iris image of the person will be compared with stored Eigen values of the iris images of the smart card. Here, we need a fast iris image data comparison and its retrieval tech- nique. At the time of online identity verification of a person, the system should accurately ver- ify the identity of the person within the short- est time. To minimize the identification time, we need to store the iris image in compressed for- mat. That is, in the form

Eigen values. Thus, we need to compress the iris image such that it should take minimum comparison and trans- mission time. The compressed iris image data will be stored in the form of Eigen values in the smartcard. The employee of the nuclear reactor will insert his smart card in the system and the system will compare the online calculated Eigen value of iris image of the person with the stored Eigen values of the smartcard. If online calcu- lated Eigen values are same as the stored Eigen values, the system may display a special beep which confirms that the person is an authen- tic person and the person will be permitted to enter the nuclear reactor. If the Eigen values stored in the smartcard are not matching with the online calculated Eigen values, then it means that the person is not an authentic person and the person will not be permitted to access the nuclear reactor. This will be considered as the compressed format of iris image. At the time of designing the smartcard, we will have to store the employee code as the person's ID in the smartcard and this employee code of the smart card will first be matched with employee code stored in the server. Once the validity of smart card is checked, then the Eigen values stored in the smart will be compared with the online cal- culated Eigen values. Smaller numbers of papers deal specifically for determining the parts of the iris region that are occluded by eyelids and eye- lashes. Occlusions due to eyelashes are some- times referred to as "noise" [Bowyer, 2008].

3. Theoretical Background

In this section, we will explain the theoretical and mathematical background required for online iris image compression and identification. A method for uniquely identifying a particular human being by biometric analysis of iris of the eye, comprises of the following steps [Matey, 2010]:

- Acquire an image of an eye of the human to be identified.
- Isolate and define the iris of the eye within the image.
- Analyze the iris to generate a presenting iris code.
- Compare the mentioned presented code with a previously generated reference iris code to generate a measure of similarity between the mentioned presented iris code and the mentioned reference code.
- Convert the (as mentioned) similarity measure into a decision that (as mentioned) iris codes either do or do not arise from the same iris.
- Calculate a confidence level for the decision.

In our Iris Image Compression and Identification method, online Iris image compression technique has been divided into five phases. In the first phase (introduction phase), we have described the basics of iris image and its colors. In the second phase (preprocessing phase), we have converted the iris image into binary image for its processing. In the third phase (radii and centre estimation phase), we have estimated the centre and radii of the iris image. In the fourth phase (segmentation phase), we have removed the noise disturbances from an iris image and then divided it into different

blocks of same size. In the fifth phase (processing phase), we have converted each block of the iris image into different matrices of size 16×16. For each matrix, Eigen values are calculated and these Eigen values are used for further human identification.

3.1. Introduction phase

Every digital image is made up of small dots known as pixels. Each pixel is in turn made up of three different colors R, G, and B (Red, Green and Blue). Each color forms a different layer of the image, which, when superimposed on one another, gives rise to the complete image. Digitally, each layer can be represented as a two dimensional matrix, where each element of the matrix ranges from 0 to 255 . Each component (R, G and B) of the image is represented by a single two-dimensional matrix in digital format, which converts it into three-dimensional structures. For our purpose, we just need one dimension of the image, and preferably, we will work with the Green component of the image in matrix format although we can work with either green or red or blue component of the image. Here, our research work can be divided into several stages. The first stage consists of obtaining the actual image. That is, extracting a single dimension of the image and then performing other operations. This stage is called the pre-processing phase.

Now we need to calculate the centre, inner radii and outer radii of the actual iris ring. This stage will be called the Estimating Radii and Centre of Iris Ring. Once we complete this stage, then we need to extract the iris ring and resize the image, depending upon the outer radius. This phase will be called the Segmentation phase. After the segmentation phase, we need to divide the actual image matrix obtained from the Segmentation phase into smaller matrices and calculate the Eigen value for each of the smaller part. This phase will be called the processing phase.

3.2. Pre-processing phase

This is the initial phase of the procedure which is applied for an input image of iris to obtain the result in compressed format.

In this phase, we obtain a process-able format of the image from the actual raw image. The raw image will be obtained from a highresolution digital camera. As mentioned in the introduction section, an image consists of three two-dimensional matrices corresponding to Red, Green, and Blue components of the image.

Here, we have used MATLAB version 7 to carry out all our experimentation. MATLAB reads an image as three-dimensional structure of numbers, where each component of the third dimension represents a layer of numerical values corresponding to R, G and B. In MATLAB, this task is performed by the following statement:

$$g = \mathrm{img}(:,:,2).$$

Here, ' g ' is the variable which holds the second layer of the image and it corresponds to the Green color. 'img' is the actualthree dimensional image in the form of a three dimensional matrix. (, : and ; are MATLAB operators). A Binary image is the image whose elements are either zero or one. Here,

in our case, if an element in the green component is equal to zero, then it will be taken as zero and if it is greater than zero, then it will be taken as one in the corresponding Binary image. Once we have obtained the green component, then we convert this component into the corresponding binary image. A green component of iris image looks like the following figure (Fig. 2). In order to convert a green image into black and white image, we have used an inbuilt function of MATLAB. The statement accomplishing this task is as follows:

$$\text{gbw} = \text{im}\,2\text{bw}(\text{g})$$

Here, 'gbw' is the variable which holds the Binary Image and 'im2bw' is the function which converts an image to binary image. Figure 3 is

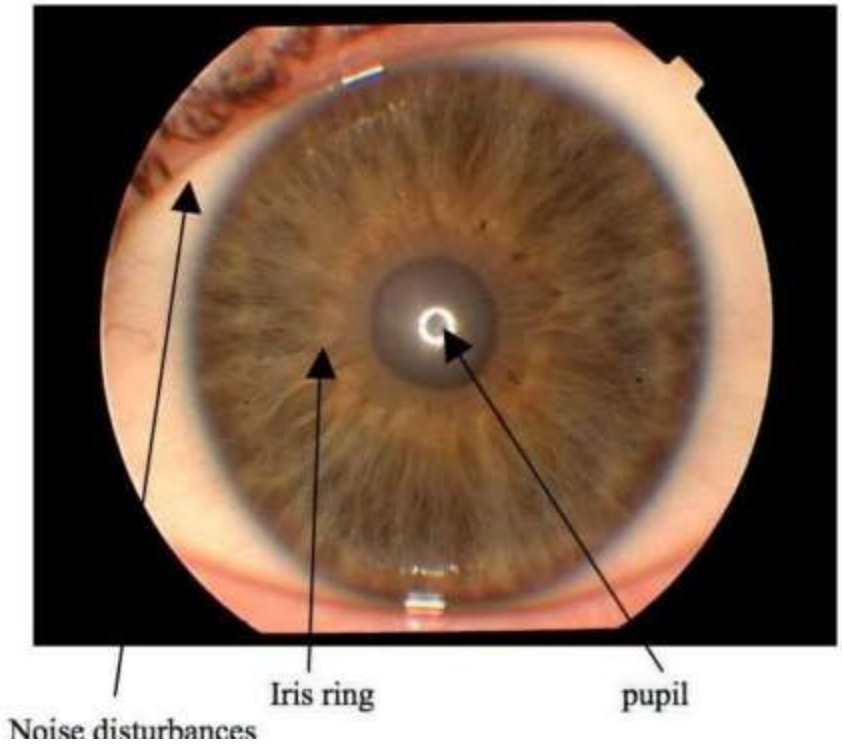

Fig. 1. A true color iris image.

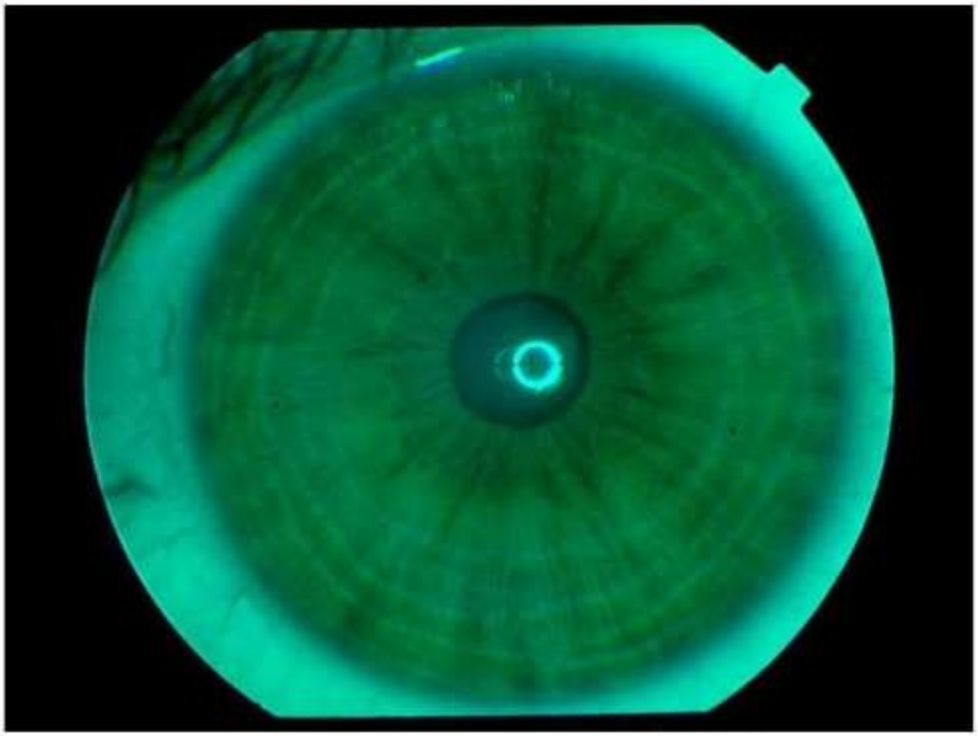

Fig. 2. Green component of image.

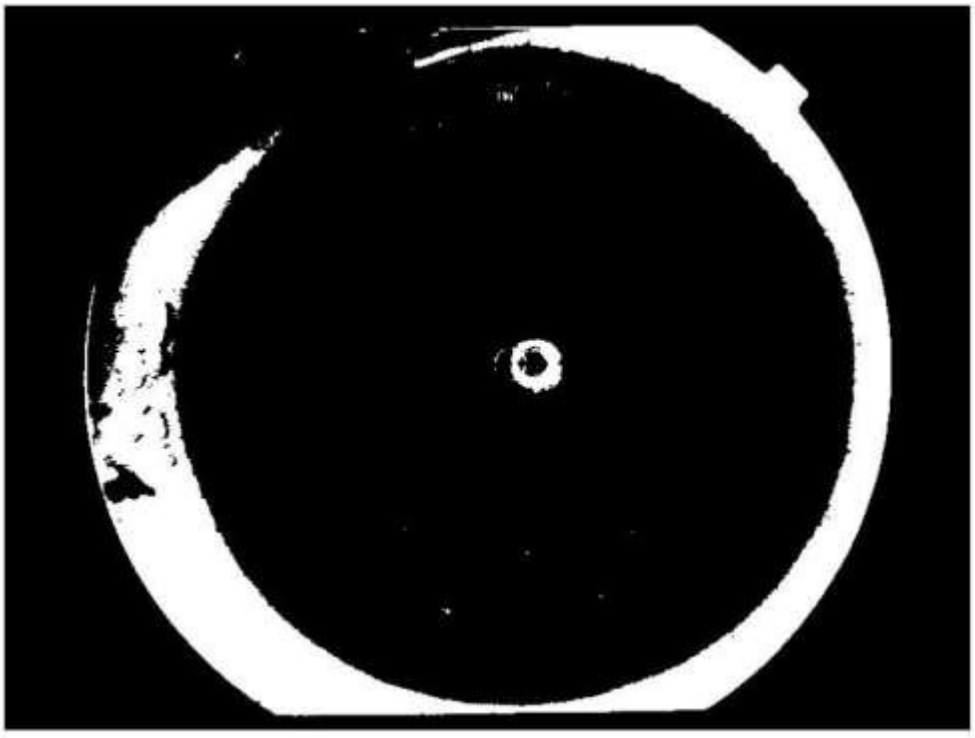

Fig. 3. Image converted to two colors (Black and White).

showing the black and white representation of Fig. 2.

Once the image has been converted into Binary Image, then this pre-processing phase will be completed and the output of this phase i.e., 'gbw' form of the Binary image will be used as the input to the next phase.

3.3. Estimating radii and centre of iris ring

In this phase, we will calculate the parameters which are defining the iris ring. The Iris Ring can be thought of as a portion of the image which is obtained after the removal of pupil and the portions of the image which is lying outside of the outer boundary of iris.

The iris ring and its other components can be easily visualized by Fig. 1. In this figure, all the portions of the image are shown. Here, two concentric black circles show the iris ring which is to be extracted. In order to extract this ring from a raw iris image, we need to calculate following three parameters:

(1) The coordinates of the Centre of iris image (g, h).

(2) The radius of the inner circle of the iris ring (ri).

(3) The radius of the outer circle of the iris ring (ro).

After calculating these parameters, we will proceed with the extraction of the iris image. In order to calculate these three parameters (gh, ri, &ro) we have adapted a separate method which has been implemented in MATLAB.

In this procedure, the output image 'gbw' obtained in the previous phase is taken as the input and the three parameters (namely 'gh', 'ri', and 'ro') are obtained as the output. These output variables will be used as input parameters to the next phase.

3.4. Segmentation phase

In this phase, the actual extraction of the iris ring is to be carried out, depending upon the parameters obtained in the previous phase. This phase proceeds in two parts.

3.4.1. Removing noise disturbances and extracting iris ring

In this part the unwanted portions of the image are removed and only the iris ring is retained. This is called the extraction of iris ring. We have written a customized MATLAB function in order to accomplish this task. The name of the function is 'extractIris( )'. This function takes the parameters obtained in the segmentation phase and the green component of actual iris image as the input and gives another image of the same dimensions as the actual image, containing only the iris ring as the output. The image containing the iris ring is the same size as the actual image. Therefore, it contains some unwanted space in it. Hence, we need to go to the second part of the segmentation phase. For purpose if reference, we call this output image as 'gRing'.

3.4.2. Resizing iris image

In this part, we remove the extra space of the iris ring of Part 1. In other words, we can say that we now resize the 'gRing' image according to the radius of the iris ring in order to remove the unwanted space. For this purpose, we have developed a MATLAB function named 'resizeM', which resizes the image according to the radius of the iris ring. This function takes the image 'gRing' and the outer radius of the iris ring 'ro' as input. The returned output is the resized image. The variable name used for resized image in our research work is 'reszgRing'. This output will be the actual resized image of the iris ring. The output of Part 2 will be used as input to the next phase. The resized image obtained after reducing the circumference will be represented as Fig. 4 .

Once the image is resized, we can proceed to the next phase called processing phase.

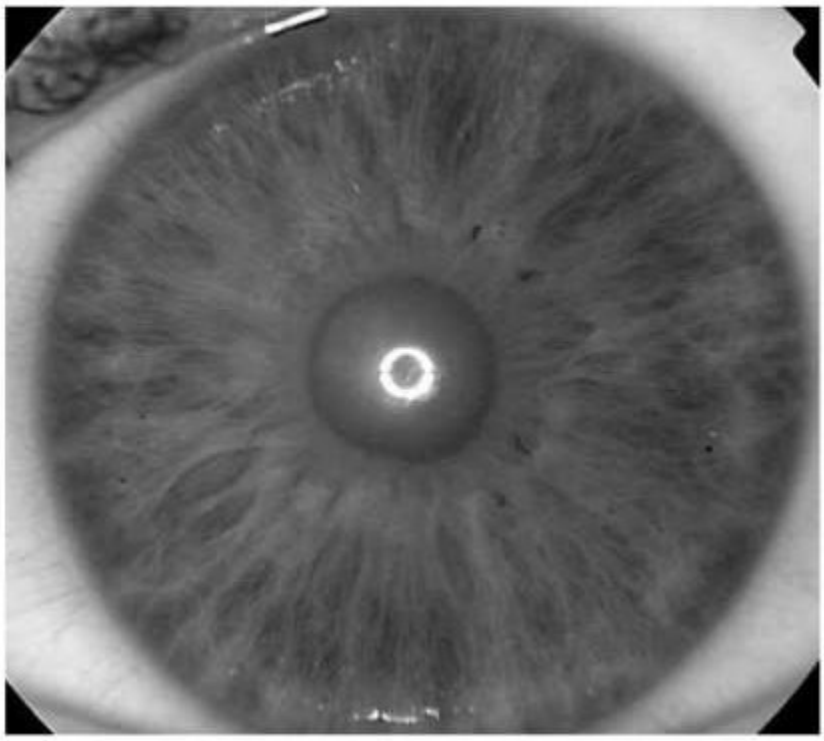

Fig. 4. Resized image after reducing the circumference of iris.

3.5. Processing phase

In this phase, the actual compression of the iris image is carried out. This phase is carried out in two steps.

4. Step 1: Converting iris ring into smaller matrices

We have explained that all these images will be in the form of two-dimensional matrices. Hence, the output image of the previous phase, i.e., 'reszgRing' will be in the form of a two-dimensional matrix. Now, we divide this bigger matrix into smaller 16×16 matrices. The input image should be a square matrix. If it is not, then we make it square either by adding or removing columns or rows. This task of dividing the matrix is carried out by an indigenous MATLAB function named 'divMat( )'. This function takes the resized image, i.e., 'reszgRing' and the dimension of the smaller

matrices as input and returns the divided smaller sized matrices in another three dimensional matrix. The smaller matrices obtained in this stage are used as the input to next stage.

5. Step 2: Calculating the Eigen values

In this stage, we calculate the Eigen values of the smaller matrices and store them in a linear array. The task of calculating Eigen values is carried out by our indigenous MATLAB function 'calEig'. In this method, we use characteristic equation of a matrix for calculating Eigen values. The characteristic equation is given by following mathematical function:

$$|\mathrm{A} - \mathrm{CI}| = 0$$

In Eq. (1) mentioned above, ' A ' is the matrix whose Eigen value is to be calculated, ' C ' is a numeric constant and ' I ' is an identity matrix whose order is same as that of ' A '. Here, the determinant of ' A $-$ CI ' results in a polynomial in ' C ' whose degree is the same as the order of the matrix ' A ', say P(C). Let the order of the matrix 'A' be 'nxn'. Hence, the degree of the polynomial P(C) will be ' n '. We will have ' n ' solutions of the polynomial P(C). Each of these ' n ' solutions is called the Eigen value of the matrix 'A'. These Eigen values will be considered as the compressed format of the iris image and these Eigen values can be further used for comparison and identification of a person.

6. Mathematical Model and Steps of IICIS

We have designed the structure of Iris Image Compression and Identification System (IICIS) mathematically in this phase. The mathematical structure is the abstract representation of IICIS which specifies the following:

(i) The inputs and outputs of IICIS at each state of processing.

(ii) The number of discrete states/stages encountered from start to the end of processing.

(iii) The interconnection between inputs, outputs and processing states.

(iv) The flow and data manipulation of different processing states of IICIS.

6.1. Definition of IICIS

The structure of IICIS can be defined by using the following six tuples:

$$\mathrm{IICIS} = \{\mathrm{Q}, \sum, \sigma, \delta, \lambda, \mathrm{q}_0\}$$

The symbols of Eq. (2) above are defined as follows:

Q: finite, non-empty set of states.

$\sum$: finite, non-empty set of input matrices.

σ : finite, non-empty set of output matrices.

δ : state transition function, which brings IICIS to the next state. The next state depends on the previous state. The transition function will be

$$\delta: \sum^* \times Q \to Q$$

λ : output function. The output depends on the state and the input elements. The output function can be defined as:

$$\lambda: \sum^* \times Q \to \sigma^*$$

Here, we should note that input and output may consist of multiple elements of $\sum$ and σ respectively.

q_0 : the initial or starting state. The IICIS will start from this state, which is an element of Q, i.e., $q_0 \in Q$. Now, we consider each state as a single processing unit where the actual processing on the input data/matrix is carried out. The output is produced in the form of output matrix, once the processing task is completed.

6.2. Description of IICIS components

The detailed description of all the components of IICIS is explained as follow:

(i) Set of states (Q)

Each state of IICIS can be considered as a discrete unit of processing in which the inputs and outputs are identified clearly. The processing carried out at each state can be represented by a method which will take an input and return the output.

(ii) The input/output sets $(\sum, \sigma)$

The input alphabet set $\sum$ is the set of all those elements which are given as input to IICIS at various states. Mathematically, $\sum = \{x \mid x$ is an alphabet taken as input by a state of IICIS $\}$

The elements of $\sum$ are matrices of order $i \times j \times k$.

$\sum = \{x \mid x_{i \times j \times k}$ is a matrix of order $i \times j \times k\}$

Let $A \in \sum$,

$\Rightarrow A_{1\times1\times1}$ is the simplest element in $\sum$, which is a real number.

If

$A \in \sum$, then $[A] = a_{l,m,n} = a(l, m, n) \forall l, m, n \in \{\mathbb{Z}^+ - 0\}$.

Here, $a_{l,m,n}$ is an element of matrix A belonging to the lth row, mth column and nth layer. The following conditions must hold for element ' a ':

- $0 < a_{l,m,n}$
- $a \in \mathbb{Z}^+$

The output alphabet set σ is the set of all those elements which are obtained as output after using various states of IICIS. The following are true for σ :

$\sum = \{x \mid x_{i\times j\times k}$ is a three dimensional matrix $\}$ If $A \in \sum$, then elements of matrix A can be denoted by $a_{i\times j\times k}$. Matrix A can be denoted by [A]

$$\Rightarrow [A] = a_{i\times j\times k} \qquad \forall i, j, k \in \{Z^+ - 0\}$$

i, j, k represents row, column and plane respectively.

The simplest element of σ will be $A_{1\times1\times1}$ which is a single element of $\{\mathbb{Z}^+ - 0\}$.

(iii) The state transition function (δ)

The state transition function directs the flow of control from one state to another. δ also determines the input required to move from one state to another and the output generated while transition is taking place. δ can be defined as follows:

$$\delta: \sum^* \times Q \to Q$$

In Eq. (3), $\sum^*$ is the set of all strings formed by the elements of $\sum$.

7. (iv) The output function (λ)

This function displays the output generated at each state. The definition of λ is as follow:

$$\lambda: \sum^* \times Q \to \sigma^* \mid \lambda: \sum^2 \times Q \to \sigma^2$$

In Eq. (4), $\sum^2 = \{x \mid x \leq 2$ and $|x| > 0\}$, and $\sigma^2 = \{y \mid y \leq 2$ and $|y| > 0\}$

The actual values of each tuple for the equations above will be represented by following equation above (Eq. (5)):

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$,

$\sum = \{I, I_G, I_{GBW}, I_{RSZ}, I_{EXT}, C, D, I_{EIG}\}$,

$\sigma = \{I_G, I_{GBW}, I_{RSZ}, I_{EXT}, C, D, I_{EIG}, M\}$, $\delta = \delta: \sum^2 \times Q \to Q$,

$$\lambda = \lambda: \sum^2 \times Q \to \sigma^2.$$

7.1. Significance of mathematical model with the actual machine

The processing tasks carried out by IICIS are divided into six algorithms. Each of these algorithms can be represented by the states of IICIS.

Now, if a $\in \sum^2$ and q $\in Q$, then $\delta(\mathrm{q}, \mathrm{a})$ shows the processing carried out on 'a' $\in \sum^2$, at state q $\in Q$. Here q represents an algorithm. Basically, δ maps an algorithm and an input symbol of $\sum^2$ onto the next algorithm, which is to be executed. Similarly, the output function σ maps an input and an algorithm to an output symbol.

The description of δ and λ functions will be as follows:

$$\begin{aligned}
\delta(\mathrm{q}_0, \{\mathrm{I}\}) &= \mathrm{q}_1, \lambda(\mathrm{q}_0, \{\mathrm{I}\}) \\
&= \{\mathrm{I}_{\mathrm{GBW}}, \mathrm{I}_{\mathrm{G}}\} \\
\delta(\mathrm{q}_1, \{\mathrm{I}_{\mathrm{GBW}}\}) &= \mathrm{q}_2, \lambda(\mathrm{q}_1, \{\mathrm{I}_{\mathrm{GBW}}\}) = \mathrm{C}, \\
\delta(\mathrm{q}_2, \{\mathrm{I}_{\mathrm{G}}, \mathrm{C}\}) &= \mathrm{q}_3, \lambda(\mathrm{q}_2, \{\mathrm{I}_{\mathrm{G}}, \mathrm{C}\}) = \{\mathrm{I}_{\mathrm{RSZ}}\}, \\
\delta(\mathrm{q}_3, \{\mathrm{I}_{\mathrm{RSZ}}, \mathrm{C}\}) &= \mathrm{q}_4, \lambda(\mathrm{q}_3, \{\mathrm{I}_{\mathrm{RSZ}}, \mathrm{C}\}) \\
&= \{\mathrm{I}_{\mathrm{EXT}}\}, \\
\delta(\mathrm{q}_4, \{\mathrm{I}_{\mathrm{EXT}}\}) &= \mathrm{q}_5, \lambda(\mathrm{q}_4, \{\mathrm{I}_{\mathrm{EXT}}\}) = \mathrm{D}. \\
\delta(\mathrm{q}_5, \{\mathrm{D}\}) &= \mathrm{q}_6, \lambda(\mathrm{q}_5, \{\mathrm{D}\}) = \{\mathrm{I}_{\mathrm{EIG}}\}. \\
\delta(\mathrm{q}_6, \{\mathrm{I}_{\mathrm{EIG}}, \mathrm{I}_{\mathrm{EIG}}\}) &= \mathrm{q}_7, \lambda(\mathrm{q}_6, \{\mathrm{I}_{\mathrm{EIG}}, \mathrm{I}_{\mathrm{EIG}}\}) \\
&= \{\mathrm{M}\}.
\end{aligned}$$

In Eq. (6), I is a True color image of three dimension, I_G is green component of image, $\mathrm{I}_{\mathrm{GBW}}$ is black and white conversion of green image, $\mathrm{I}_{\mathrm{RSZ}}$ is resized image, $\mathrm{I}_{\mathrm{EXT}}$ is extracted image having iris ring only, C is the matrix having coordinates of centre and approximate radius of iris, D is a three-dimensional matrix with size of $16 \times 16 \times$ m having size of 16×16 divided matrix of extracted image $\mathrm{I}_{\mathrm{EIG}}$ is a twodimensional matrix with size of $16 \times$ m, having Eigen values of all m number of 16×16 matrices, and M is a matrix containing match result of the two images.

7.2. Steps of IICIS

The Iris Image Compression and Identification System (IICIS) uses six steps for completing the task of online iris image compression and identification. We have designed these steps in the form of different programmes using MATLAB 7. The steps of IICIS are as follow:

Step 1: Getting the centre and radius of an iris image

This step reads the main m $\times n$ image and performs the analysis of this image to gauge the actual centre of iris according to the image pattern. It takes an arbitrary centre on the image and then traces the row from centre of image to both sides for reaching the iris border. After that, it fixes the centre on repeated examination of the border and calculates the radius of the iris image.

The input to Step 1 will be a $m \times n \times 1$ image, midpoint of the width of the image (gs), midpoint of the height of the image (hs), and the limit of continuous white pixels (lt).

The algorithm will have the following steps:

Algorithm getCircle( )

{

1. convert image [][]=

MGTOBLACKWHITE(image [][]);

2. d =GETDIMENSIONS(image[ ][ ]);

3. $g = gs$;

4. $h = hs$

5. $height = d(1)/2$;

6. $width = d(2)/2$;

7. $new\_matrixrads[d(2)][1] = 0$;

8. $wquad1 = 0$, wquad $2 = 0$, wquad $3 = 0$, wquad $4 = 0$;

9 .$bquad1 = 0$, bquad $2 = 0$, bquad $3 = 0$, bquad $4 = 0$;

10. limit = lt;

11. FOR $i = 0$ TO height-1 STEP 1

11.1. new_matrixrowX []= image[h-1][:];

11.2. new_matrixrowNX[] = image $[h + 1][:]$;

11.3. FOR $j = 0$ TO width-1 STEP 1

11.3.1. IF($wquad1 <$ limit)

11.3.1.1. IF $(\text{row}X[g + j] == 0)$

11.3.1.1.1. $bquad1 = bquad\ 1 + 1$

+wquad1;

11.3.1.1.2. wquad1 = 0;

11.3.1.2. ELSE IF (rowX[g + j] == 1)

11.3.1.2.1. wquad1 = wquad 1 + 1;

11.3.2. IF (wquad2 < limit)

11.3.2.1. IF (row I[g − j] == 0)

11.3.2.1.1. bquad2 = bquad 2 + 1 + wquad2;

11.3.2.1.2. wquad 2 = 0;

11.3.2.2. ELSE IF (rowX[g − j] == 1)

11.3.2.2.1. wquad2 = wquad 2 + 1;

11.3.3. IF (wquad1>=limit AND

wquad2 > =limit)

11.3.3.1. rads[i + 1] = (bquad1 +

bquad2)/2; 11.3.3.2. BREAK;

11.4. FOR j = 0 TO width-1 STEP 1

11.4.1. IF (wquad3 < limit)

11.4.1.1. IF (rowNX[g − j] == 0)

11.4.1.1. bquad3 = bquad 3 + 1 + wquad3;

11.4.1.1.2. wquad3 = 0;

11.4.1.2. ELSE IF (rowNX[g − j] == 1)

11.4.1.2.1. wquad3 = wquad 3 + 1;

11.4.2. IF (wquad4 < limit)

11.4.2.1. IF (rowNX[g + j] == 0)

11.4.2.1.1. bquad4 = bquad 4 + 1 + wquad4;

11.4.2.1.2. wquad4 = 0;

11.4.2.2. ELSE IF (rowNX[g + j] == 1)

11.4.2.2.1. wquad4 = wquad 4 + 1;

11.4.3. IF (wquad3>=limit AND

wquad4 >= limit)

11.4.3.1. rads[i + 2] = (bquad 3 +

bquad 4)/2

11.4.3.2. BREAK;

12. RETURN rads;

}

Step 2: Resize the image according to the iris ring

This step accepts the image, radius, and centre of iris image as inputs and plots circle around the iris ring [World Wide Iris Database, 2010]. Once the circle is plotted, it crops the image from all four sides such that the image border touches the iris ring from its outer circle on all the four sides.

The inputs to step- 2 will be a m × n × 1image[ ] [], outer radius of the iris (rad), centre of the iris (h, g) and it will return the image size (ims). The resizecen() function will have the following steps:

Algorithm resizecen( )

{

1. d =GETDIMENSION(image[ ][ ]);
2. x = h, y = g;
3. j = 0;
4. FOR i = y − radTOy + rad

4.1. j = j + 1;

4.2. IF (i < $d(2)$ AND i > 0)

4.2.1. new_matrixim[:][j]= image[:][i];

5. j = 0; 6. FOR i = x − radTOx + rad

6.1. j = j + 1;
6.2. IF (i < $d(2)$ AND i > 0)
6.2.1. new_matriximS [:][j] = im[i][:]

8. RETURN ims;

}

Step 3: Extracting the ring iris from the main image

This step takes the $m \times n \times 1$ image matrix, i.e., the green image, inner radius, outer radius of iris, and it produces an image of the same size which consists of only the iris ring. In this step, all the noise disturbances outside of the iris ring and inside of the iris ring are removed [Daugman, 1999, 2004].

The algorithm will have the following steps:

Algorithm extractIris( )

{

1. d =GETDIMENSIONS(image[ ][ ]);
2. x = ROUND(d(1)/2);
3. y = ROUND(d(2)/2);
4. peri = 2*3.14 * r;
5. new_matriximS[d(1), d(2)];
6. FOR i = 0 TO 2* PI STEP 1/r1

6.1. IF((r1 * sin(i) + x) > 0

AND(r1*cos(i) + y) > 0)

6.1.1. xi = ABS(r1 * sin(i) + x);

6.1.2. yi = ABS(r1*cos(i) + y);

6.1.3. IF (i >= 0 AND i <= (PI/2))

6.1.3.1. ims[xi: $d(1)$][yi: $d(2)$] = im[xi: $d(1)$]

[yi: d(2)]

6.1.4. ELSE IF (i>PI/2 AND i <= *PI*)

6.1.4.1. ims[xi: d(1)][1: yi] = im[*xi*: *d*(1)][1: *yi*]3*PI/2)

6.1.5. ELSE IF (i > *PI* AND i <= yi]

6.1.5.1. ims[1: xi][1: yi] = im[1: xi][1: 2 * PI)

d(2)]

6.1.6. ELSE IF (i>2*PI/3 AND i <=

6.1.6.1. ims[1: xi][*yi*: *d*(2)] = im[1: *xi*][*yi*:

7. FOR i = 0 TO 2* PI STEP 1/r

+y) > 0)

7.1. IF ((r*sin(i) + x) > 0 AND (r*cos(i)

7.1.1. xi = ABS(r*sin(i) + x);

7.1.2. yi = ABS(r*cos(i) + y);

7.1.3. IF (i >= 0 AND i <= (*PI*/2)) 7.1.3.1. ims [xi: d(1)][yi: d(2)] = 0;

7.1.4. ELSE IF (i>PI/2 AND i <= *PI*)

7.1.4.1. ims[xi: d(1)][1: yi] = 0; 3*PI/2)

7.1.5. ELSE IF (i > *PI* AND i <=

7.1.5.1. ims[1: xi][1: yi] = 0;

2*PI)

7.1.6. ELSE IF (i>2*PI/3 AND i <=

7.1.6.1. ims[1: xi][yi: d(2)] = 0;

8. RETURN ims;

}

Step 4: Dividing iris ring into square matrices

This step reads the iris image having the resized iris ring and then it divides the iris ring into smaller square matrices. The dimensions of the new matrices are also specified in step 4.

The input to step 4 will be the resized image and dimensions of the new smaller matrices (dim). This algorithm will have the following steps:

Algorithm divMat( )

{

1. FOR i = dim TO image_width STEP dim

1.1. FOR j = dim TO image_height STEP dim

1.1.1. x[:][:] = rszimg[i − (dim − 1): i]

[j − (dim − 1): j];

1.1.2. IF (ANYELEMENT (x) > 0)

1.1.2.1. k = k + 1

1.1.2.2. NEWIMAGE imr[:][:][k] = x;

2. RETURN imr[][][];

}

Step 5: Calculating Eigen values

This step takes the pixel matrix as input and calculates the Eigen values for each matrix. These Eigen values will be stored in another matrix. The Eigen value calculation matrix will have following steps:

Algorithm calEig( )

{

1. d = GETDIMENSION(mat[][][]);

2. accumulator j = 1;

3. FOR i = 1TOd STEP 1

3.1. z = EIGEN(mat[:][:][i]);

3.2. new_matrixmodz []=

ABSOLUTE(z); 3.3. new_matrixegmat[1: 16][i] = modz;

4. RETURN egmat;

}

Example: If the actual image matrix (544×544)

is divided into smaller matrices of the order 16×16, then the matrix (mat) will contain all the 16×16 matrices in the three dimensions as $16 \times 16 \times 34$.

Step 6: Comparing two iris images using the Eigen values

This step compares the Eigen values of the stored image with the online calculated Eigen values. The algorithm for the comparison of the stored Eigen values of an iris image with its online calculated Eigen values will be as follows:

Algorithm compareEig( )

{

1. eigmat1=CONVERT2DMATTO1DMAT (eigmat1);
2. eigmat2 = CONVERT2DMATTO1DMAT (eigmat2);
3. hit = 0, mis = 0, found = 0, tot = 0;
4. d1 =GETDIMENSION(eigmat1);
5. d2 =GETDIMENSION(eigmat2);
6. FOR i = 1TOd1(2) STEP 1

6.1 . found = 0;

6.2. FOR j = 1TOd2(2) STEP 1

6.2.1.IF (eigmat1 [i] == eigmat 2[i][j])

6.2.1.1. found = 1;

6.2.1.2. BREAK;

6.3. IF (found == 1)

6.3.1. hit = hit +1 ;

6.4. ELSE

6.4.1. mis = mis + 1;

6.5. tot = tot +1

7. hitper= ((hit/tot)*100)

misper= ((mis/tot)*100);

8. comparison.hitPer=hitper;

9. comparison.misPer= misper;

10. comparison · eig ValHit = hit;

11. comparison.eigValMis= mis;

12. comparison totEigVals= tot;

13. RETURN comparison;

}

Steps 1 to 6 represent a system called Iris Image Compression and Identification System (IICIS) which computes online Eigen values of an iris image of a person and compares these

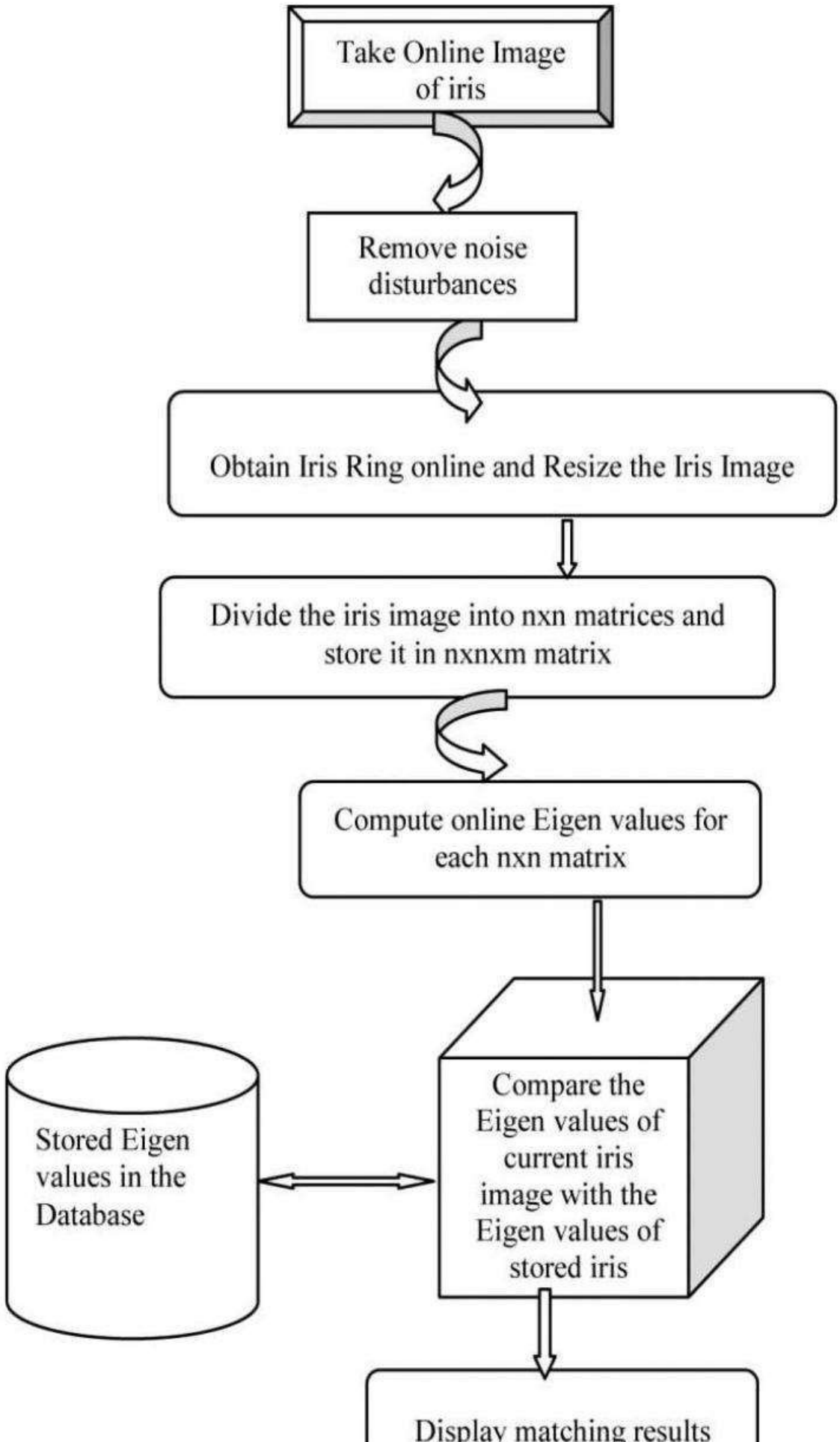
Take Online Image of iris
Remove noise disturbances
Obtain Iris Ring online and Resize the Iris Image
Divide the iris image into nxn matrices and store it in nxnxm matrix
Compute online Eigen values for each nxn matrix
Stored Eigen values in the Database
Compare the Eigen values of current iris image with the Eigen values of stored iris
Display matching results

Fig. 5. Online Identification of Iris Image using Eigen values.

Eigen values with the stored Eigen values. In the next phase (Experimental Analysis of Results phase), we will analyze the performance of our algorithms for CASIA database.

The complete procedure for online identification of a person using Eigen values can be explained by the flowchart as shown in Fig. 5 . The flowchart of Fig. 5 shows the steps of comparing online calculated Eigen values of an iris image with the stored Eigen values. The performance of IICIS for an individual and identical twins are explained in the result analysis phase.

9. Experimental Analysis of Results with Actual Data

In this section, we have displayed the results obtained at the various steps with the execution Table 1. Sample images and its size in KBs.

S. No.	Image	Dimensions	Size (in KB)
1	p1le1	576×768	448
2	p2re2	576×768	468
3	p3le1	576×768	456

of our procedure mentioned in IICIS phase. In order to verify our procedure, we have used three sample images obtained from three different individuals of CASIA database. The details of these images are as mentioned in Table 1.

We started with a consideration of the raw image. The raw image is a true color image which is taken with the help of a high-resolution camera. It is the image represented as a combination of three primary colors: Red, Green and Blue, i.e., RGB. A computing device interprets this image as a three-dimensional matrix, with each dimension representing one of the three primary color components R, G or B. A pixel comprises of some combination of values of these three primary color components [Iridian Technologies, 2009]. The raw images of three samples listed in Table 1 are shown in Fig. 6 .

After obtaining the original true color image in the three-dimensional matrix format, one of the three-dimensions is extracted and we proceeded with the noise disturbance removal, as

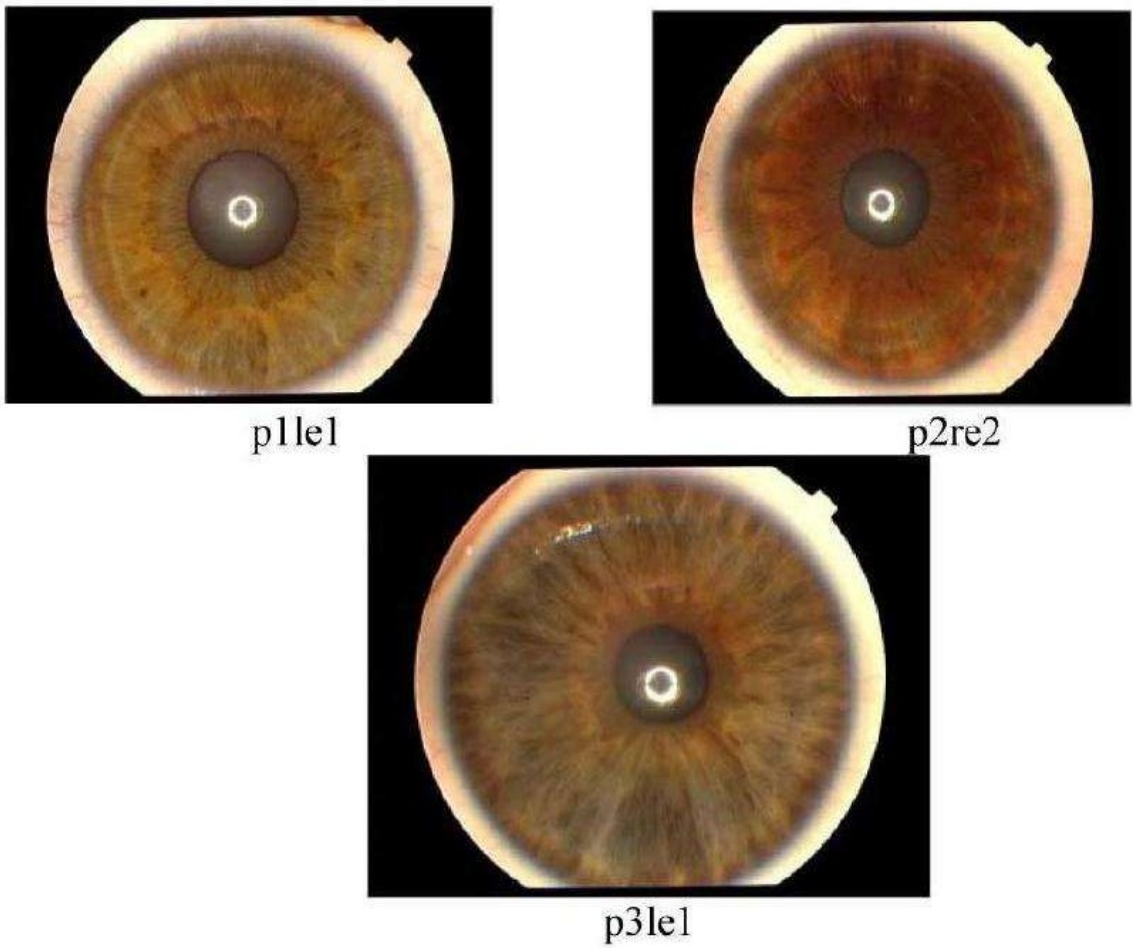

Fig. 6. The raw images of three samples. explained in the theoretical background. After the removal of noise disturbances, we calculated the centre and radii of the iris ring. The results obtained while calculating the centre and radii of the iris ring for sample images of Table 1 are shown in Table 2.

Now, we divide the resultant image matrix into smaller matrices of dimensions 16 × 16. The segmented iris ring obtained after removing the pupil using our method will be like Fig. 7 [Balaji, 2006]. The pixels representing Image_Mat_1, Image_Mat_2, and Image_Mat_3 are shown in Fig. 8.

After dividing the actual iris ring into smaller matrices of size 16 × 16, we calculate the Eigen values. For each 16 × 16 matrix, we got 16 Eigen values. These Eigen values are floating

Table 2. Centre and radius of image sample.

S. No.	Image	Centre		Radius (in pixels)	Time taken (in ms)
		G	H		

1	p1le1	389	293	278	1.478
2	p2re2	387	280	274	0.796
3	p3le1	439	314	280	1.418

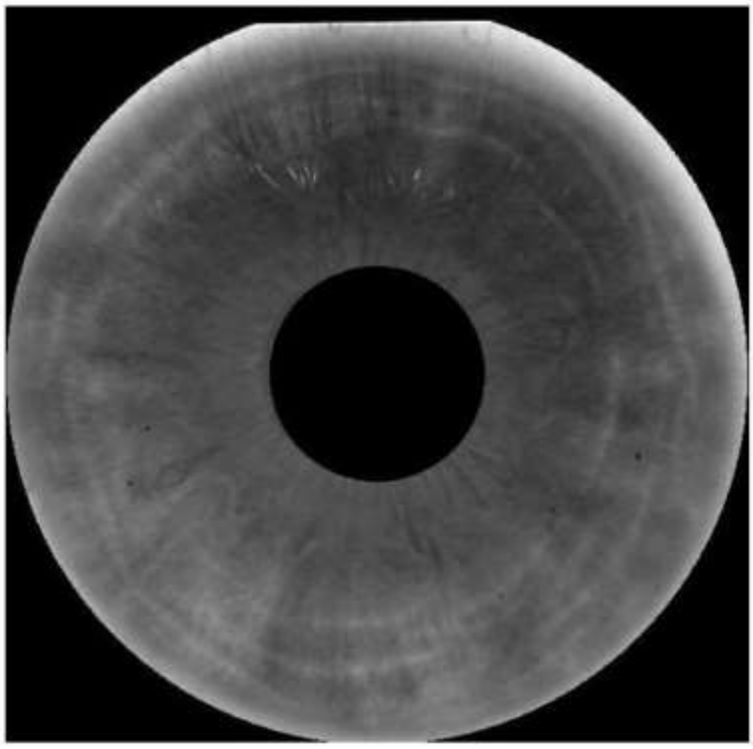

Fig. 7. Segmented Iris Ring obtained after removing pupil.

Image_Mat_1 (286th matrix of p1le1)

{{73,75,81,79,76,78,80,79,79,78,75,73,79,83,81,80},
{75,76,75,74,77,82,81,81,80,78,80,83,81,81,84,84},
{76,79,79,79,79,79,80,80,79,79,79,80,82,85,88,87},
{75,77,82,84,81,81,84,81,79,81,82,83,83,87,86,85},
{80,81,79,79,78,76,82,86,81,80,83,84,82,83,85,83},
{78,80,79,79,80,79,78,81,79,81,83,81,81,81,82,84},
{78,81,81,78,77,78,77,77,80,81,78,77,77,77,76,79},
{79,83,84,83,84,81,81,81,81,81,80,81,82,80,76,77},
{84,85,85,80,80,80,80,83,81,82,85,84,83,80,80,83},
{85,85,82,80,79,82,87,84,83,87,86,81,82,85,84,84},
{85,84,83,82,80,82,83,84,84,83,83,83,83,86,87,89},
{84,81,81,80,81,80,79,84,82,83,86,86,87,88,86,86},
{82,82,79,76,79,85,88,88,87,87,91,91,89,87,87,92},
{81,78,80,79,78,77,81,85,85,86,85,86,89,89,90,94},
{82,83,83,82,81,80,82,85,87,87,87,86,86,89,92,93},
{81,82,82,82,80,83,84,82,84,85,87,86,87,88,88,88}}

Image_Mat_2 (842nd matrix of p2re2)

{{99,95,96,96,91,85,85,85,85,84,83,79,75,78,80,80},
{99,95,93,95,89,82,82,85,84,83,81,80,80,78,76,78},
{97,95,91,90,92,90,88,85,81,81,83,80,77,75,76,76},
{100,99,99,99,95,88,86,85,83,83,83,86,83,79,77,77},
{109,109,106,103,100,94,91,90,87,85,90,92,87,82,78,
72},{109,113,118,119,111,102,94,92,97,99,96,89,86,
86,85,82},{113,118,122,126,119,108,101,95,102,114,
118,101,94,90,87,88},{113,116,123,130,121,104,101,
105,111,138,155,116,92,88,87,91},{106,110,115,114,
112,106,98,103,111,136,156,128,103,92,90,95},{103,
102,104,103,101,95,93,99,107,119,132,132,107,89,87,92},
{102,102,102,97,94,95,92,90,97,103,101,105,101,92,87,
92},{100,96,98,96,90,88,90,90,92,96,95,95,95,96,95,96},
{102,101,101,99,94,94,93,88,92,94,94,94,97,99,99,97},
{102,100,100,96,98,97,97,96,97,97,96,95,94,95,99,104},
{105,102,102,102,99,102,101,97,98,103,100,98,98,101,
106,107},{108,107,104,102,101,102,105,107,114,112,
106,104,105,107,107,107}}

Image_Mat_3 (542nd matrix of p3le1)

{{71,66,67,72,72,69,72,74,71,71,73,75,76,78,76,72},
{74,70,69,75,77,72,71,72,72,72,72,72,78,81,76,70},
{73,74,76,78,78,75,74,75,74,73,71,74,82,84,79,76},
{73,75,79,82,78,73,73,72,70,68,71,74,77,78,74,73},
{72,74,74,76,76,75,79,78,72,69,70,74,80,80,78,74},
{68,73,76,75,74,72,75,77,72,71,73,75,79,82,78,73},
{72,75,73,74,76,75,77,74,73,71,72,77,80,82,79,72},
{74,79,79,74,74,75,75,75,73,74,75,77,81,79,75,73},
{71,79,79,75,74,77,76,71,74,80,79,78,79,75,72,71},
{72,75,78,75,76,79,79,74,75,71,71,73,77,79,77,74},
{70,77,77,73,74,79,79,74,71,74,77,79,84,84,78,72},
{72,73,76,74,72,74,75,74,74,75,77,78,81,79,74,73},
{77,81,79,72,75,80,76,75,77,74,73,77,83,82,73,69},
{75,73,73,76,76,78,78,75,72,72,76,78,76,73,73,72},
{79,77,79,76,78,83,82,75,69,68,73,79,81,78,72,72},
{79,77,74,73,79,87,85,75,72,70,73,81,87,79,73,73}}

Fig. 8. Pixels representing Image_Mat_1, Image_Mat_2, and Image_Mat_3.

$$\text{Eigen\_Values\_Image\_Mat\_1} = \{1.3136, 0.0096, 0.0096, 0.0085, 0.0059, 0.0059, 0.0040, 0.0032, 0.0038, 0.0038, 0.0056, 0.0056, 0.0045, 0.0045, 0.0022, 0.0018\}$$

$$\text{Eigen\_Values\_Mat\_2} = \{1.5543, 0.0603, 0.0293, 0.0293, 0.0184, 0.0112, 0.0099, 0.0065, 0.0065, 0.0033, 0.0033, 0.0021, 0.0020, 0.0012, 0.0012, 0.0001\}$$

$$\text{Eigen\_Values\_Image\_Mat\_3} = \{1.2036, 0.0201, 0.0092, 0.0092, 0.0072, 0.0072, 0.0059, 0.0059, 0.0056, 0.0035, 0.0035, 0.0010, 0.0028, 0.0028, 0.0013, 0.0013\}$$

Fig. 9. Eigen values for image matrices.

point numbers. The Eigen values obtained for Image_Mat_1, Image_mat_2, and Image_Mat_3 are shown in Fig. 9.

The results of calculating the Eigen values for image matrices are listed in Table 3.

In the below mentioned table (Table 3), TM represents the Total Matrices obtained for the iris images p1le1, p2re2 and p3le1, where the size of each matrix is 16 × 16. The result shows that we need less than 1 second of time to calculate all the Eigen values of an iris image and we need approximately 1.25 seconds to compare online calculated Eigen values with the stored Eigen values. We have stored Eigen values of an iris image for further identification of a person. The memory required for storing the Eigen values of iris images p1le1, p2re2, p3re1 and its comparison with actual memory required for storing these iris images are given in Table 4.

The robustness of comparing the two iris images will have following the three steps:

Table 3. Calculating Eigen values and its comparison with stored Eigen values for iris images.

S. No.	Image	TM	Time required for dividing image into matrices (in ms)	Eigen value calculation time (in ms)	Iris image comparison time (in ms)

1	p1le1	834	25.40	9.93	125.83
2	p2re2	870	26.0	9.68	121.49
3	p3le1	670	17.9	7.97	135.56

Table 4. Memory required in kb for storing iris images and Eigen values.

		Original size of true color image (in KB)	Size after compression (in KB)	Compression ratio
1	p1le1	448	52.13	8.59
2	p2re2	468	51.06	9.16
3	p3le1	456	41.87	10.88

9.1. Comparison of two iris images

All the procedures above (step 1 to 6) have been carried out to accomplish a successful comparison between the already stored Eigen values of a person and the newly calculated Eigen values from the Iris image of the same person so that the identification of person can be uniquely verified. In order to carry out comparison, we took the Iris image (I1) of a person whose identity is to be verified and applied the above procedures (Step 1 to 6) for converting it into Eigen values. Let these calculated Eigen values of Image I1 be known as E1. Let the already stored Eigen values of the same person be known as E2. Now, E1 and E2 were compared to verify the identity of the person. When comparing the two sets of Eigen values E1 and E2, we compared the first Eigen value of E1 with all the Eigen values of E2, and then the second Eigen value of E1 was compared with all the Eigen values of E2. This procedure was repeated until all the Eigen values of E1; compared with the Eigen values of E2.

9.2. Match percentage

The number of Eigen values which are common in two iris images are known as Match Percentage. Depending on this match percentage, we can verify whether the two images (i.e., the one stored already and the one taken freshly from the camera) are exactly the same or not. We have carried out a series of Robustness Tests to get a basic idea of how the match percentage varies with the increasing amount of noise in the image. The robustness test is explained in the next phase.

10. Discussion of Robustness Testing in Iris Image Compression

The robustness testing determines the application of our IICIS for its practical use. The result analysis is incomplete without checking its robustness. Robustness means the extent to which our algorithm can withstand the corresponding changes or modifications in the input image. We have tested the robustness of the image by replacing randomly chosen pixels of the image with random values. If ' I ' is the image on which the robustness test is to be carried out, then the overall test will be carried out as per the following steps:

S1: Replace n pixels of ' I ' to get the new image 'I1'.

S2: Compare 'I' with 'I1'.

S3: Record the comparison results in tabular form.

S4: Repeat the steps S1, S2, and S3 for different values of $n(n = 100,200,300$ and 500).

The procedure which we have used to replace the pixels of an iris image randomly has the following inputs:

- ims - the image in which the pixels are to be replaced/changed. The image must be a two-dimensional image.
- The number of pixels to be replaced (n).
- The segment of the image to be altered (segment).
- The image is divided into four quarters (q for whole image, q1 - for quadrant1, q2 for quadrant2, q3 - for quadrant3, q4 - for quadrant4).

The output of our procedure is the altered iris image. The procedure above randomly selects ' n ' pixels in the image and then replaces them with random values. The random values will be between 0 and 255 . The IICIS algorithm divides actual image matrix into small 16×16 blocks and then the Eigen values of these blocks are calculated. Therefore, the distribution of these randomly modified pixels throughout the image plays an important role in calculating match percentage for two images (actual image and randomly modified image). Our analysis is divided into two cases. These cases are as follow:

Case 1: If the changed pixels are confined to a small region of the image.

Case 2: If the changed pixels are distributed throughout the image.

In Case 1, the changes were confined only to one of the four quadrants (i.e., q1, q2, q3 or q4) of any given input image. The tests were carried out on three sample images of Table 1 and comparison results were recorded. In the first image of Table 1 (i.e., p1le1) we changed the first quadrant (q1) and kept the rest of the image unaltered. In the second image (p2re2) of Table 1, we altered the pixels of second quadrant (q2). In the next test, we changed the pixels of the third quadrant (q3) and

the fourth quadrant (q4) of the third image (p3le1) and left the rest of the image unaltered. If n represents the number of pixels to be altered, then the procedure above is repeated for four different values of $n(n = 100, 200,300,500)$ for verifying the accuracy of our algorithm in robustness testing. This is for each time we compared the altered image with the actual image and stored result in tabular form.

In Case 2, we took the whole image as a single unit and randomly modified the pixels throughout the iris image. We again recorded the results of the comparison of the iris images of Table 1 and their corresponding modified images. The results of robustness testing of our system are shown in Table 5.

In Table 5, the total number of Eigen values for different images are different. This is due to the fact that each image belongs to a different person and these images of different persons are different in size after resizing. The reasons for getting different number of 16×16 matrices for

Table 5. Results of robustness testing.

S. No.	Image	Quadrant/ portion of image	Total Eigen values	No. of pixels modified	No. of Eigen valu affected by pixe modification	% of Eigen values affected	% of Eigen values unaffected	Hit percent
1	P1le 1	Q1	1334 4	100	719	5.38	94.61	94.61
2	P1le 1	Q1	1334 4	200	1136	8.51	91.48	91.48
3	P1le 1	Q1	1334 4	300	1605	12.02	87.97	87.97
4	P1le 1	Q1	1334 4	500	2285	17.12	82.87	82.87
5	P2re 2	Q2	1307 2	100	560	4.28	95.71	95.71
6	P2re 2	Q2	1307 2	200	1104	8.44	91.55	91.55
7	P2re 2	Q2	1307 2	300	1656	12.66	87.33	87.33

8	P2re 2	Q2	1307 2	500	2315	17.70	82.29	82.29
9	P3le 1	Q3	1296 0	100	608	4.69	95.30	95.30
10	P3le 1	Q3	1296 0	200	1024	7.90	92.09	92.09
11	P3le 1	Q3	1296 0	300	1407	10.85	89.14	89.14
12	P3le 1	Q3	1296 0	500	2016	15.55	84.44	84.44
13	P3le 1	Q4	1296 0	100	784	6.04	93.95	93.95
14	P3le 1	Q4	1296 0	200	1276	9.84	90.15	90.15
15	P3le 1	Q4	1296 0	300	1769	13.64	86.35	86.35
16	P3le 1	Q4	1296 0	500	2428	18.73	81.26	81.26
17	P1le 1	Q	1334 4	100	800	5.99	94.00	94.00
18	P1le 1	Q	1334 4	200	1328	9.95	90.04	90.04
19	P1le 1	Q	1334 4	300	2127	15.93	84.06	84.06
20	P1le 1	Q	1334 4	500	3235	24.24	75.75	75.75
21	P2re 2	Q	1307 2	100	640	4.89	95.10	95.10

22	P2re 2	Q	1307 2	200	1417	10.83	89.16	89.16
23	P2re 2	Q	1307 2	300	2138	16.35	83.64	83.64
24	P2re 2	Q	1307 2	500	3111	23.79	76.20	76.20
25	P3le 1	Q	1296 0	100	659	5.08	94.91	94.91
26	P3le 1	Q	1296 0	200	1356	10.46	89.53	89.53
27	P3le 1	Q	1296 0	300	1809	13.95	86.04	86.04

two different people's iris images after dividing it into matrices are as follows:

- The iris size of an eye is different for different persons. In our experiments, we found that generally, the outer radius of iris ring for CASIA and multimedia university databases lie in the range of 270 pixels to 290 pixels.

- The sizes of the pupils of eyes of different persons are different, and we are discarding the pupil during the extraction phase. Therefore, we are getting different number of 16×16 matrices for two different people's iris images. But, for two different iris images of the same, we are getting the same number of 16×16 matrices.

- The size of pupil is inversely proportional to the number of matrices. Since the number of 16×16 matrices varies for different iris images, we get different number of Eigen values for different persons' iris images.

Finally, we have represented the observations in the form of bar graphs. In these graphs, the number of modified pixels is represented by X axis and the percentage of matched Eigen values are represented by Y-axis. We have drawn a separate graph for each test image. Case 1 testing contributed to four graphs and Case 2 contributed to three graphs. These graphs are represented by Figs. 10-16.

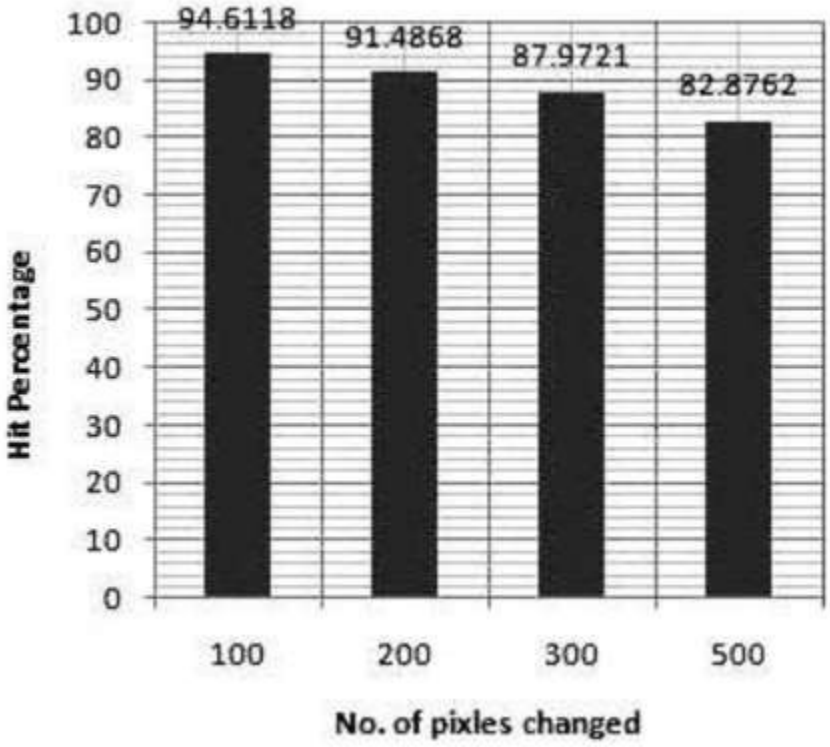

Fig. 10. Robustness testing results obtained after altering the pixels of Q1 for image p1le1.

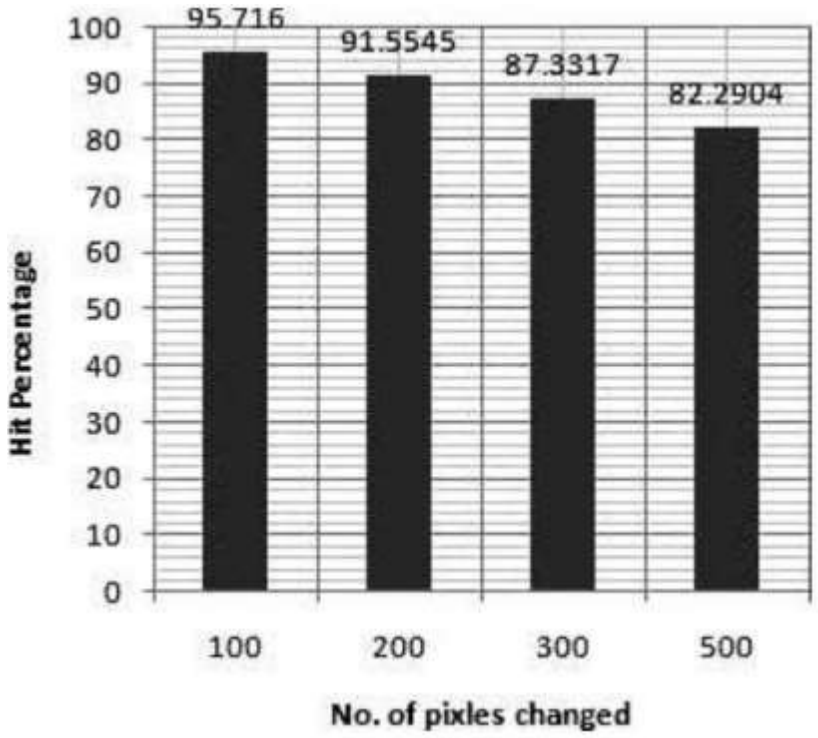

Fig. 11. Robustness testing results obtained after altering the pixels of Q2 for image p2re2.

Based on the above mentioned bar graphs (Figs. 10-16), we have concluded that the match percentage is inversely proportional to n (the number of pixels modified). Figures 10-16 show that when n was 0 , the match percentage was 100%. But, when n became 50 , the match percentage started decreasing and finally, it became zero. While testing our IICIS software, we have

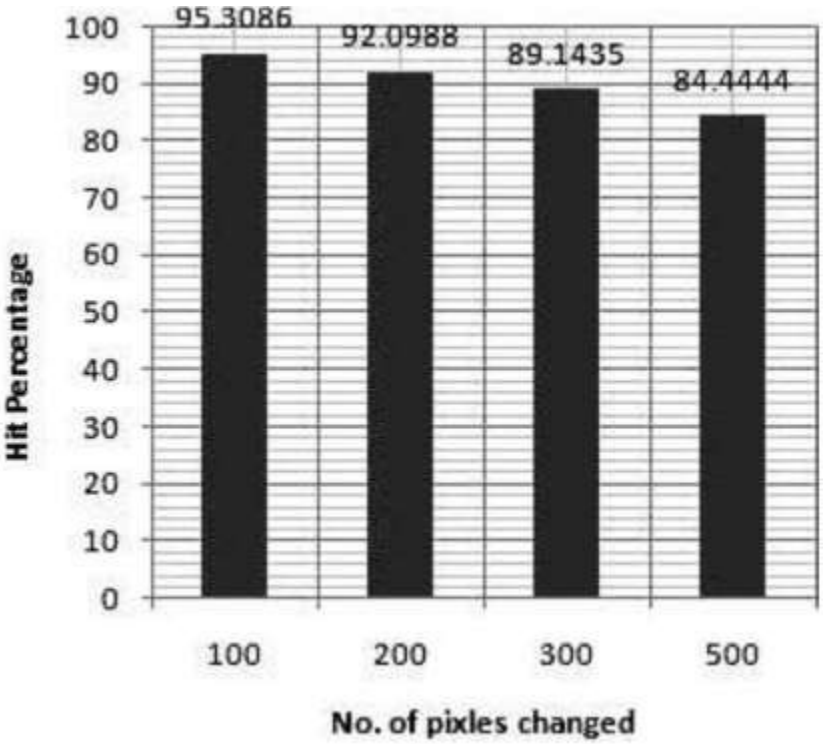

Fig. 12. Robustness testing results obtained after altering the pixels of Q3 for image p3le1.

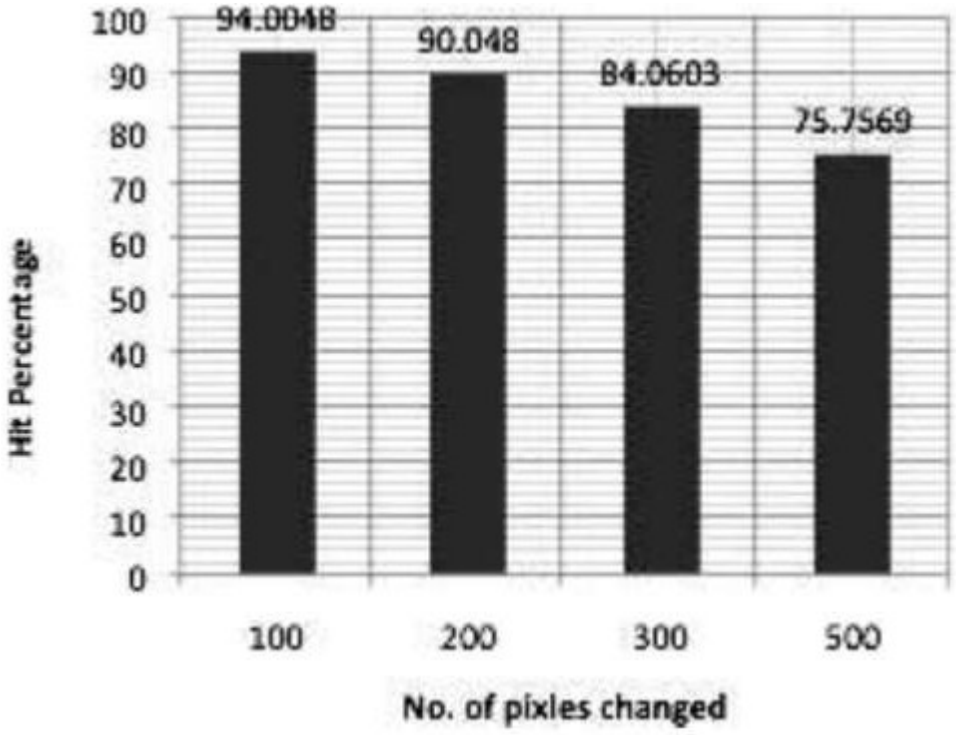

Fig. 13. Robustness testing results obtained after altering the pixels of Q4 for image p3le1.

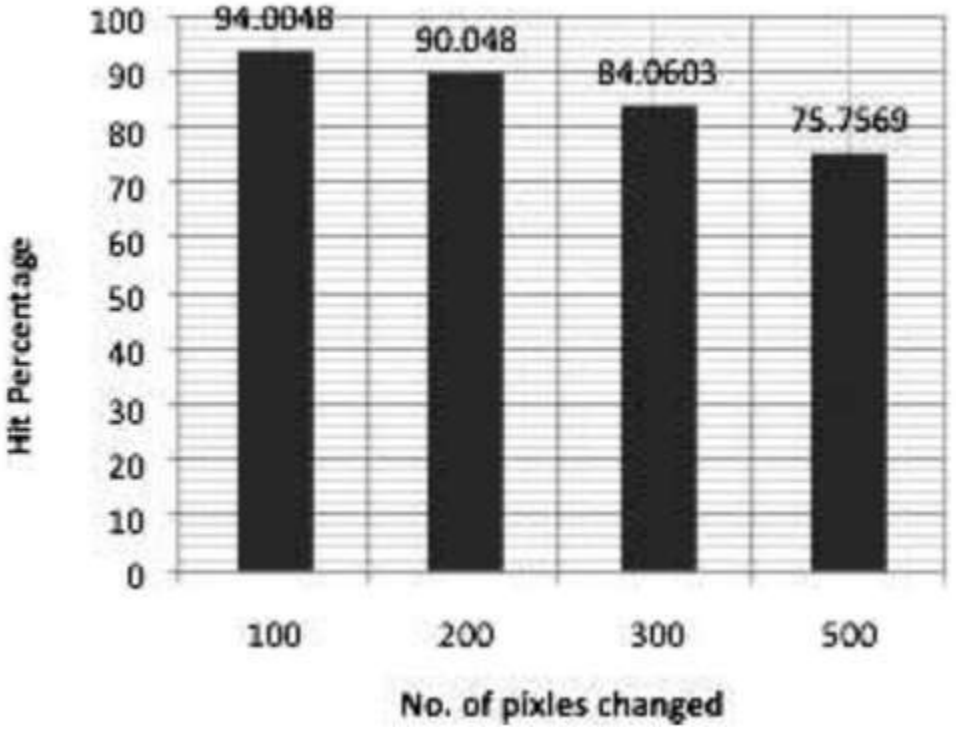

Fig. 14. Robustness testing results obtained after altering the pixels of complete image for image p1le1.

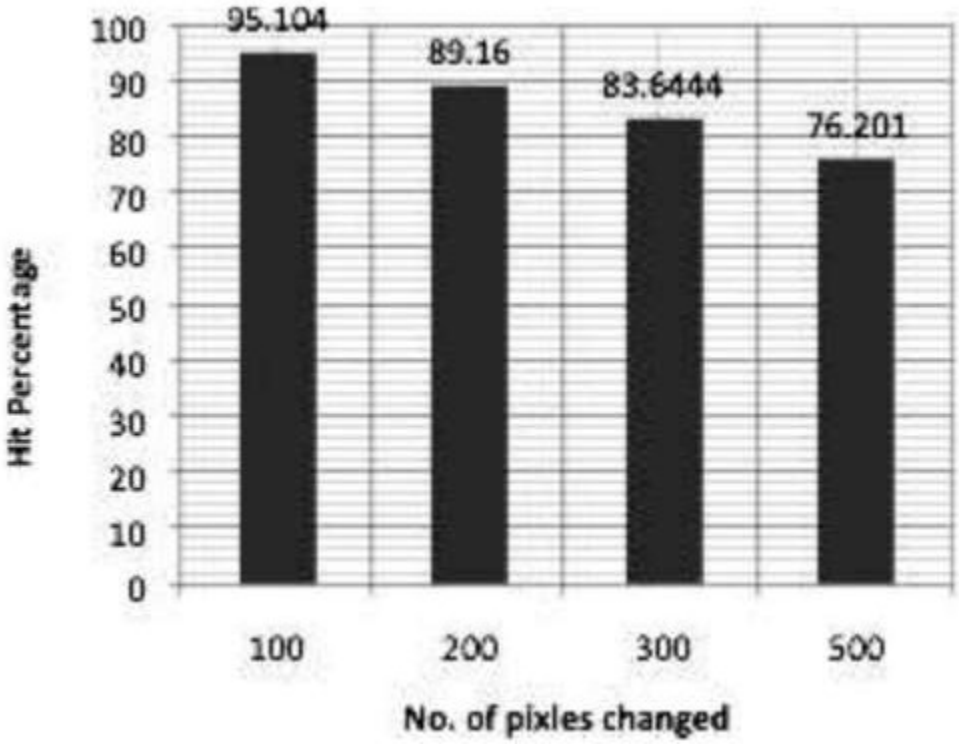

Fig. 15. Robustness testing results obtained after altering the pixels of complete image for image p2re2.

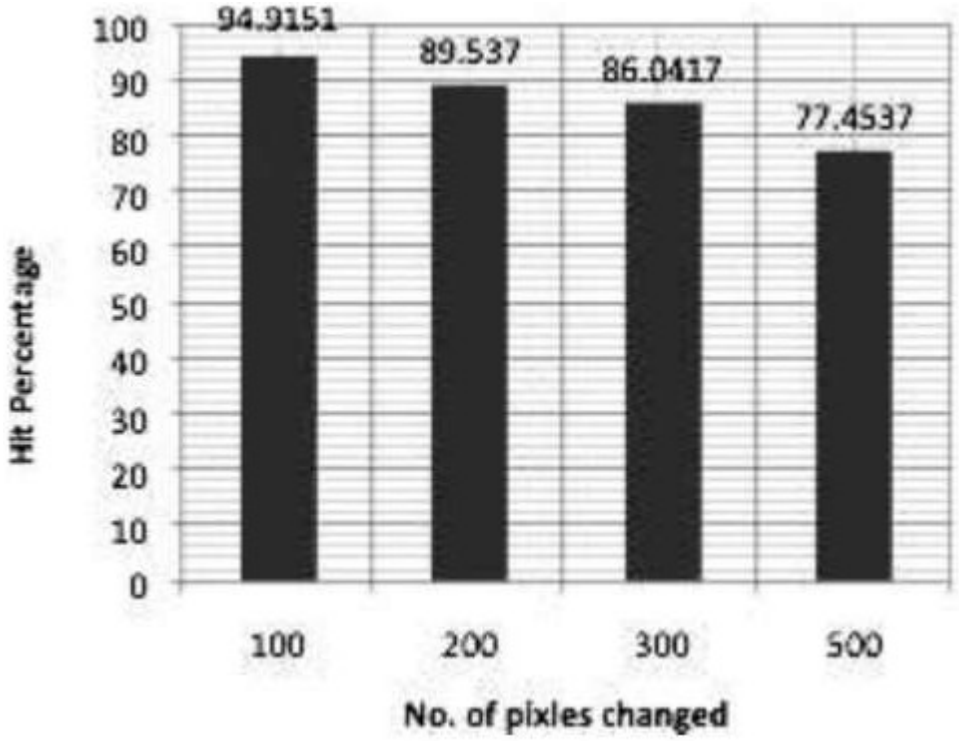

Fig. 16. Robustness testing results obtained after altering the pixels of complete image for image p3le1.

assumed a few constraints. These constraints are:

- The distance of the iris image from the camera will remain constant while repeating the procedure.
- All the iris images are taken from the same angle.
- The intensity of light does not change drastically while taking the iris image.

11. Conclusion and Future Work

In this paper, we have introduced a new method called IICIS. This method can identify an individual and identical twins with an accuracy of 99.99%. It means that for every 10,000 persons, our IICIS will correctly identify 9999 persons. Our algorithm has used six steps for personal identification of an individual or identical twins. Using these steps, we have tested the iris images of individuals and identical twins for CASIA database. In our testing, we found that our algorithm is able to store an iris image of 450KB in the form of Eigen values which uses approximately 50KB of memory. These Eigen values which were stored in 50KB memory were sufficient to identify a person or identical twins.

To check the robustness of our algorithm, we have altered 100, 200, 300, and 500 pixels of the iris image randomly and then compared the Eigen values of the altered iris image with the Eigen values of the actual iris image. In this comparison, we have observed the following results:

Result 1: If maximum altered pixels are lying in the first few 16×16 matrices, we found that the altered pixels are not affecting the identity of a person and by more than 99.50% the Eigen values of the altered iris image are the same as the Eigen values of the actual iris image.

Result 2: If all the altered pixels are scattered such that each pixel is in a separate 16×16 matrix, our IICIS software has given promising result. In the worst case scenario, when the Eigen values of these 100,200,300, and 500 pixels of the altered iris image were not same as the Eigen values of the actual image. In this case we also found more than 90% of the Eigen values of the altered iris image and the actual iris image are the same.

Result 3: If a few of the altered pixels are lying in the first few 16×16 matrices and others are scattered in other matrices, we found that approximately 97% Eigen values of the altered image are exactly the same as the Eigen values of the actual image.

The researchers can develop a technique for human identification which can include the cases of these constraints. In our algorithm, we have converted the iris image in the form of Eigen values for further human identification. These Eigen values are sufficient for personal identification of an individual and identical twins. But we could not get the actual image from the Eigen values. The researchers can develop a method for obtaining actual iris image from the Eigen values.

13. References

Belkin, M. and Niyogi, P. [2003] "Laplace eigenmaps for dimensionality reduction and data representation," Neural Network Compute 15, 1373-1396.

Bowyer, K. W., Hollingsworth, K. P. and Flynn, P. J. [2008] "Image understanding for iris biometrics: A survey," Computer Vision and Image Understanding 110, 281-307.

CASIA [2010] "Iris image database - iris recognition research group national laboratory of pattern recognition (NLPR)," Institute of Automation. Chinese Academy of Sciences. http://www.sinobiometrics.com

Daugman, J. [1999] Recognizing Persons by Their Iris Patterns. Biometrics: Personal Identification in Networked Society (Kluwer Academic Publishers).

Daugman, J. [2004] "How iris recognition works," IEEE Transactions On Circuits and Systems for Video Technology 14(1), 21-30.

Daugman, J. [2007] "New methods in Iris recognition," IEEE Transactions on Systems, Man, Cybernetics 37(5), 1167-1175.

Doroteo, T. and Toledano [2006] "Usability evaluation of multi-modal biometric verification systems," Interacting with Computers 18, 1101-1122.

Folm, L. and Safir, A. [1987] "Iris recognition system," US Patent No. US4641349.

Ganeshan, B., Theckedath, D., Young, R. and Chatwin, C. [2006] "Biometric iris recognition system using a fast and robust iris localization and alignment procedure," Optics and Lasers in Engineering 44, Elsevier, 1-24.

Iridian Technologies [2009], http://www.iriscan.com. Jain, A., Bolle, R. and Pankanti, S. [1999] Biometrics: Personal Identification in Network Society (Springer Publisher), pp. 369-384.

Karni, Z. and Gotsman, C. [2000] Special Compression of Mesh Geometry (Computer Graphics Proceedings, ACM Press, New York), pp. 279-286.

Ma, L., Yunhong, W. and Tieniu, T. [2002] "Iris recognition using circular symmetric filters," in Proceedings of International Conference on Pattern Recognition 2, 414-417.

Mattey, J. R., Broussard, R. and Kernnell, L. [2010] "Iris image segmentation and sub-optimal images," Image and Vision Computing 28(1), 215 – 222. Mishra, K. N., Agrawal, A., Srivastav, P. C. and Hadi, E. A. [2010] "An efficient horizontal and vertical method for online DNA sequence compression and identification," International Journal of Computer Applications, Foundation of Computer Science 3(1), 39-45.

Reuter, M., Franz-Erich, W. and Peinecke, N. [2006] "Laplace-Beltrami spectra as shape DNA of surface and solids," The Journal of Computer Aided Design 38, Elsevier, 342-366.

SchonBerg, D. and Kirovski, D. [2004] "Iris compression for cryptographically secure person identification," IEEE Computer Society, Proceedings of the Data Compression Conference, 459- 468.

Shih, S.-W., Chen, W.S., Lin, L.Y. and Lio, J.C. [2002] "Automatic iris recognition technique based on divider dimension and Karhunen-Loeve transform," in Proceedings of Conference on Computer Vision, Graphics and Image Processing (CVGIP), 78-85.

Tisse, C. [2003] "Person identification technique using human iris recognition," Journal of System Research 4, 67-75.

Turk, M. A. and Pentland, A. P. [1991] "Face recognition using Eigen faces," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 586-591.

Wildes, R. P. [1997] "Automated iris recognition: An emerging biometric technology," Proceedings of IEEE 85(9), 1348-1363.

World Wide Iris Database [2010] www. advancedsourcecode.com/irisdatabase.asp