U.S.N.A. --- Trident Scholar project report; no. 331 (2005)

Non-orthogonal Iris Segmentation

by

MIDN 1/C Bradford L. Bonney, Class of 2005
United States Naval Academy
Annapolis, MD

_____
(signature)

Certification of Adviser's Approval

Professor Delores M. Etter
Electrical Engineering Department

_____
(signature)

_____
(date)

Assistant Professor Robert W. Ives
Electrical Engineering Department

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Deputy Director of Research & Scholarship

_____
(signature)

_____
(date)

USNA-1521-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	9 May 2005	

4. TITLE AND SUBTITLE
Non-orthogonal iris segmentation

5. FUNDING NUMBERS

6. AUTHOR(S)
Bonney, Bradford Ludwig, 1982

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
US Naval Academy Annapolis, MD 21402		Trident Scholar project report no. 331 (2005)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
This document has been approved for public release; its distribution is UNLIMITED.	

13. ABSTRACT: The goal of this Trident Scholar project was to isolate the iris, the colored part of the eye, in a non-orthogonal, digital image of the human eye. A non-orthogonal image is an image where the eye is not looking directly at the camera. Iris pattern differs significantly between individuals (including identical twins), which allows for its use as an accurate biometric identifier. Both commercial and research iris recognition systems are becoming widespread in government and industry for logical security and access control. These iris recognition systems assume that captured iris images are normal, or orthogonal, to the sensing devices, and therefore search for circular patterns in the image. Off-angle, or non-orthogonal, images of irises cannot currently be used for identification because the iris appears elliptical; commercial algorithms cannot isolate an elliptical iris in order to start the identification process. This research expanded the functionality of iris recognition technology by developing a set of new algorithms to isolate a non-orthogonal iris in a digital image. The algorithmic approach was to first isolate the pupil, the dark portion in the center of the eye. The pupil was isolated using bit-plane processing. The pupil appeared as a large homogenous region surrounded by insignificant noise, which allowed for easy definition of the pupil-iris boundary. Next, the limbic boundary (the outer edge of the iris) was determined in the cardinal directions, and an ellipse was calculated that incorporated those points. After all boundaries were calculated, an "iris mask" was created to identify pixels in the image that contained the iris data, the only pixels of value for the identification of an individual. The functionality of the algorithm was tested using a database collected at the United States Naval Academy. Both orthogonal and non-orthogonal iris images were used to collect quantitative results.

14. SUBJECT TERMS: non-orthogonal, iris, segmentation, biometric, bit-plane, identification 2	15. NUMBER OF PAGES 121
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Abstract

The goal of this Trident Scholar project was to isolate the iris, the colored part of the eye, in a non-orthogonal, digital image of the human eye. A non-orthogonal image is an image where the eye is not looking directly at the camera. Iris pattern differs significantly between individuals (including identical twins), which allows for its use as an accurate biometric identifier.

Both commercial and research iris recognition systems are becoming widespread in government and industry for logical security and access control. These iris recognition systems assume that captured iris images are normal, or orthogonal, to the sensing devices, and therefore search for circular patterns in the image. Off-angle, or non-orthogonal, images of irises cannot currently be used for identification because the iris appears elliptical; commercial algorithms cannot isolate an elliptical iris in order to start the identification process. This research expanded the functionality of iris recognition technology by developing a set of new algorithms to isolate a non-orthogonal iris in a digital image.

The algorithmic approach was to first isolate the pupil, the dark portion in the center of the eye. The pupil was isolated using bit-plane processing. The pupil appeared as a large homogenous region surrounded by insignificant noise, which allowed for easy definition of the pupil-iris boundary. Next, the limbic boundary (the outer edge of the iris) was determined in the cardinal directions, and an ellipse was calculated that incorporated those points. After all boundaries were calculated, an “iris mask” was created to identify pixels in the image that contained the iris data, the only pixels of value for the identification of an individual.

The functionality of the algorithm was tested using a database collected at the United States Naval Academy. Both orthogonal and non-orthogonal iris images were used to collect quantitative results.

Keywords: non-orthogonal, iris, segmentation, biometric, bit-plane, identification

Acknowledgements

The following individuals have been incredible resources for guidance and support, and have given freely of their time and talents. Thank you.

Dr. Delores Etter, Electrical Engineering Department, USNA – Primary Project Advisor

Dr. Robert Ives, Electrical Engineering Department, USNA – Secondary Project Advisor

Dr. Yingzi Du, Electrical Engineering Department, USNA – Research Asst. Professor

Mr. Michael Wilson, Electrical Engineering Department, USNA – Laboratory Technician

Mr. Jeffery Dunn, National Security Agency – Chief, R3B

Dr. David Murley, National Security Agency – Lead Scientist, R3B

Dr. Michael King, National Security Agency – Technical Director, R3B

Mr. Robert Kirchner, National Security Agency – General Engineer, R3B

Ms. Janice Atwood, Booz Allen Hamilton– National Security Agency Liaison

Mr. David Smith, Science Application International Corporation – Laboratory Technician

Dr. James Matey, Sarnoff Corporation

Trident Scholar Committee Members

Table of Contents

List of Figures

Definition of Terms

Binary Image: A binary image is an image that consists of only two colors – black and white. An integer value of zero is represented by a black pixel and an integer value of one is represented by a white pixel (see Fig. 1).

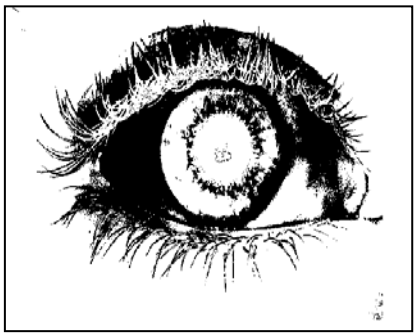


Figure 1: Binary image.

Binary Threshold: A threshold is a pivot value set for grayscale images from which a new binary image is formed. In a 256 level grayscale image, an image value of 0 is black and an image value of 255 is white. All other intensity shades fall as integer values between 0 and 255. Image pixel values of equal or greater value than the threshold are set to *one*, and image values less than the threshold are set to *zero*. The resulting image is a binary, two-tone representation of the original image. Figure 2 shows a grayscale image and a resulting binary image with a threshold value of 128.

Figure 2: (a) Original cameraman image in 256 grayscale. (b) Image after threshold at a pixel value of 128.

Bit-Plane: A bit-plane is a matrix of binary numbers (ones and zeros only) that is formed by removing one bit from the same position of every pixel value in an image. Every color in an image is represented by an integer number. For grayscale images, the values (or shades of gray intensity) range between 0 and 255, and each integer can be represented with eight bits. Figure 3 illustrates the formation of a bit-plane by removing the last bit in each of the nine binary numbers represented.

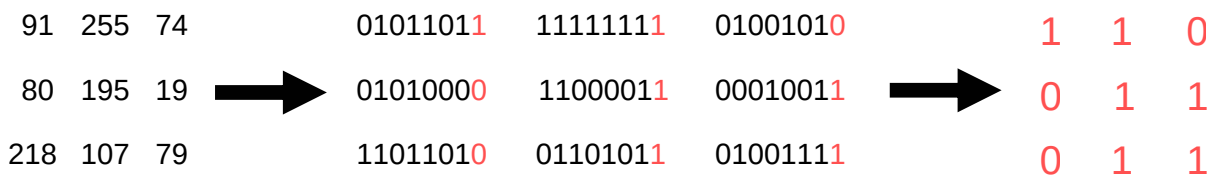


Figure 3: Generating a bit-plane: Original decimal number matrix (left). 3x3 matrix of 8-bit numbers (grayscale image) (center). Resulting 3x3 matrix of last bit only (binary image) (right).

Cornea: The cornea is the transparent part of the coat of the eyeball that covers the iris and pupil, and admits light to the interior.

False Acceptance Rate: False acceptance rate (FAR) is the percent of false matches. In physical security applications, it is the percentage of unauthorized users that are incorrectly identified as authorized users, and are granted access to a secure facility. This is mathematically computed as:

$$FAR(\%) = \frac{\text{Number of false matches}}{\text{Number of attempts}}. \quad (1)$$

False Rejection Rate: False rejection rate (FRR) is the percent of false non-matches. In physical security applications, it is the percentage of authorized users that are incorrectly identified as unauthorized users, and are consequently denied access to a secure facility. This is mathematically computed as:

$$FRR(\%) = \frac{\text{Number of false rejections}}{\text{Number of attempts}}. \quad (2)$$

Hamming Distance: In comparing two bit patterns, the Hamming distance is the percentage of bits that are different in the two patterns. For commercial iris recognition algorithms, a Hamming distance of less than 0.32 constitutes a positive match. More generally, if two ordered lists of items are compared, the Hamming distance is the number of items that do not identically agree [2].

Iris: The iris is the opaque, colored portion of the eye that surrounds the pupil. Under near infrared light, it appears as the annular gray region outside the dark pupil (see Fig. 4).

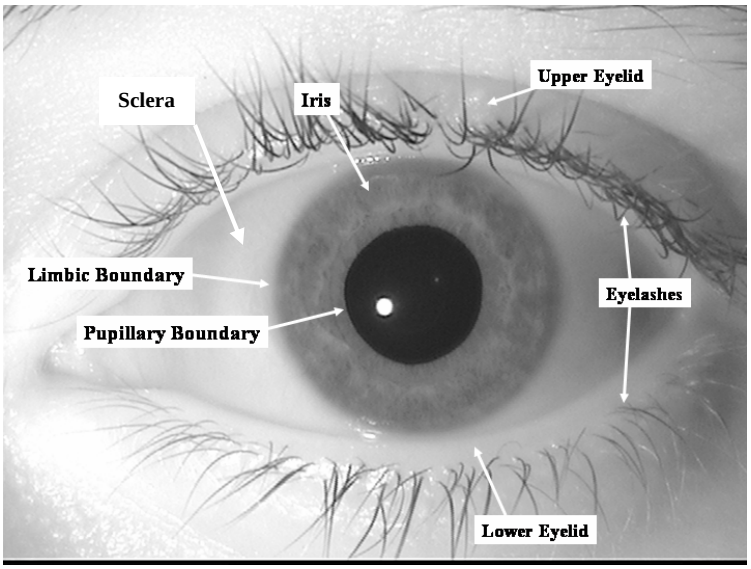


Figure 4: Sample near-infrared iris image.

Iris Code: An iris code is a binary matrix, representative of an individual's iris pattern. An iris code stored in a database is considered the template to which future iris codes are tested in order to determine similarity. Figure 5 is a sample binary iris code.

Figure 5: Sample binary iris code.

Limbic Boundary: The limbic boundary is the boundary distinguishing the separation between the iris and the sclera. This is the outer boundary of the iris (see Fig. 4).

Pupil: The pupil is the homogenous aperture in the center of the iris.

Pupillary Boundary: The pupillary boundary is the boundary distinguishing the separation between the pupil and the iris. This is the inner boundary of the iris (see Fig. 4).

Sclera: The sclera is the opaque white outer coating enclosing the eyeball, except the part covered by the cornea (see Fig. 4).

Standard Deviation Window: A standard deviation window is a neighborhood about each pixel in an image in which the standard deviation of all values within the neighborhood is calculated and saved in the window's origin. The standard deviation is the best measure of dispersion around the arithmetic mean. The standard deviation for a set of numbers is given by:

$$SD = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n-1}}, \tag{3}$$

where x_i is the i^{th} data value, μ is the arithmetic mean of the data points, and n is the number of data points. For example, a 3x3 neighborhood is a matrix with 9 data points.

1. Introduction

Biometric recognition refers to the automatic authentication of a person based on his or her physiological or behavioral characteristics [3]. This emerging field uses unique and measurable physical, biological, or behavioral characteristics that can be processed electronically to establish identification, and to perform identity verification or automated recognition of a person [4]. While traditional means of authentication, primarily passwords and personal identification numbers (PINs), have recently dominated computing, stronger authentication technologies capable of providing higher degrees of certainty are becoming commonplace [5]. Identification, positively matching an individual to a member of a database, and verification, making sure an individual is who he or she claims to be, are the two main uses and areas of expansion in the field of biometrics.

Identifying or verifying (henceforth to be referred to as identifying) a subject through the use of biometrics takes a biological feature of an individual and matches it against templates previously stored in a database. Traditional features include iris (see Fig. 6), retina, fingerprint,

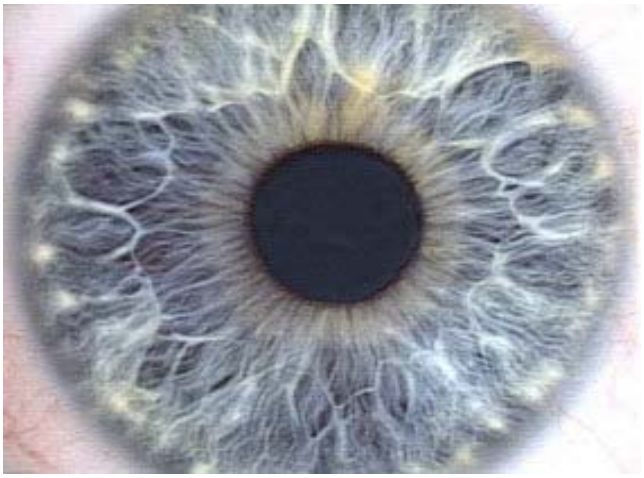


Figure 6. A human eye.
http://www.cl.cam.ac.uk/users/jgd1000/sampleiris.jpg

voice, and face images. Over the last ten years, algorithms used to digitize and process biometric signals have been enhanced to increase both accuracy over repeated uses and precision in

matching users to their respective database entries. Of the five biometric features cited, iris images and fingerprint patterns are currently the most reliable and trusted forms of biometric identification.

The iris is the round, pigmented tissue that lies behind the cornea [6]. The patterns within the iris are very unique to each person, and even the left eye is unique in comparison to the right eye [7]. Compared with other biometric features such as face and fingerprint, iris patterns are more stable and reliable [8, 9].

Since ophthalmologists Flom and Safir first noted the uniqueness of the iris patterns in 1987 [10], various algorithms have been proposed for iris recognition [6, 11-17], which include the quadrature 2D Gabor wavelet method [6], the Laplacian parameter approach [13], zero-crossings of the one-dimensional (1D) wavelet [14], the independent component analysis (ICA) approach [15], Gabor filtering and wavelet transform [16], and the texture analysis using multi-channel Gabor filtering and the wavelet transform [17]. Recently, Du *et al.* designed a local texture analysis algorithm to calculate the local variances of iris images and generate a one-dimensional iris signature [11, 12], which relaxed the requirement of an entire iris for identification and recognition [12]. However, all of these algorithms assume that a circular iris pattern has been successfully extracted from an image of the eye.

In practice, the iris pattern must be extracted from the image prior to analysis. Currently, iris recognition systems require a cooperative subject [8]. Both commercial systems that utilize Daugman's algorithm [18] and other separately developed iris recognition techniques like the one-dimensional approach developed by Du *et al.* [11, 12] rely on this supposition to detect the iris pattern using circular edge detection. As an iris image is rotated away from the normal (ninety degrees perpendicular, see Fig. 7) with respect to the imaging device, these systems are unable to successfully locate the iris pattern in order to proceed to recognition and matching.

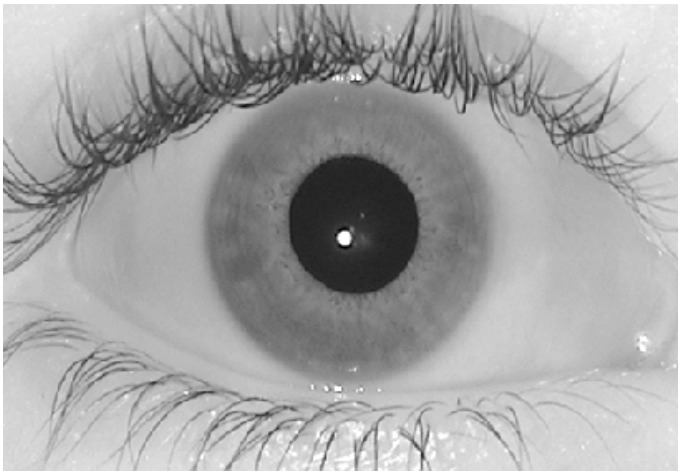


Figure 7: Iris image normal to imaging device.

This project utilized MATLAB to develop algorithms that would analyze a digitized image of a subject's iris in order to extract the iris pattern using elliptical boundaries. Currently, an individual needs to approach an iris scanner, look directly into the imaging device (a digital camera), allow the device to capture an image, and then wait for identification. This process currently takes approximately three to five seconds. By requiring a cooperative user, the imaging device ensures that the iris image will be circular since the user is always forced to look directly into the imaging device. The ability to assume a circular iris pattern simplifies the iris localization algorithms, allowing the algorithms to use circular edge detection. The newly written algorithms no longer require a circular iris for the successful extraction of the iris pattern. Instead, an iris pattern is removed from the image even if the iris is imaged at an angle that is not normal to the sensor. A varying range of off-angle iris images were evaluated to demonstrate algorithmic applicability over a diverse data set.

One major application for such an algorithm is covert surveillance [3]. The use of biometrics for covert identification as opposed to more intrusive forms of authentication poses many technical challenges such as inconsistent viewing angles, varying distances from the detector, and subjects that do not remain stationary [3]. As the circular iris is rotated away from

the normal when captured by an imaging device as seen in Fig. 8, the iris becomes elliptical and the edge detection algorithms employed by current commercial systems reject the images for iris extraction. With advanced algorithms able to process elliptical iris patterns, steps toward

Figure 8: Iris image non-orthogonal to imaging device.

creating a covert system for personnel identification begin. By eliminating the need for a cooperative user to interact with an imaging device, the identification of unsuspecting individuals becomes plausible. Results from this research show that it is possible to fit an elliptical approximation to both orthogonal and non-orthogonal iris patterns using a single set of algorithms. The approximation is used to extract iris pattern data from digital images, which can potentially be adapted for implementation in a non-orthogonal iris recognition system.

2. Background

Every iris is unique and thus can distinguish one individual from another. Iris patterns, unlike facial features used for facial recognition, do not change over time. The iris and its uniquely individual patterns are formed before birth and remain stable throughout an individual's lifetime [5]. The patterns of an iris, once captured by a camera, can be analyzed, and the resulting features can be used to quantitatively and positively distinguish one eye from another. The iris contains many collagenous fibers, contraction furrows, coronas, crypts, colors, serpentine vasculatures, striations, freckles, rifts, and pits as seen in Fig. 9. Measuring the patterns of these features and their spatial relationships to each other provides quantifiable parameters useful to the identification process [19].

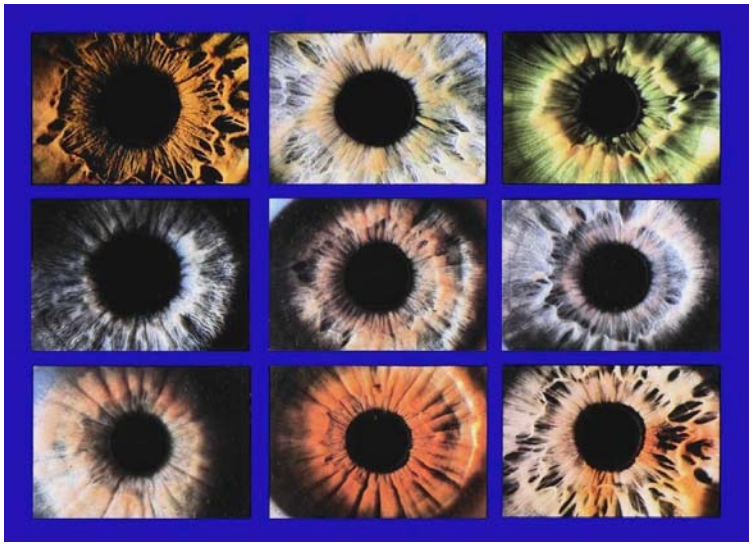


Figure 9: Near infrared iris images with artificial color depicting surface textures.
http://www.cl.cam.ac.uk/users/jgd1000/iriscollage.jpg

Each iris differs due to embryonic genetic development. From the time a person is developing at the cellular level, patterns unique to that individual form in the iris by the random tearing of iris tissue, ensuring that even identical twins are distinguishable. Iris patterns even differ between eyes – an individual is identifiable by either his or her right or left eye [5]. Thus the uniqueness of every iris makes it an incredibly accurate biometric identifier. Based on

mathematical analysis of iris code comparisons performed at the Computer Laboratory at Cambridge University using the current commercial algorithm, the odds that two different irises generate sufficiently similar codes to produce a false match is theoretically 1 in 1.2 million [19]. This theoretical probability is generated based on the number of bits stored in each iris code and the requirement that two iris codes must be identical in at least one-third of the bits in order to return a match. This probability makes iris recognition extremely effective for security applications.

The process for biometric recognition using the iris can be described as a series of steps that make up an architectural framework. This framework for recognition can be divided into five subcomponents, each with its own set of algorithms governing the tasks required of it. Figure 10 shows a flow chart of the architectural framework for iris recognition.

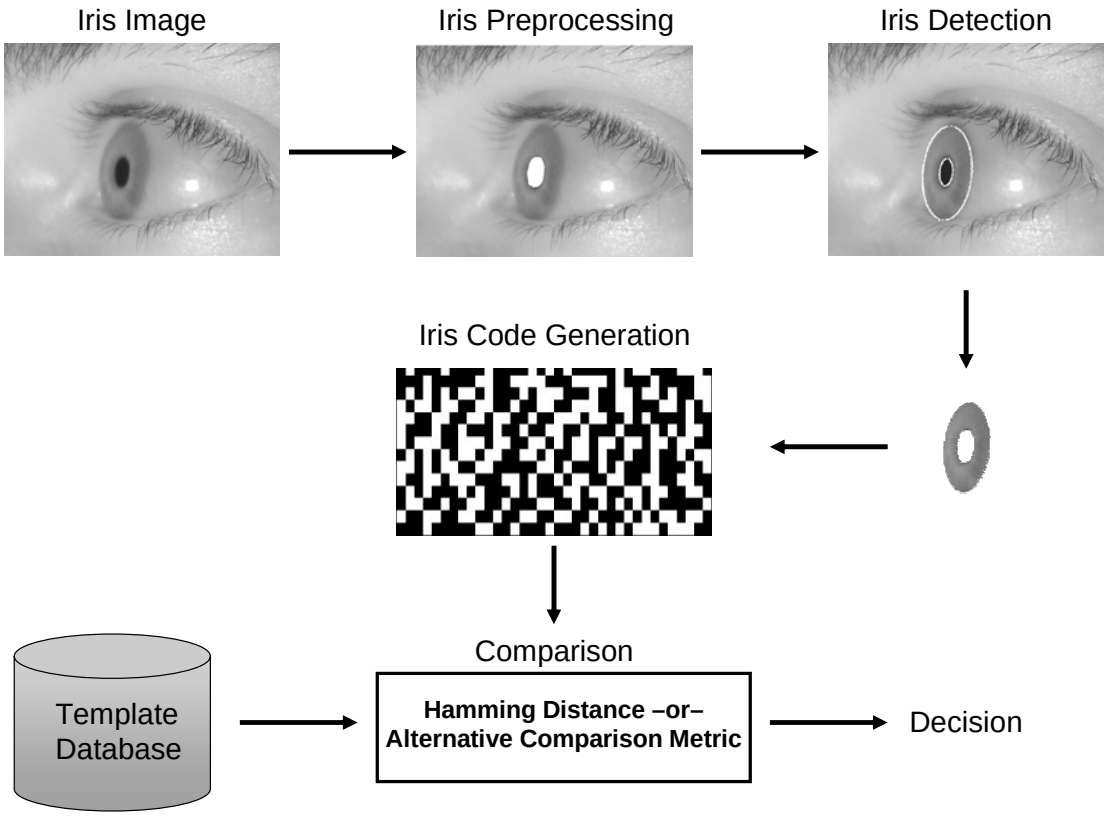


Figure 10: Architectural framework for iris recognition

These subcomponents depicted above are described as:

1) *Image Preprocessing*. In this stage, the first step is to determine the location of the pupil. Then, the iris is characterized using a variety of techniques such as a full unwrapping of the iris to polar coordinates, and making adjustments for illumination, scale and rotation variation. For non-orthogonal iris recognition, there is no transformation to polar coordinates from rectangular coordinates; however, illumination and orientation are still accounted for.
2) *Iris Detection*. This stage involves locating the outer edge of the iris and separating it from the remaining portions of the eye. The data representing the iris itself is called the iris pattern. The iris pattern contains all useful data required for making a positive identification.
3) *Iris Code Generation*. Here, the actual templates (sometimes referred to as the iris code) that will be stored in the database for authorized individuals are created. Additionally, the iris code that will be tested against the database for identification/verification is created.
4) *Comparison*. This stage performs identification/verification by comparing the now-processed code of the presented iris to the iris codes that have been stored in a database. This stage computes the differences between the processed iris code and the stored templates.
5) *Decision*. As the final stage in the iris identification process, this stage makes decisions based on comparisons performed in the preceding stage. It can return up to *N* possible matches above a given threshold that has been set by a system designer. Current commercial systems, however, force a logical output: either an individual is identified or is rejected.

This research focuses on the first two of the five “plug-and-play” steps for biometric identification: image preprocessing and iris detection. The iris detection stage is critical to the successful completion of recognition.

3. Project Description

The main goal of this research was to expand the functionality of iris recognition technology by developing a set of new algorithms to isolate the iris in a non-orthogonal state within a digital image. Coding for the algorithms was completed primarily with MATLAB 7.0. Developed code can be found in Appendix A. The functionality of the algorithm was tested using 129 orthogonal and 236 non-orthogonal iris images from a database collected at the United States Naval Academy. In order to conduct biometric research on human subjects at the U.S. Naval Academy, a proposal was submitted by the biometrics faculty in the Electrical Engineering Department to the Institutional Research Board (IRB) outlining the methods for collection and the ways in which data would be used for research. Before any human subject allows for the capture of his or her biometric data, he or she must complete and sign a copy of the Consent and Information Form found in Appendix B.

3.1. Pupillary Boundary Detection

To find the pupillary boundary, the least significant bit-plane must be extracted for analysis. Bit-plane 0, the least significant bit-plane, is used to determine the pupillary boundary because it not only provides a relatively homogenous region that is easily identifiable as the pupil, but also because it is fast and easy to extract from the original image by using modulo division. In integer division, the result is a quotient and a remainder. However, modulo division is the operation with a result equal only to the remainder. When dividing by two, the remainder can only be either a one or a zero. After performing modulo-2 division on the image, the result forms the least significant bit plane. To aid in extracting the least significant bit-plane, a series of image preprocessing steps much be performed.

The first step towards achieving a homogenous region that can be identified as the pupil (see Fig. 11) is to adjust the original image by setting the pixel values below 60 and above 240 equal to 255. These upper and lower bounds were derived empirically for use with the USNA database

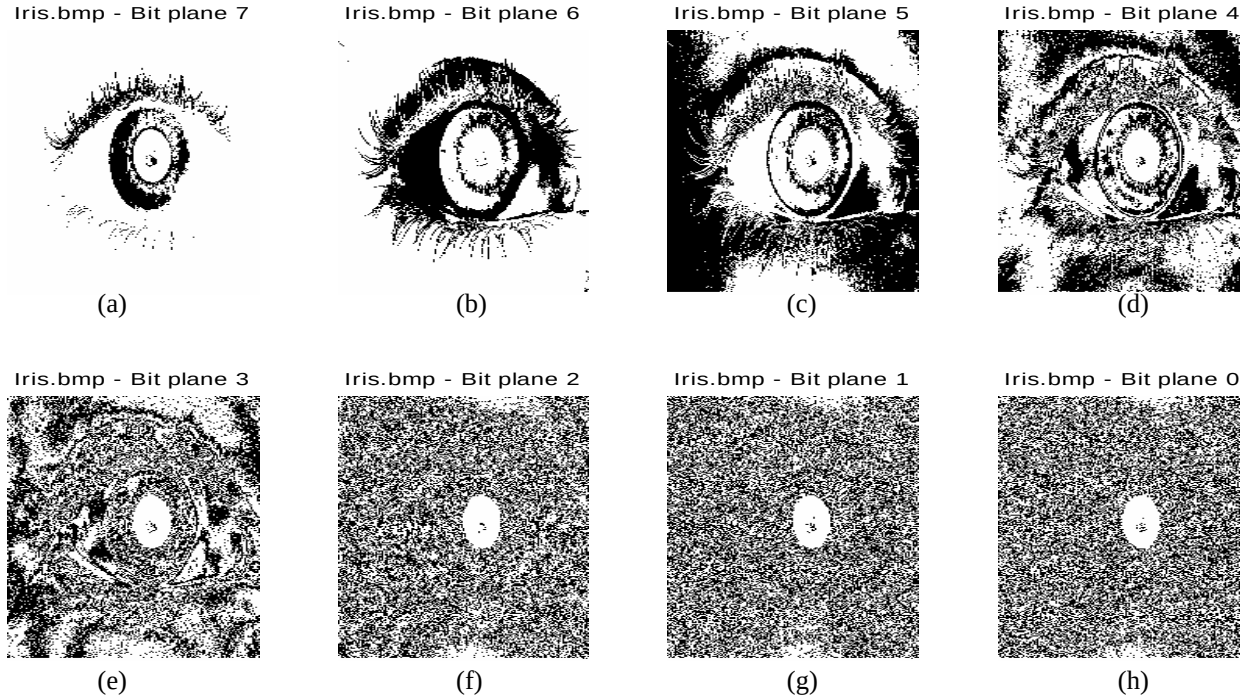


Figure 11: (a) Most significant bit-plane 7. (b) bit-plane 6. (c) bit-plane 5. (d) bit-plane 4. (e) bit-plane 3. (f) bit-plane 2. (g) bit-plane 1. (h) bit-plane 0.

images. The pixel intensity in a grayscale image ranges from 0 to 255, with 0 representing black and 255 representing white. By adjusting the original iris image, all extremely dark and extremely light regions are forced white. Once the bit-planes are extracted from the original image, the pupil is represented by a homogeneous mass of binary ones in all bit-planes as seen in Fig 11. Figure 12(a) shows the adjusted least significant bit-plane after it has been extracted from the original iris image.

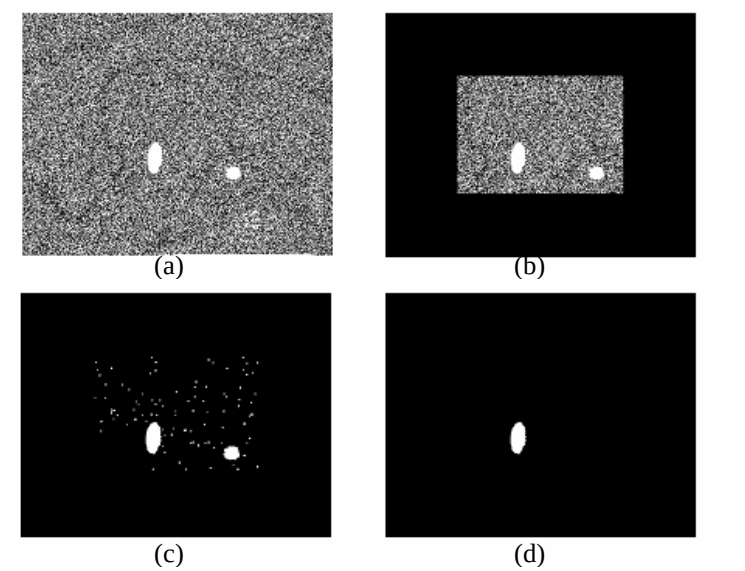


Figure 12: Iris pupil location (a) Least significant bit-plane (bit-plane 0). (b) Bit-plane 0 with borders removed. (c) Bit-plane 0 after morphological "open" performed. (d) Final mask of pupil extracted.

The purpose for adjusting values between 240 and 255 – extremely light values – to white is to reduce the effect of specularities, or glare, that may be present in the pupil. Pixel intensities of the pupil region – extremely dark values – in iris images collected from the LGIris 3000, the imaging device used to collect the Naval Academy database, were found to typically range from 30 to 50 due to the infrared illumination generated by the camera. Therefore, a lower threshold of 60 was used for adjusting the pupil to a single uniform value.

Since the eye is approximately in the center of the images collected, the borders of the bit-plane are removed (Fig. 12(b)) to minimize the effects of near-infrared glare reflecting off the

lighter pigmented skin above and below the iris. Glare is represented by high pixel values which appear as clusters of binary ones in all bit-planes. These larger groupings of ones must be removed prior to pupil extraction. Once the borders have been removed, excessive noise must be eliminated from the binary image.

To remove the random noise, the binary morphological operation "open" is performed. The open function is defined as binary erosion followed immediately by binary dilation. Binary erosion shrinks groupings of ones (white pixels) that are larger than three pixels in diameter, and completely eliminates groupings of ones that occur in pairs or stand alone. Conversely, binary dilation expands groupings of ones. In order to maintain the original size and shape of the pupil, binary dilation must follow binary erosion. The result, as shown in Fig. 12(c), is the removal of a substantial portion of the image's noise. Despite the small amounts of remaining clustered noise, the pupil can now be extracted by isolating the largest mass of binary ones that remains in the bit-plane (Fig. 12(d)).

To aid in properly extracting the pupil from the image, the built-in MATLAB function *regionprops(…)* is utilized to provide image statistics. In binary images, the function returns information including the area (number of pixels), lengths of the major and minor axes, orientation, and centroid (the center pixel location) for every cluster of ones. By quickly searching the list of areas generated by the function, the pupil statistics are readily available since the pupil is the only large remaining group of ones. Once the pupil and its associated statistics are obtained, the boundary points in the cardinal directions – N, E, S, and W – are determined. Because the center location of the pupil is known, these boundary points are easy to find. By starting at the center and checking outward in a given direction, the end points of the pupil are quickly located and used to calculate the elliptical curve that will ultimately define the pupillary boundary. Equation (4) is used to determine the elliptical curve,

$$\frac{x^2}{b^2}+\frac{y^2}{a^2}=1, \tag{4}$$

where b and a are the lengths of major and minor axes, and x and y are the row and column coordinates of the ellipse. Equation (4) can be rewritten as

$$y=\pm\sqrt{a^2\left(1-\frac{x^2}{b^2}\right)}, \tag{5}$$

and the elliptical curve calculated from these points forms the pupillary boundary as seen in Fig. 13. The orientation, or tilt, of the pupil is accounted for by rotating the ellipse in accordance

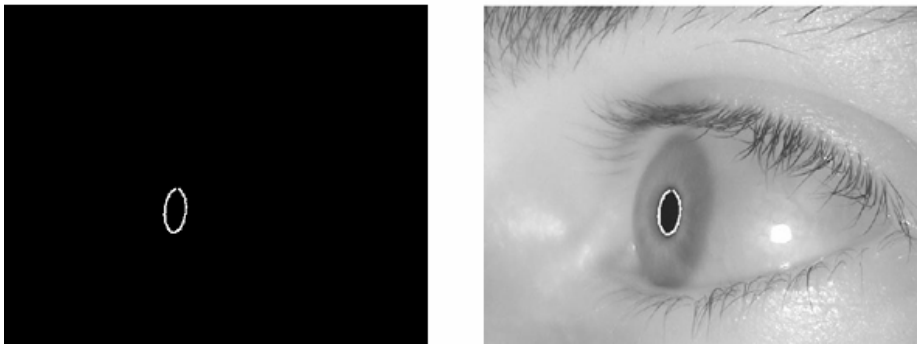


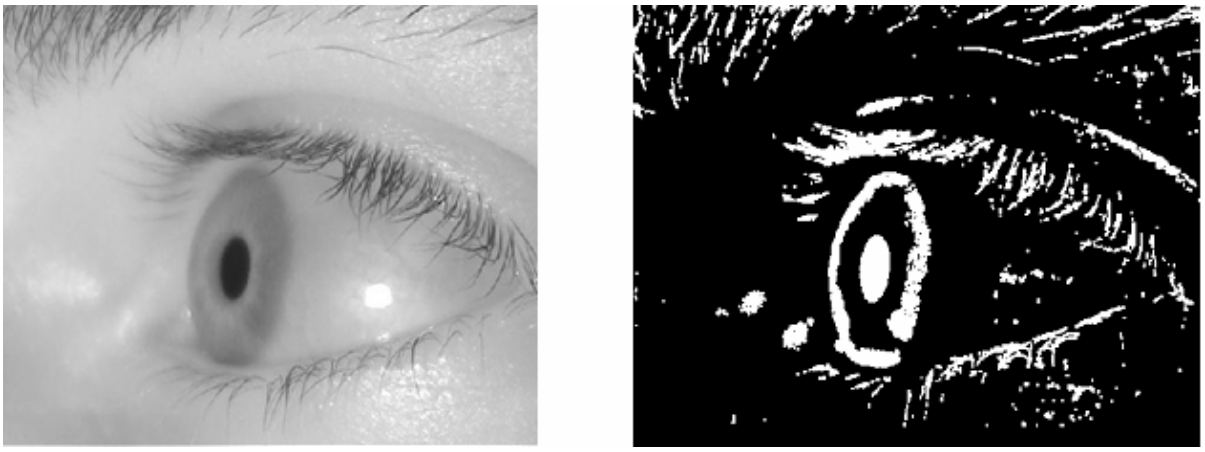
Figure 13: Elliptical curve fit through cardinal points of pupil mask (left). Pupillary boundaries overlain on original iris image (right).

with the orientation factor previously determined using the MATLAB function *regionprops(…)*. Since the pupil is always fully contained within the iris, these cardinal boundary locations provide the starting point in identifying the limbic boundary, the next step in iris pattern segmentation.

3.2. Limbic Boundary Detection

Once the pupillary boundary has been successfully determined, the limbic boundary must be isolated. To find the limbic boundary (the division between the iris and the sclera) a local standard deviation window is used to create a new image that represents local changes in image grayscale intensity. For image locations where the difference in grayscale values from one pixel to the next is large, the new image is represented by light colors. On the other hand, if there is minimal grayscale value variation in a region of the original image, the newly created image is dark in the corresponding areas. A 45x45 standard deviation window is applied throughout the original iris image. A new image is formed in which each pixel contains the local standard deviation from the 45x45 standard deviation window. The new image is converted into a binary image based on a dynamic threshold. The dynamic threshold is computed by taking the local mean of each window. The local mean for a window is the average value of each pixel contained in the window. Figure 14 shows an original iris image and the resulting binary image based on the dynamic threshold after a 45x45 standard deviation window has been applied.

In order to isolate the iris pattern from the binary image in Fig 14(b), the scope of



(a) (b)

Figure 14: (a) Original Iris Image. (b) Resulting binary image after a square standard deviation window of side length forty-five and a dynamic local threshold have been applied.

applicable search regions must be limited. By limiting the detection directions to vertical and horizontal bands, the cardinal locations from which to derive an elliptical boundary can be isolated. To create the vertical and horizontal bands needed for iris segmentation, the pupil's centroid (previously determined in the pupillary boundary location phase) is used as a reference. While not always located exactly in the center of the iris, the pupil's location is such that it is reasonable to estimate that the pupil's centroid is equivalent to the iris's centroid. Therefore, a single pixel-wide band is extracted from the binary image in the vertical and horizontal direction as shown in Fig. 15. These bands effectively eliminate unimportant regions of the

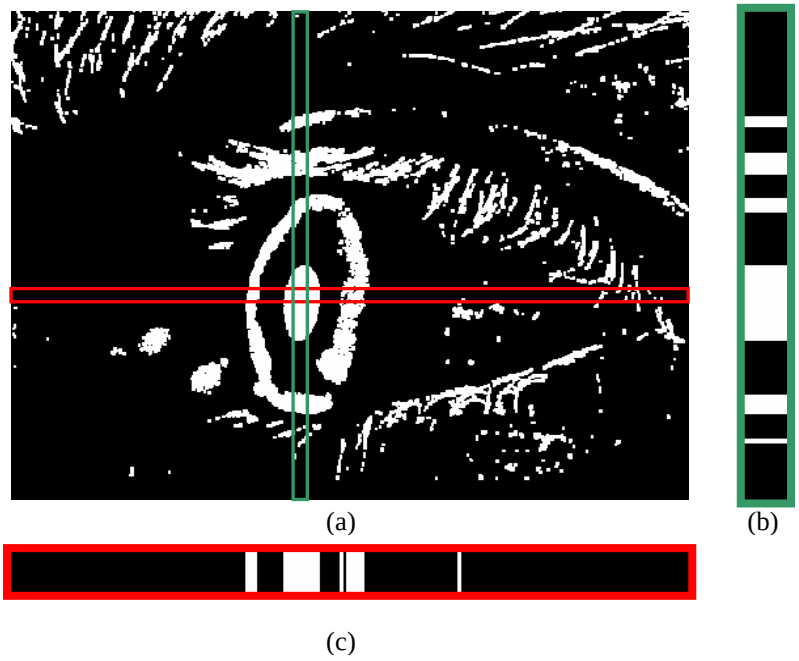


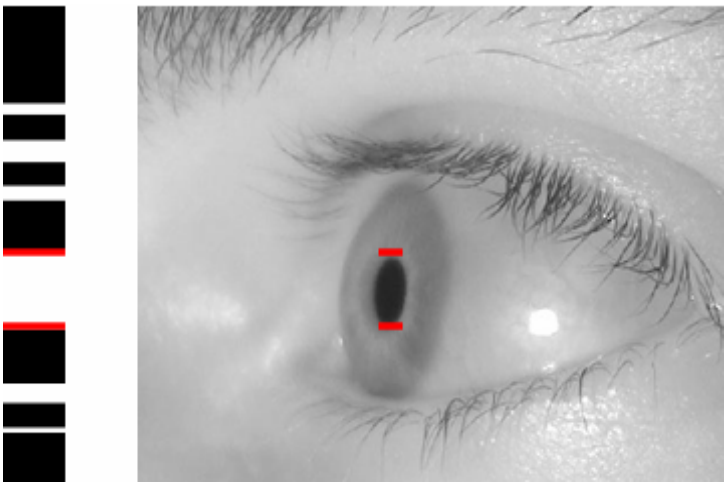
Figure 15: (a) Binary image of eye after local standard deviation window applied. (b) Binary vertical band taken through center of pupil. The band is one pixel wide and 480 pixels high. (c) Binary horizontal band taken through center of pupil. The band is 640 pixels wide and one pixel high.

original image and allow for concentration of limbic boundary detection in the cardinal directions.

In order to determine the limbic boundaries from the binary horizontal and vertical bands, the pupillary boundary coordinates in the cardinal directions must be known. These

coordinates provide a starting location for the search of the limbic boundary because the limbic boundary must be located outside of the pupillary boundary. By starting at the previously determined pupillary boundaries, and searching outward in the cardinal directions, the limbic boundaries can be isolated.

To automatically isolate the limbic boundary using the horizontal and vertical local standard deviation bands, an iterative method is used. In the vertical direction, the binary vertical band is analyzed for potential boundaries. Based on the previously defined local standard deviations and dynamic thresholding, major image intensity changes are displayed as horizontal white lines within the band. Two of the lines represent the pupillary boundaries and two of the lines represent the limbic boundaries in the N and S directions. The first step is to mark the pupillary boundary lines as the initial guess of the limbic boundary locations. Although this cannot be the actual location, it provides an absolute minimum location for the limbic boundaries. Figure 16 shows the initial starting search locations of the limbic boundaries (based on the previously determined pupillary boundaries) in both the binary vertical band, Fig. 16(a), and in an overlay of the original iris image, Fig. 16(b).

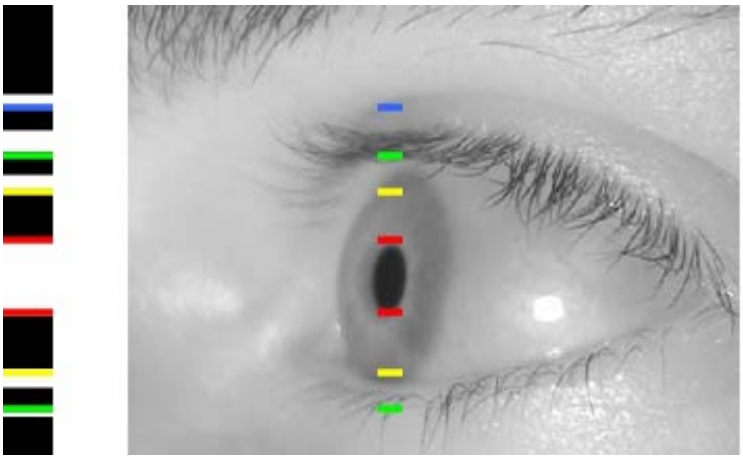


(a) (b)

Figure 16: (a) Initial limbic boundary locations in the vertical direction. (b) Overlay of initial limbic boundaries on original image.

The second step is to search for the next potential limbic boundary. To do so, the next closest white line in the binary band is identified. This new limbic boundary replaces the former marker. A check to ensure the new boundary is a realistic possibility is performed. The check verifies that the new marker is not more that twenty pixels from the previous marker. A distance of twenty pixels was heuristically determined to yield the best results for automatic limbic boundary detection. Once the new marker is determined to be legitimate, step two is repeated until a marker is found to be more than twenty pixels from the previous marker. In the case that the new markers are never more than twenty pixels away from the previous marker, the edge of the image is used as the location of the limbic boundary – a situation that will automatically alert one to a failure to properly segment the iris from the image.

Upon verification that a marker is outside the twenty pixel limit, the previous marker is set as the limbic boundary. Figure 17 shows the iterative process of automatically finding and



(a) (b)

Figure 17: Automated process for determining limbic boundaries. (a) The iterative process for determining limbic boundaries using the binary vertical band: Red to Yellow to Green to Blue (if necessary). (b) An overlay of corresponding locations for limbic boundary potentials.

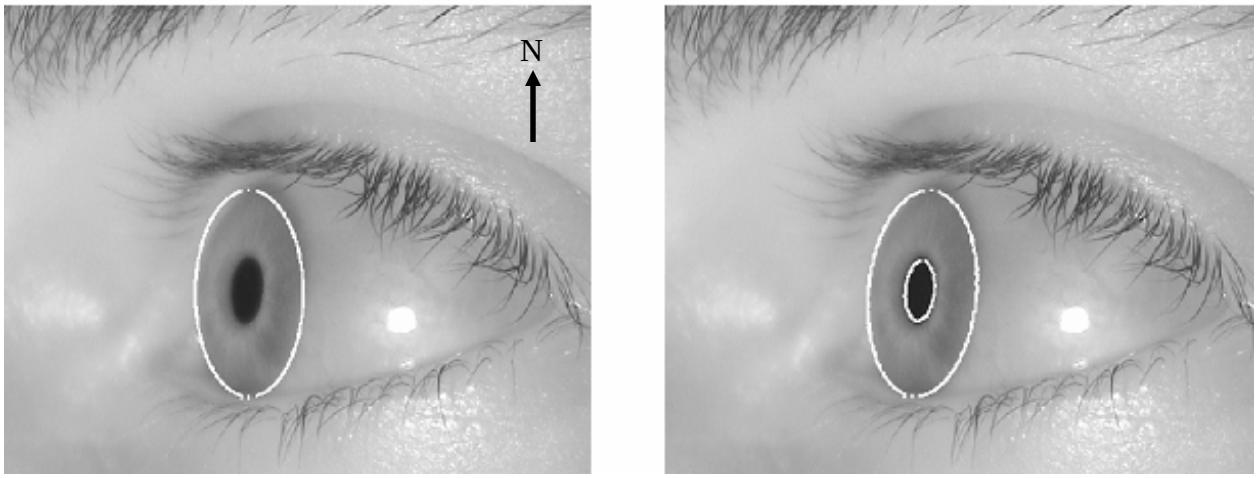
checking the validity of the limbic boundary. Both the progression through the binary bands and their respective locations within the iris image are displayed. Steps one through three are performed in an identical manner for locating and verifying the position of the horizontal limbic

boundaries.

After all four limbic boundaries are determined in the cardinal directions, (5) is used to calculate the elliptical equation that will define the boundary of the iris. The right and left portions of the limbic boundary are calculated and added separately. By adding right and left boundaries, North and South locations are fixed and the East and West positions are varied. This method was purposefully chosen to provide the most accurate iris mask model. As the severity of the iris angle increases, the cornea covering the iris eventually causes one half of the iris to appear more elliptical than the other. By modeling each half (right and left) of the limbic boundary separately, this phenomenon does not affect algorithm performance.

The limbic boundary equations are calculated independently of one another. For example, when forming the right half of the limbic boundary, the N, S, and W portions of the boundary are known. The E coordinate, while available, is not necessarily directly opposite and equidistant from the vertical axis to properly form an ellipse. To find this appropriate coordinate, the location of the W boundary is mirrored across the vertical axis, and this imaginary location is used for an accurate calculation of the limbic boundary. The same process occurs for the formation of the right limbic boundary. As the angle at which the image of the iris is captured increases, the shape of the iris transforms from circular to elliptical. However, at extreme angles, optical distortion effects of the cornea cause the closer portion of the iris to appear more vertical while the farther portions of the iris are more circular. Each half can be accurately modeled as an ellipse; however, the ellipses are independent of one another and must be calculated separately.

The final step for limbic boundary creation is to account for changes in the vertical tilt of the iris. As seen in Fig. 18, the initial limbic boundary does not necessarily match the orientation



(a) (b)

Figure 18: (a) Initial limbic boundary determination. (b) Boundary after orientation adjustments.

of the iris in the image. Recall during pupil extraction that one of the parameters returned by the *regionprops(…)* function was "orientation". This parameter took into account the major and minor axes of each mass in the binary image and calculated its orientation relative to the Cartesian plane. By rotating the pupillary and limbic boundaries to coincide with the orientation acquired earlier, the final boundaries calculated match the vertical tilt of the iris as seen in Fig. 18(b). The resulting area between the pupillary boundary and the limbic boundary forms the iris mask. The mask allows for proper extraction of the iris pattern. Figure 19 shows an original non-orthogonal iris image and the resulting segmented iris pattern.

(a) (b)

Figure 19: (a) Original non-orthogonal iris image. (b) Segmented iris pattern.

4. Testing and Results

A "Quality Bound" is used as the metric for assessing the performance of the algorithm. It is the ability of the algorithm to properly extract iris pixels that is being measured. The United States Naval Academy biometrics laboratory iris database was used for algorithm testing. There were 129 orthogonal and 236 non-orthogonal iris images used. To test the algorithms' performance, a graphical user interface (GUI) was developed. The GUI allowed for a systematic routine to be followed for testing control. Additionally, a quality measurement was created to quantitatively establish the algorithms' performance. Three quality bounds were calculated using:

$$QualityBound = \frac{TP \cap MP - K \cdot \left(MP - TP \cap MP\right)}{TP}, \tag{6}$$

with the variables defined as:

TP	True Pixels	The number of pixels in the "truth" mask (acquisition covered in 4.1),
MP	Mask Pixels	The number of pixels in the iris segmented by the algorithm,
K	Penalty	The error penalty factor assessed for incorrectly identified iris pixels.

The number of common pixels (CP) to both the segmented iris and the true mask is represented by the intersection of TP and MP as seen below:

$$CP = TP \cap MP\,. \tag{7}$$

The number of error pixels (EP) is defined as the number of pixels in the segmented iris minus the number of pixels common to both the segmented iris and the true mask.

$$EP = MP - CP\,. \tag{8}$$

By combining (6) – (8), the quality bound can be written simply as:

$$QualityBound = \frac{CP - K \cdot EP}{TP}\,. \tag{9}$$

The penalty factor, K, is used to help correctly assess the performance of the segmentation algorithms. Factors of 10, 40, and 70 percent are used to scale the overall error penalty (8) in calculating the quality bound. A factor of 10 is a lesser penalty than a factor of 70. The error penalty is based on the number of pixels that differ between the "test" and the "truth" iris masks. The application of the penalty factor to the error penalty gives an understanding of the algorithm's performance. By evaluating the three separate quality bounds after the 10, 40, and 70 percent penalties have been assessed, a quality spread (or variation) can be determined. The variation is directly related to the number of superfluous pixels segmented as "truth" pixels (eyelashes, eyelids, etc.). For example, if all three quality bounds are close in proximity, the iris pattern that was segmented automatically by the algorithm has relatively few pixels that are not true iris pattern pixels. However, as the difference in quality bounds increases, the iris pattern that was segmented by the algorithm contains eyelashes and additional extraneous information. If all automatically generated "test" mask pixels fall within the "truth" iris mask, the error penalty is 0. If the "truth" and "test" iris masks match perfectly, all three quality bounds will be 1.0.

4.1. Graphical User Interface Design and Implementation

A graphical user interface (GUI) was designed for aid in testing the developed algorithms. The GUI required the filenames of a "truth image" and an associated "iris image" as shown in Fig. 20. The truth images were created prior to algorithm testing by using Adobe®

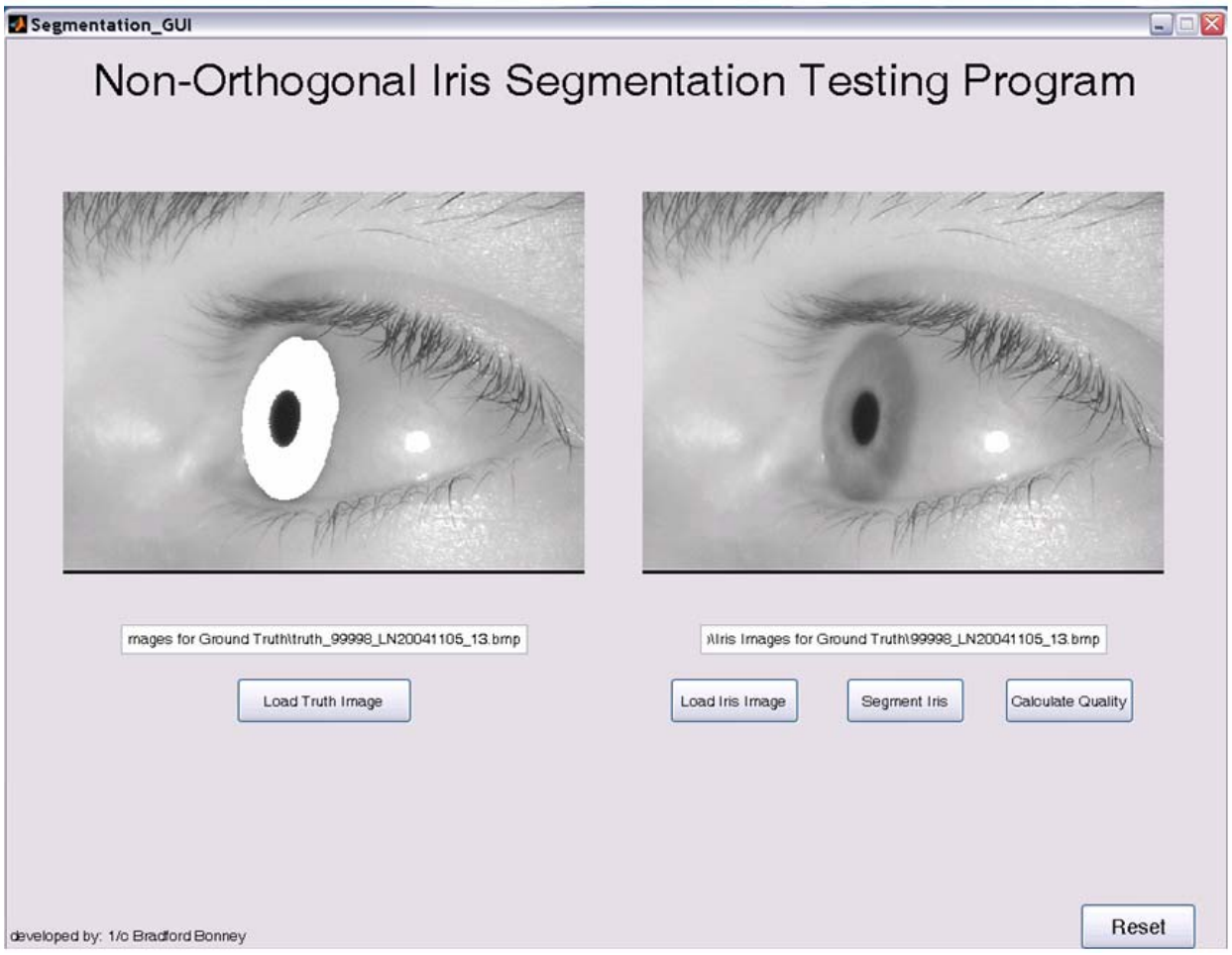


Figure 20: "Truth" and initial "Iris Image" files loaded into GUI by user.

Photoshop CS. The iris's boundaries were isolated using the *magnetic rope* tool – an imbedded feature of Photoshop CS. Once the boundaries were isolated, the internal area of the boundaries was filled with white. The white, value 255, allowed the GUI to quickly identify the iris pattern and extract the proper pixels associated with the pattern. This provided a baseline for comparison to the automated process.

The original image was also a required input for the GUI to function properly. Once loaded, the original image underwent the iris segmentation steps described in Sections 3.1 and 3.2 (Fig. 21). After the iris was properly segmented, a pixel-by-pixel comparison between the

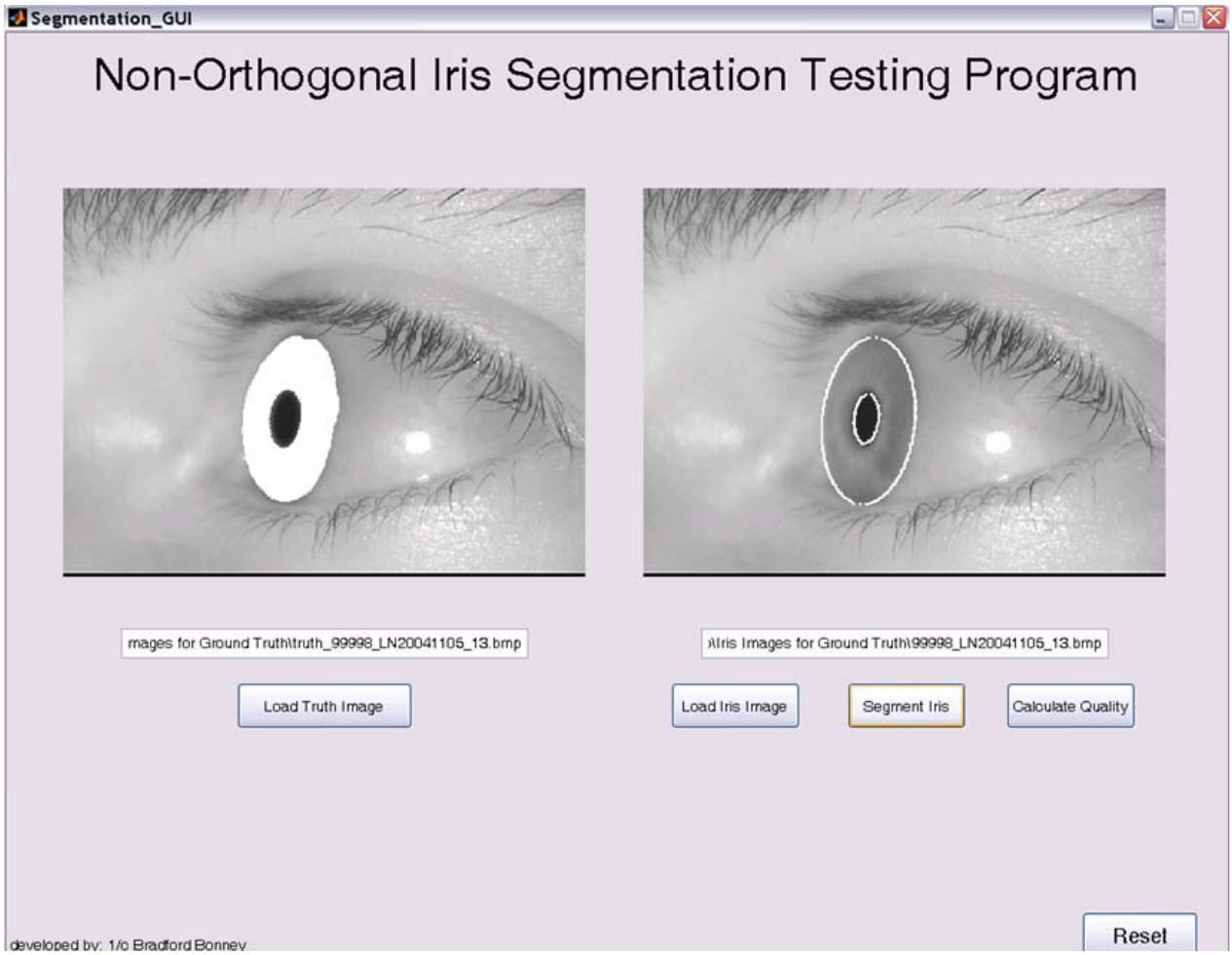


Figure 21: Iris segmentation completed by clicking on "Segment Iris."

original and truth images was conducted. The location of each pixel in the segmented iris pattern was compared to each pixel location of the truth image. The number of common pixels (CP) was recorded and used as a parameter in (9). Similarly, the total number of pixels in the truth image and the total number of pixels in the original test image were also used in (9) for calculation of the overall quality factor. All three quality factors – 10%, 40%, and 70% – were simultaneously displayed as output within the GUI (Fig. 22).

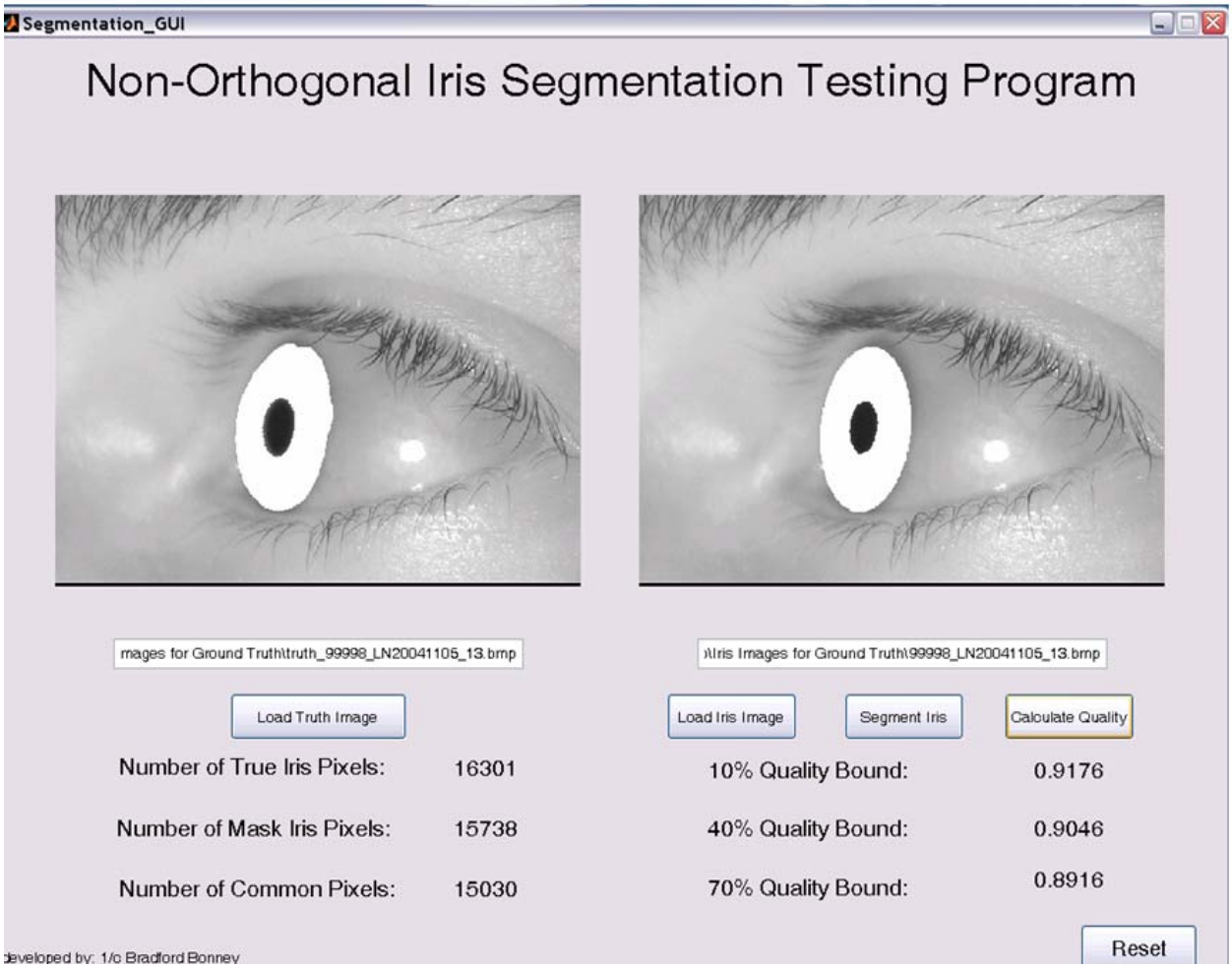


Figure 22: Final calculation of iris segmentation mask quality factors.

4.2 Experimental Data and Analysis

A full listing of experimental data can be found in Tables (1) and (2) in Appendix B. Figure 23 and Fig. 24 compare performance of the algorithms for segmenting the iris pattern.

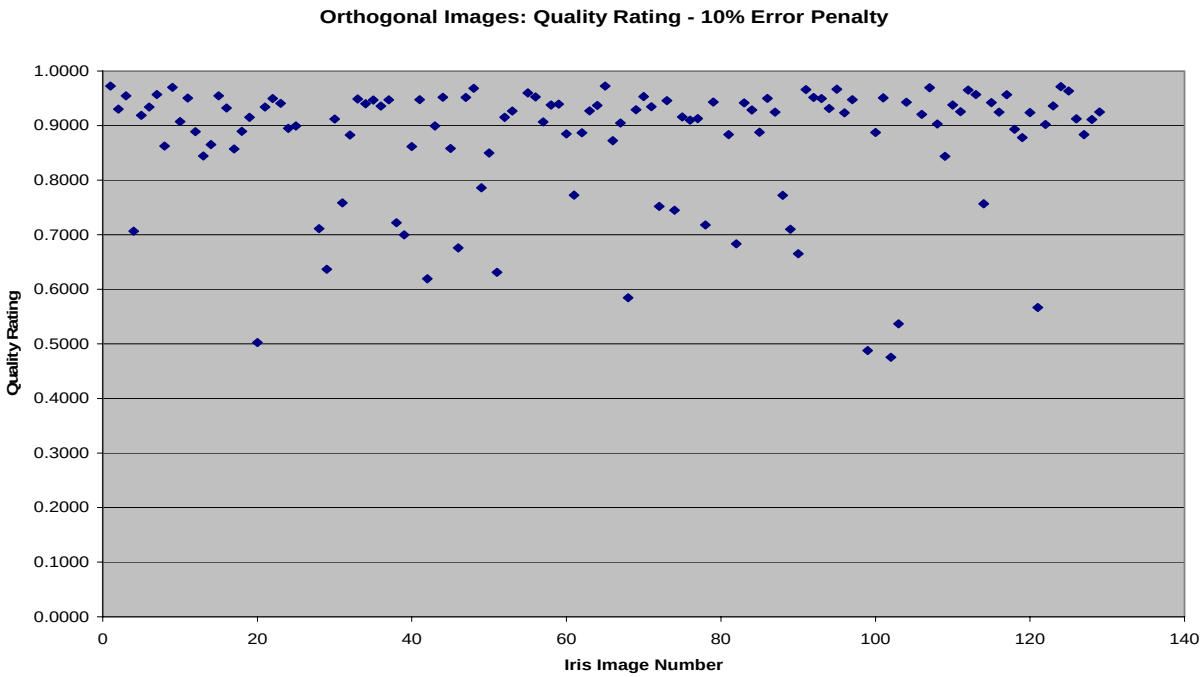


Figure 23: Orthogonal iris segmentation quality with 10% error penalty.

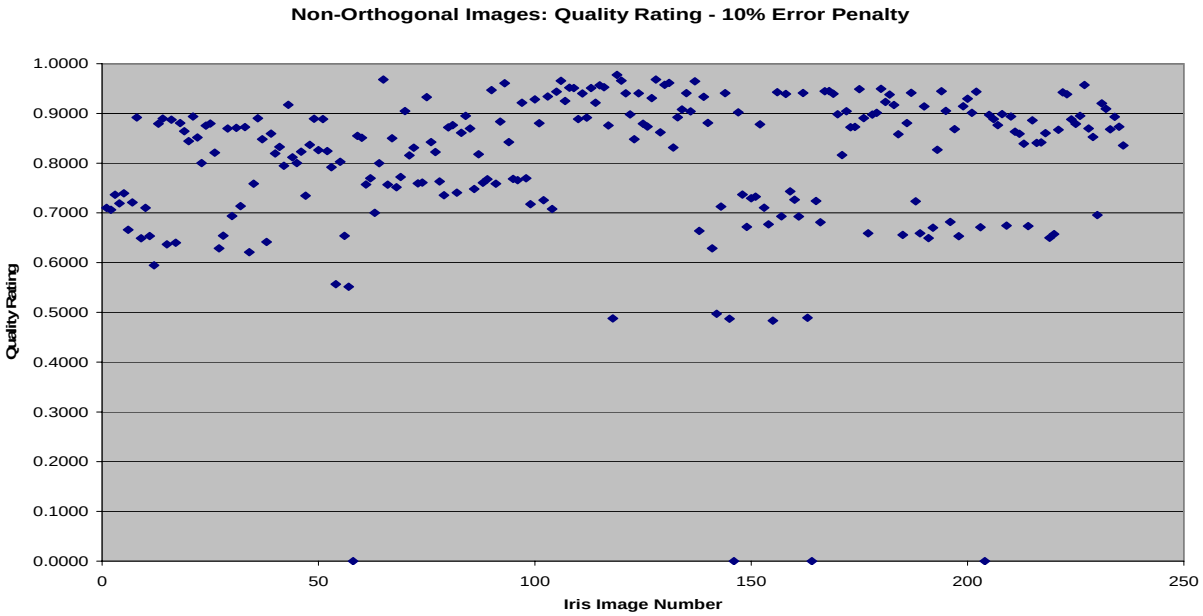


Figure 24: Non-orthogonal iris segmentation quality with 10% error penalty.

The Quality Bound is shown on the y-axis as a value between 0 and 1, and the number of the image segmented is shown on the x-axis. There were 129 orthogonal images and 236 non-orthogonal images that were processed to segment the iris pattern[1].

The 10% error penalty quality bounds cluster primarily above 0.9000 for the orthogonal images and above 0.8000 for the non-orthogonal images. Similarly, the 40% and 70% error penalty quality bounds tend to cluster approximately ten percent lower for non-orthogonal iris images than orthogonal iris images (see Figs. 25-28). This is primarily explained by the increased presence of eyelashes in the non-orthogonal iris masks. As the angle at which an iris is captured increases, the more obstruction from eyelashes and eyelids tend to exist.

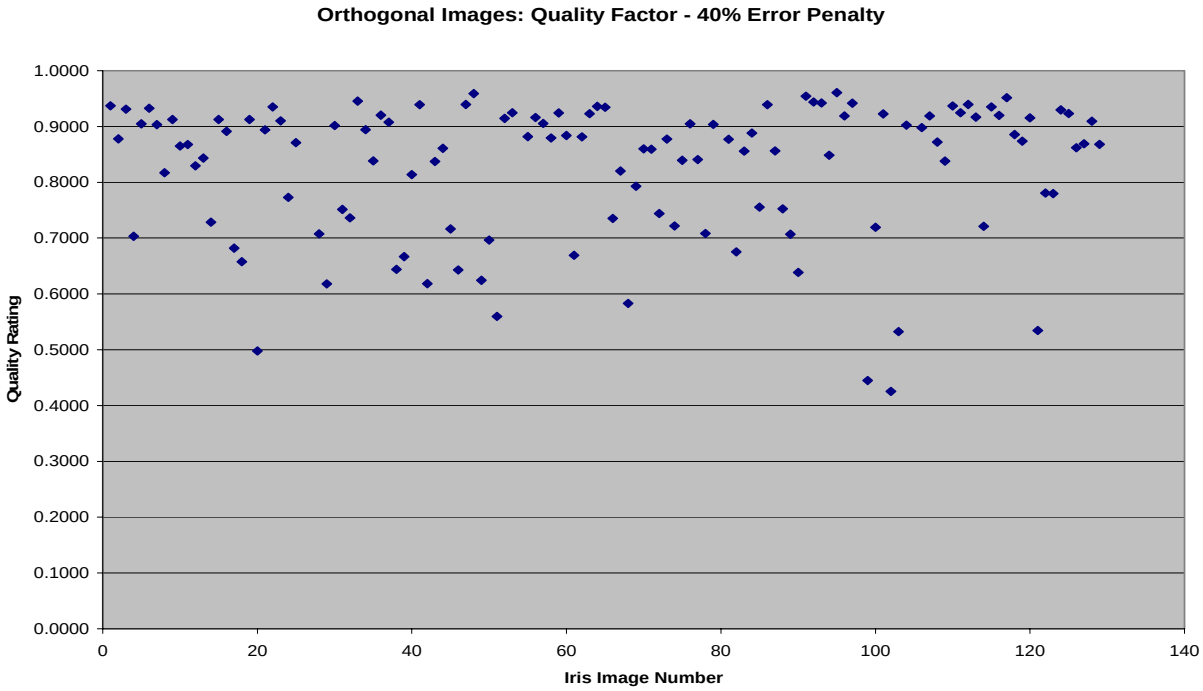


Figure 25: Orthogonal iris segmentation quality with 40% error penalty.

[1] The orthogonal images tested were taken from a database of 129 different eyes. The non-orthogonal images tested were taken from a database of 32 different eyes. 236 images were used for testing by analyzing multiple images of the 32 non-orthogonal eyes.

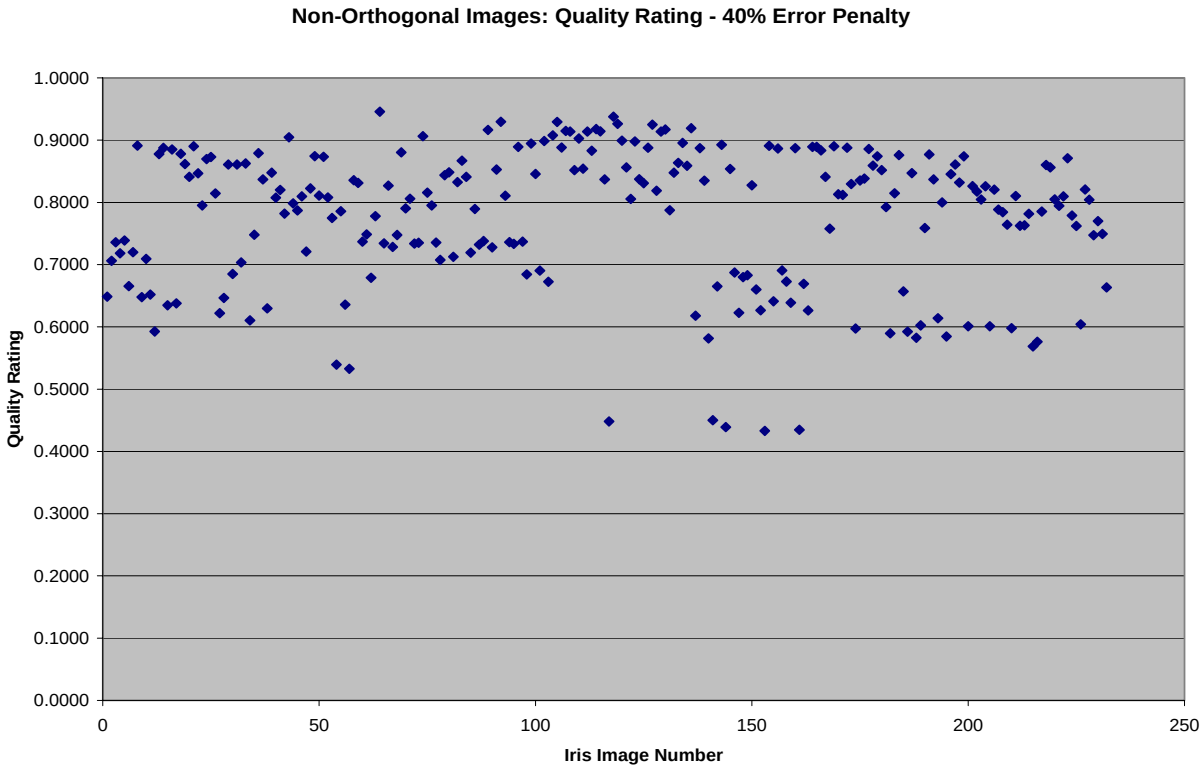


Figure 26: Non-orthogonal iris segmentation quality with 40% error penalty.

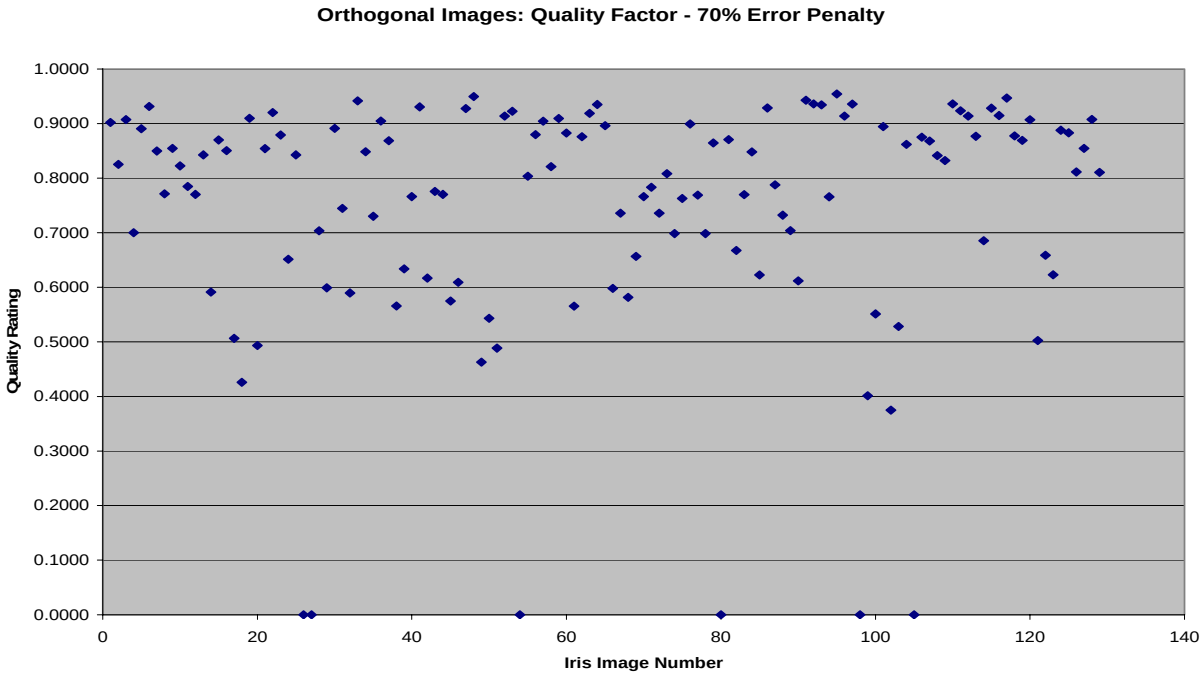


Figure 27: Orthogonal iris segmentation quality with 70% error penalty.

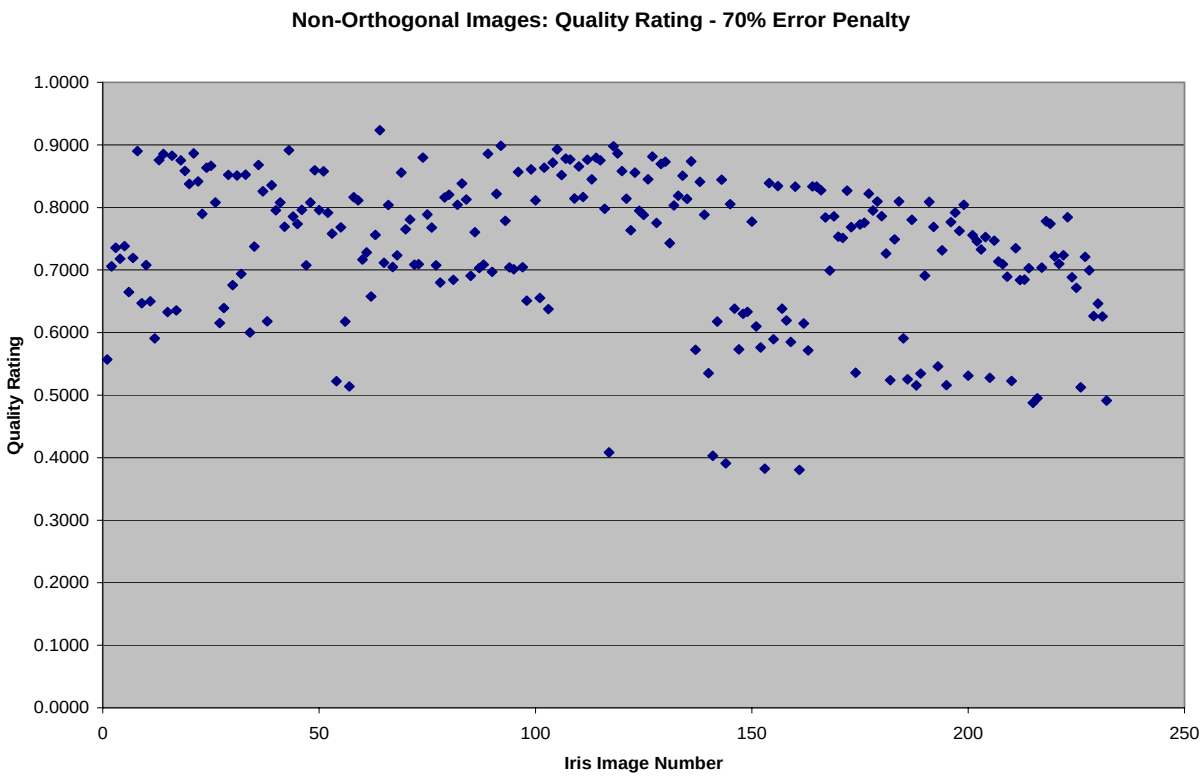
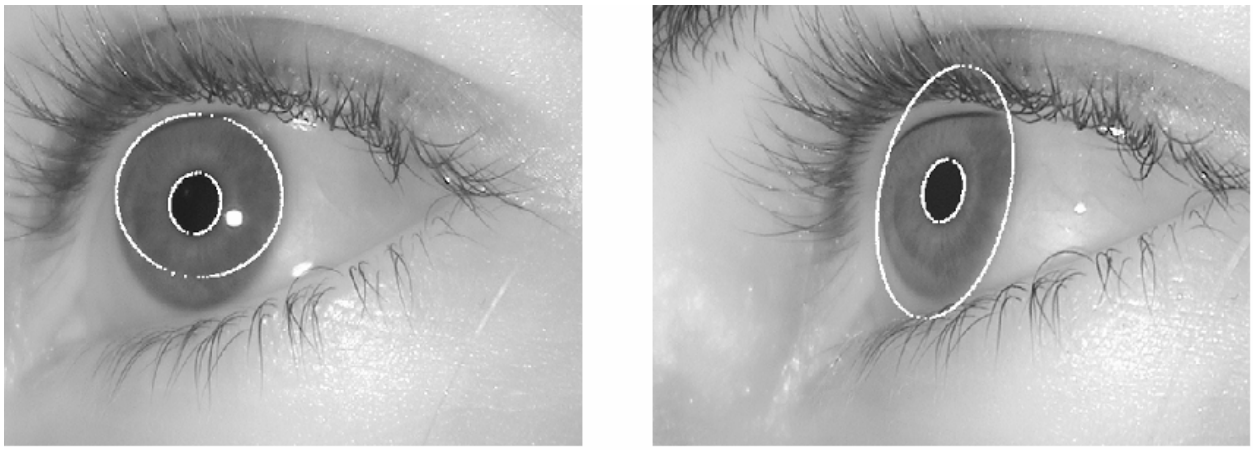


Figure 28: Non-orthogonal iris segmentation quality with 70% error penalty**.**

4.3 Analysis of Quantitative Data

For each iris tested, all three quality bounds were recorded and a variation was calculated. The variation was equal to the average change in quality factors for a given iris. The variation is an indication of how well the segmentation process performed for an iris image. It is directly related to the segmentation discrepancies between the "truth" mask pixels and the automatically segmented "test" mask pixels. The value of the variation relates to the presence of unusable data and, therefore, a lower variation means fewer eyelashes, eyelids, and other impurities in the iris mask. For example, an iris with a lower quality factor and low variation may have a more usable iris mask for identification than an iris with a higher quality factor and a high variation (Fig. 29).



(a) (b)

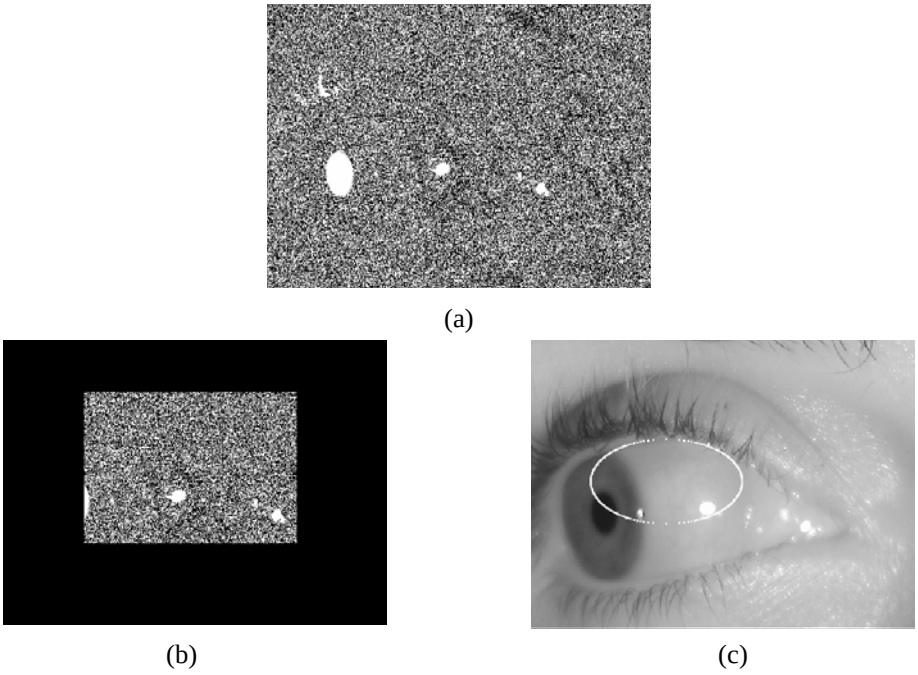
Figure 29: (a) Iris image 3 segmented with 10% quality factor of 0.7364 and variation of 0.0005 (low quality factor, low variation). (b) Iris image 231 segmented with 10% quality factor of 0.9201 and variation of 0.0995 (high quality factor, high variation).

While Fig. 29(b) captures virtually all of the true iris pixels as shown by its 10% quality factor of 0.9201, the iris pattern mask also contains much additional noise (i.e. eyelashes and sclera).

Associated research that explored the effects of using partial iris patterns for identification states that it is possible to identify an individual using the Du, *et al* 1-dimensional (1-D) algorithm with as little as twenty-five percent of an iris pattern (Appendix F). For the 1-D algorithm to yield such results, the small fraction of an iris must come from the interior section

of the iris (closest to the pupil). This area of the iris was shown to contain the most distinguishable data. Most of the results returning low values of quality ratings (ratings below 0.7 at a 10% penalty) can be classified as similar to Fig. 29(a): the iris mask boundaries fall within the true iris.

Of the 129 orthogonal images, 6 images were FS, or *failure to segment*. Four of the 236 non-orthogonal images were FS. An FS image was one in which the algorithms failed to properly segment the iris. The main cause for a failure to acquire was usually due to image quality. For example, in the image in Fig. 30, the iris was not located in the center of the image.

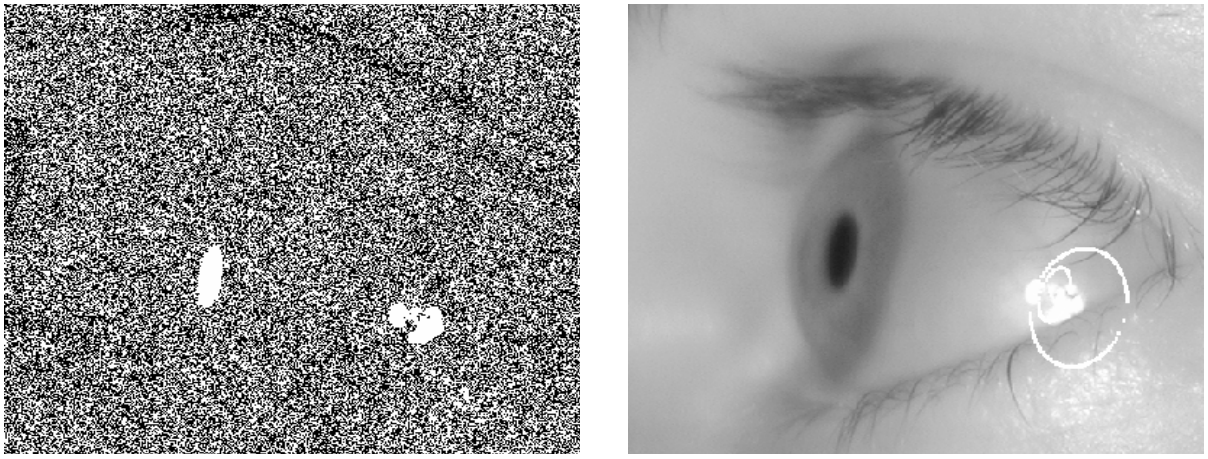


(a) (b) (c)

Figure 30: Iris image that failed to segment properly. (a) Iris's original least significant bit-plane. (b) Least significant bit-plane with borders removed. (c) Resulting segmentation attempt.

While some leniency was given as to iris placement, the iris does need to be located closer to the center of the image due to the border elimination that took place during the pupillary boundary location stage. As indicated earlier, the borders are removed from the bit-plane to help reduce the effects of glare in the pupil isolation stage. However, when an image of insufficient quality and placement is processed, the borders that are removed may contain the pupil, causing an FS.

Another source of iris FS was specularities in an image. Figure 31 shows a non-orthogonal image whose specularities were more prominent than the pupil. As the bit plane in Fig. 31 (a) demonstrates, the excessive glare from the near infrared camera caused the algorithm to assume that the specularities were, in fact, the pupil.



(a) (b)

Figure 31: (a) Iris least significant bit plane. (b) Iris image that failed to segment properly due to large specularities. The specularities were extracted as the pupil.

Despite these minor problems due to image quality, however, the overall FS percentages were low: 4.6% for orthogonal images and 1.7% for non-orthogonal images. FS images were not used in calculating the average quality factors.

The algorithms' performances using orthogonal and non-orthogonal irises were relatively comparable. On average, the orthogonal iris segmentation's quality factor was five percent greater than the non-orthogonal iris segmentation's quality factor, 0.8719 and 0.8208 respectively for a 10% error penalty. The non-orthogonal performance had a slightly better average variation, 0.0412 compared to 0.0443, which means that fewer occlusions (eyelids and eyelashes) were included in the non-orthogonal iris masks.

Performance of iris segmentation in reference to the overall iris recognition architectural framework (Fig. 10) indicates a successful module that can easily be incorporated into future

research, specifically modules for template encoding and database matching. Based on the partial iris research by Du *et al.* (Appendix F) and because the non-orthogonal segmentation algorithms provide upwards of 80% of the iris pattern on average, there is ample pattern available for accurate identification. Additionally, by improving the algorithms to remove eyelashes and eyelids that are incorrectly identified as iris pattern, the overall error of the algorithms will decrease and only pure iris pattern will remain.

5. Conclusions

The quantitative results presented were calculated using iris images in which the user was (1) looking directly at the imaging device (Table 1), and (2) looking at an angle non-orthogonal to the imaging device (Table 2). The purpose for approaching the issue of iris pattern extraction without the assumption that iris patterns were circular was to allow for the iris extraction of non-orthogonally captured iris images. Testing showed that a non-orthogonal approach to iris segmentation was possible as seen in Fig. 32.

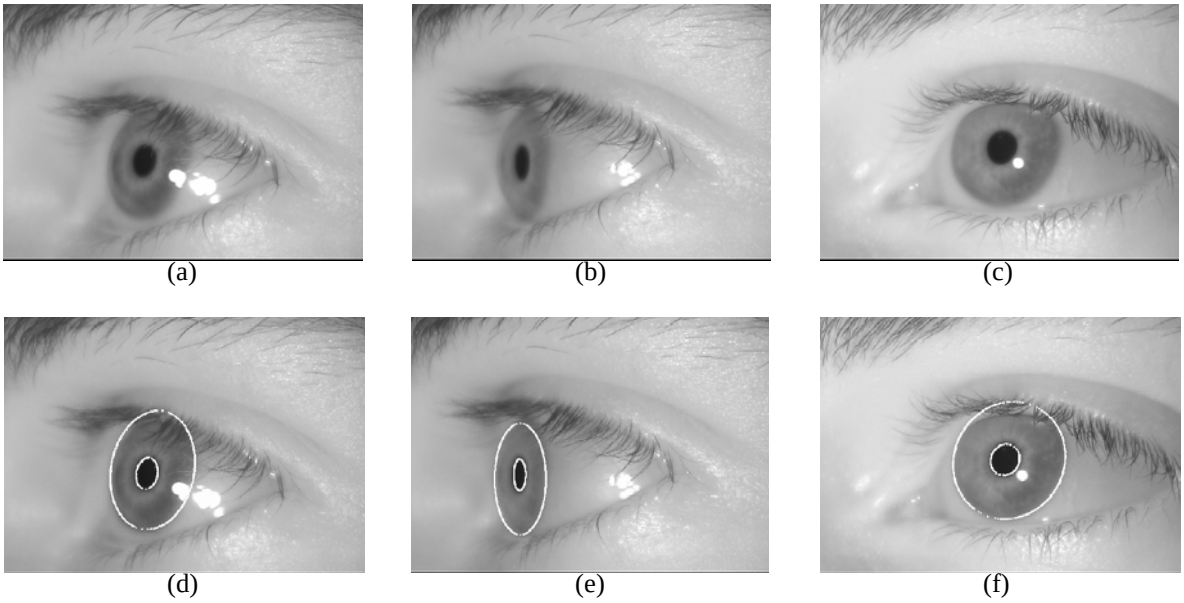


Figure 32: Iris images: (a)–(c) Non-orthogonal irises. (d)–(f) Resulting elliptical iris masks.

Because the iris images in Fig. 32 were rotated away from the normal with respect to the imaging device, current commercial systems developed complications in extracting the iris patterns. Non-orthogonal iris images were tested at the National Security Agency's Biometric Laboratory using research utilities provided by Iridian Technologies Inc., the patent holder on commercial iris recognition algorithms. The commercial system would not recognize that an iris pattern existed within the image, and an error was returned. However, results using the algorithms developed during this Trident research showed that it was possible to fit a close

elliptical approximation to non-orthogonal iris patterns using bit-planes and standard deviation windows.

Once successful elliptical approximations of non-orthogonal iris patterns are isolated, steps one and two of the biometric architectural framework are complete. The images have been successfully preprocessed, the pupillary boundary and limbic boundary determined, and the iris pattern removed. Future research, proposed for next year, will complete the remaining steps of iris code generation, comparison, and matching determination. The non-orthogonal iris pattern, extracted using the bit-plane and local standard deviation window techniques, will be used as the foundation for this research.

6. Works Cited

[1] Gonzalez, Rafael C., R. E. Woods, and Steven L. Eddins. *Digital Image Processing using MATLAB.* Upper Saddle River, New Jersey: Prentice Hall, 2004.

[2] Bock, Rudolf K., Hamming Distance. 7 April 1998. http://rkb.home.cern.ch/rkb/AN16pp/node114.html.

[3] *Biometric Solutions for Authentication in an E-World.* Edited by David Zhang. Boston: Kluwer Academic Publishers, 2002.

[4] Y. Du, R.W. Ives, D.M. Etter, and T.B. Welch, "A new approach to iris pattern recognition", *Proceedings of the SPIE European Symposium on Optics/Photonics in Defence and Security*, London, UK, October 2004.

[5] Nanavati, Raj, Samir Nanavati, and Michael Thieme. *Biometrics: Identity Verification in a Networked World.* New York: Wiley Computer Publishing, 2002.

[6] J. Forrester, A. Dick, P. McMenamin, and W. Lee, *The Eye: Basic Sciences in Practice*. London: W B Saunder, 2001.

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

[8] Y. Du, R. W. Ives, and D. M. Etter, "Iris Recognition", T*he Electrical Engineering Handbook*, 3rd Edition, Boca Raton, FL: CRC Press, 2005 (in press).

[9] J.D.Woodward, N.M.Orlans, and P.T.Higgins, *Biometrics*, California: McGraw-Hill, 2002.

[10] L. Flom and A. Safir, United States Patent No. 4,641,349 (issued February 3, 1987), *Iris Recognition System*, Washington D.C.: U.S. Government Printing Office.

[11] Y. Du, R. W. Ives, D. M. Etter, T. B. Welch, and C.-I Chang, "One Dimensional Approach to Iris Recognition", *Proceedings of SPIE, Volume 5404*, pp. 237-247, Apr., 2004.

[12] Y. Du, R. W. Ives, D. M. Etter, and T. B. Welch, "Use of One-Dimensional Iris Signatures to Rank Iris Pattern Similarities", submitted to *Optical Engineering*, 2004.

[13] R.P. Wildes, J.C. Asmuth, G.L. Green, S.C. Hsu, R.J. Kolczynski, J.R. Matey, and S.E. McBride, "A Machine Vision System for Iris Recognition", *Mach. Vision Application*, Vol. 9, 1-8, 1996.

[14] W.W. Boles and B. Boashash, "A Human Identification Technique Using Images of the Iris and Wavelet Transform", *IEEE Transactions on Signal Processing*, Vol. 46, No. 4, 1998.

[15] Y.-P. Huang, S.-W. Luo, E.-Y. Chen, "An Efficient Iris Recognition System", *Proceedings of the First International Conference on Machine Learning and Cybernetics*, pp. 450-454, 2002.

[16] Y. Zhu, T. Tan, and Y. Wnag, "Biometric Personal Identification Based on Iris Patterns", *Pattern Recognition, 15th International Conference on*, Vol. 2 , pp. 801 –804, 2000.

[17] L. Ma, Y. Wang, and T. Tan, "Iris Recognition Using Circular Symmetric Filters", *16th International Conference on Pattern Recognition*, Vol. 2 , pp. 414-417, 2002.

[18] Daugman, John G. "Biometric Personal Identification System Based on Iris Analysis", U.S. Patent 5,291,560. 1994.

[19] Daugman, John. How Iris Recognition Works. 25 Oct 2003. www.cl.cam.ac.uk/users/jgd1000/irisrecog.pdf.

[20] CASIA Iris Image Database, http://www.sinobiometrics.com

7. Works Consulted

[21] *Image Recognition and Classification: Algorithms, Systems, and Applications.* Edited by Bahram Javidi. New York: Marcel Dekker, Inc., 2002.

[22] *Intelligent Biometric Techniques in Fingerprint and Face Recognition.* Edited by L.C. Jain. New York: CRC Press LLC, 1999.

[23] Introduction to Iris Recognition. 25 Oct 2003. <http://www.iridiantech.com/basics.php>.

[24] Reid, Paul. *Biometrics for Network Security*. Upper Saddle River, New Jersey: Prentice Hall Professional Technical Reference, 2004.

[25] Williams, Gerald O. Iris Recognition White Paper: An Overview. 01 Jan 2001. 24 Oct 2003. http://www.politec.com/iris.

[26] Woodward, John D. Jr., Nicholas M. Olans, and Peter Higgins, *Biometric: Identity Assurance in the Information Age.* New York: McGraw-Hill, 2003.

[27] A. J. Bron, R. C. Tripathi, and B. J. Tripathi. *Wolff's Anatomy of the Eye and Orbit, 8th Edition.* London: Chapman and Hall Medical, 1997.

8. Appendices

Appendix A: MATLAB source code developed for iris extraction algorithms

Appendix B: Experimental Data

Appendix C: Consent and Information Form – U.S. Naval Academy Biometrics Laboratory

Appendix D: B. L. Bonney, R. W. Ives, and D. M. Etter, "Iris Pattern Extraction using Bit-Planes and Standard Deviations", *Proceedings Thirty-Eighth Asilomar Conference on Signals, Systems & Computers*, pp. 582-586. Pacific Grove, California. November 7-10, 2004.

Appendix E: Y. Du, B. L. Bonney, R. W. Ives, D. M. Etter, and R. Schultz, "Analysis of Partial Iris Recognition Using a 1-D Approach", *Proceedings 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Philadelphia, Pennsylvania. March 18-23, 2005.

Appendix F: Y. Du, R. W. Ives, B. L. Bonney, and D. M. Etter, "Analysis of Partial Iris Recognition", *Proceedings 2005 SPIE Defense and Security Symposium*, Volume 5779, pp. 31-40. Orlando, Florida. March 28-April 1, 2005.

Appendix G: R. W. Ives, B. L. Bonney, and D. M. Etter, "Effect of Image Compression on Iris Recognition", To be presented at the *2005 IEEE Instrumentation and Measurement Technology Conference.* Ontario, Canada. May 17-19, 2005.

Appendix A: MATLAB Source Code

Function List:

```
function z = add_iris_boundaryL(pupil_mask, iris_stats, top, bottom);

%This function takes a pupil mask, iris statistics, top, and bottom of the
%iris.  It then adds the elliplical shape of the iris to the logical pupil
%mask for the left half of the iris.
%
% usage: add_iris_boundaryL(pupil_mask, iris_stats, top, bottom);
%
%where pupil_mask = logical array holding the pupil mask, iris_stats =
%statistcs of the iris in a 1x1 structure, top = top boundary pixel of
%iris, and bottom = bottom boundary pixel of iris.

X0 = iris_stats.Centroid(1);
y0 = iris_stats.Centroid(2);
a = floor(iris_stats.MajorAxisLength / 2);
b = floor(iris_stats.MinorAxisLength / 2);

y = [top:1:bottom];
x1 = round(abs(sqrt((1-(y-y0).^2/b^2)*a^2) + x0));

point1 = [y; x1];

for I = 1:length(y)
   pupil_mask(point1(1, i), point1(2, i)) = 1;
end

z = pupil_mask;
```

```
function z = add_iris_boundaryR(pupil_mask, iris_stats, top, bottom);

%This function takes a pupil mask, iris statistics, top, and bottom of the
%iris.  It then adds the elliplical shape of the iris to the logical pupil
%mask for the right half of the iris.
%
% usage: add_iris_boundaryR(pupil_mask, iris_stats, top, bottom);
%
%where pupil_mask = logical array holding the pupil mask, iris_stats =
%statistcs of the iris in a 1x1 structure, top = top boundary pixel of
%iris, and bottom = bottom boundary pixel of iris.

X0 = iris_stats.Centroid(1);
y0 = iris_stats.Centroid(2);
a = floor(iris_stats.MajorAxisLength / 2);
b = floor(iris_stats.MinorAxisLength / 2);

y = [top:1:bottom];
x2 = round(-abs(sqrt((1-(y-y0).^2/b^2)*a^2)) + x0);

point2 = [y; x2];

for I = 1:length(y)
   pupil_mask(point2(1, i), point2(2, i)) = 1;
end

z = pupil_mask;
```

```
function z = add_pupil_boundaryL(pupil_mask, pupil_stats, top_edge, bottom_edge);

%This function takes a pupil mask, pupil statistics, top, and bottom of the
%pupil.  It then adds the elliplical shape of the iris to the logical pupil
%mask for the left half of the pupil.
%
%usage: add_pupil_boundaryL(pupil_mask, pupil_stats, top_edge, bottom_edge)
%
%where pupil_mask = logical array holding the pupil mask, pupil_stats =
%statistcs of the pupil in a 1x1 structure, top_edge = top boundary pixel
%of the pupil, and bottom_edge = bottom boundary pixel of the pupil.

X0 = pupil_stats.Centroid(1);
y0 = pupil_stats.Centroid(2);
a = floor(pupil_stats.MajorAxisLength / 2);
b = floor(pupil_stats.MinorAxisLength / 2);

y = [top_edge:1:bottom_edge];
x1 = round(abs(sqrt((1-(y-y0).^2/b^2)*a^2) + x0));
% x2 = round(-abs(sqrt((1-(y-y0).^2/b^2)*a^2)) + x0);

% equations below are the equivalent functions to those above...not needed
% x = [left:1:right];
% y1 = round(abs(sqrt((1-(x-x0).^2/a^2)*b^2) + y0));
% y2 = round(-abs(sqrt((1-(x-x0).^2/a^2)*b^2)) + y0);

point1 = [y; x1]; %point2 = [y; x2];

pupil_mask = zeros(size(pupil_mask));

for I = 1:length(y)
   pupil_mask(point1(1, i), point1(2, i)) = 1;
   %pupil_mask(point2(1, i), point2(2, i)) = 1;
end

% figure, gimage(pupil_mask), title('Iris Boundary');

z = pupil_mask;
```

```
function z = add_pupil_boundaryR(pupil_mask, pupil_stats, top_edge, bottom_edge);

%This function takes a pupil mask, pupil statistics, top, and bottom of the
%pupil.  It then adds the elliplical shape of the iris to the logical pupil
%mask for the right half of the pupil.
%
%usage: add_pupil_boundaryR(pupil_mask, pupil_stats, top_edge, bottom_edge)
%
%where pupil_mask = logical array holding the pupil mask, pupil_stats =
%statistcs of the pupil in a 1x1 structure, top_edge = top boundary pixel
%of the pupil, and bottom_edge = bottom boundary pixel of the pupil.

X0 = pupil_stats.Centroid(1);
y0 = pupil_stats.Centroid(2);
a = floor(pupil_stats.MajorAxisLength / 2);
b = floor(pupil_stats.MinorAxisLength / 2);

y = [top_edge:1:bottom_edge];
%x1 = round(abs(sqrt((1-(y-y0).^2/b^2)*a^2) + x0));
x2 = round(-abs(sqrt((1-(y-y0).^2/b^2)*a^2)) + x0);

% equations below are the equivalent functions to those above...not needed
% x = [left:1:right];
% y1 = round(abs(sqrt((1-(x-x0).^2/a^2)*b^2) + y0));
% y2 = round(-abs(sqrt((1-(x-x0).^2/a^2)*b^2)) + y0);

%point1 = [y; x1];
point2 = [y; x2];

%pupil_mask = zeros(size(pupil_mask));

for I = 1:length(y)
   %pupil_mask(point1(1, i), point1(2, i)) = 1;
   pupil_mask(point2(1, i), point2(2, i)) = 1;
end

% figure, gimage(pupil_mask), title('Iris Boundary');

z = pupil_mask;
```

```
function  bit0 = adjusted_bitzero(iris_image);

%This function takes an iris image and returns the least significant bit
%plane of that image.
%
%               usage: bit0 = adjusted_bitzero(iris_image)
%
%where iris_image is the image of the desired least significant bit plane.

[col, row] = size(iris_image);

adjusted_iris = norim(iris_image);

for i=1:col
   for j=1:row
      if adjusted_iris(I,j) < 60
         adjusted_iris(I,j) = 255;
      elseif adjusted_iris(I,j) > 240
         adjusted_iris(I,j) = 255;
      end
   end
end

bit0 = mod(double(adjusted_iris), 2);
```

```
function B = bottom_edge_test(b, v, b_e, old)

%This function takes the current location of the bottom of the iris, the
%vertical vector holding the thresholded iris data, the bottom edge of the
%iris, and the former value of the bottom edge. It compares the next
%potential iris edge to the current iris edge.  If the new value is within
%the default distance (N=20 pixels), the old value is replaced with the
%new. Otherwise, the old value is returned.  This function is used
%recursively.
%
%   usage: B = bottom_edge_test(b, v, b_e, old)
%
%where b is the current location of the bottom of the iris, v is the
%vertical vector holding the thresholded iris data, b_e is the bottom edge
%of the pupil, and old is the former value of the bottom edge used for
%comparison.

B = b; N = 20;

b_e_new = b;
b_bound = v((b_e_new):length(v));
[B_new_front, B_new_back] = middle_bottom_band(b_bound, b_e_new);
if B_new_front > B + N | B_new_front == old
   B = b;
else
   B = B_new_back;
   B = bottom_edge_test(B, v, b_e, B);
End
```

```
function [L,R,T,B] = get_iris_edges(h_test, v_test, l_e, r_e, t_e, b_e)

%This function takes the 1-D logical mask of the std deviations after they
%have undergone morphology as well. It returns the initial L,R,T,B edges of
%the IRIS.  The "test_iris_edges()" function later checks to ensure the
%values returned are the correct edges of the IRIS.
%
% usage: [L, R, T, B] = get_iris_edges(h_test, v_test, l_e, r_e, t_e, b_e);
%
%where L,R,T,B are the pixel locations of the IRIS boundary such that
%L = Left, R = Right, T = Top, B = Bottom.  H_test and v_test are the 1-D
%logical arrays displaying the standard deviations are you progress N,S,E,W
%from the center point of the iris.  L_e = left edge, r_e = right edge,
%t_e = top edge, b_e = bottom edge of PUPIL boundary.

N = 10;

l_bound = h_test(1 : (l_e – n));
r_bound = h_test((r_e + n) : length(h_test));
t_bound = v_test(1 : (t_e – n));
b_bound = v_test((b_e + n) : length(v_test));

[L, temp] = middle_left_band2(l_bound);

[R, temp] = middle_right_band2(r_bound, r_e);


if find(t_bound == 1)
   T = find(t_bound == 1);
   T = T(length(T));
else
   T = 1;
end

if find(b_bound == 1)
   B = find(b_bound == 1);
   B = B(1) + b_e + 15;
else
   B = 480;
end

if L < 1
   L = 1;
end
if R > 640
   R = 640;
end
```

```
if T < 1
   T = 1;
end
if B > 480
   B = 480;
end
```

```
function y = get_mask(eye)

%This function takes an image of the eye in which the “ground truth” has
%been determined in Adobe Photoshop. The iris consists of a grouping
%of pixels with similar values equal to 255. This function returns a binary
%mask of the iris mask that represent the location of the iris pixels
%
%       usage: y = get_mask(eye)
%
%where eye is the “ground truth” iris image.

Eye_mask = zeros(480, 640);

eye_mask( find(eye == 255) ) = 1;

L = bwlabel(eye_mask);
stats = regionprops(L, ‘Area’, ‘PixelList’);
[A B] = size(stats);

%%% FIND LARGEST BLOCK OF ONES!
Largest = stats(1);
for I = 2:A
   if stats(i).Area > largest.Area
      largest = stats(i);
   end
end

blank = zeros(480, 640);
for I = 1:length(largest.PixelList)
   blank(largest.PixelList(I, 2), largest.PixelList(I, 1)) = 1;
end

y = blank;
```

```
function [L,R,T,B] = get_pupil_edges(pupil_mask, pupil_stats);

%This function takes the logical pupil mask and the pupil center pixel. It
%returns the L,R,T,B coordinates of the pupil edge.
%
% usage: [L, R, T, B] = get_pupil_edges(pupil_mask, pupil_center);
%
%where L,R,T,B are the pixel locations of the pupil boundary such that
%L = Left Edge, R = Right Edge, T = Top Edge, B = Bottom Edge

pupil_center = round(pupil_stats.Centroid);

horizontal = find(pupil_mask(pupil_center(2),☺ == 1);

L = horizontal(1);
R = horizontal(length(horizontal));

vertical = find(pupil_mask(:,pupil_center(1)) == 1);

T = vertical(1);
B = vertical(length(vertical));
```

```
function gimage(x)

%This function displays an image using the grayscale colormap.
%
%    usage: gimage(x)
%
%where x is the image desired to be displayed in grayscale.

%converts image values to doubles for math operations
d = double(x);

%normalizes image x for a range of [0,255]
norim(d);

%displays x
colormap(gray(256));
image(round(d));
%imwrite(x, 'image.jpeg');
```

```
function [y, stats] = iris_segmentation(iris);

%This function performs an iris segmentation using elliptical curves to
%define the pupillary and limbic boundaries.
%
%   usage: y = iris_segmentation(iris)
%
%where iris is a 480 x 640 uint8 grayscale eye image.

Bitplane_zero = adjusted_bitzero(iris);

[pupil_mask, stats] = pupil_morph2(bitplane_zero);

[y, stats] = limbic_boundary6(stats, pupil_mask, iris);
```

```
function L = left_edge_test(l, h, l_e, Lold)

%This function takes the current location of the left of the iris, the
%horizontal vector holding the thresholded iris data, the left edge of the
%iris, and the former value of the left edge. It compares the next
%potential iris edge to the current iris edge.  If the new value is within
%the default distance (N=20 pixels), the old value is replaced with the
%new. Otherwise, the old value is returned.  This function is used
%recursively.
%
%   usage: L = left_edge_test(l, h, l_e, Lold)
%
%where l is the current location of the left side of the iris, h is the
%horizontal vector holding the thresholded iris data, l_e is the left edge
%of the pupil, and Lold is the former value of the left edge used for
%comparison.

L = l;N = 20;

l_e_new = l;
l_bound = h(1 : (l_e_new));
[L_new_front, L_new_back] = middle_left_band2(l_bound);

if L_new_front < L – N | L_new_front == Lold
   L = l;
else
   L = L_new_back;
   L = left_edge_test(L, h, l_e, L);
end
```

```
function [y, stats] = limbic_boundary6(pupil_stats, pupil_mask, iris_image);

%This function takes the pupil statistics (include centroid, major and
%minor axis), the logical binary pupil mask, and the original iris image.
%The function returns a new binary mask which contains both the pupil mask
%as well as the elliptical limbic boundary.
%
% usage: y = limbic_boundary6(pupil_stats, pupil_mask, iris_image)
%
%where pupil_stats is the 1x1 structure containing the pupil statistics,
%pupil_mask is the binary mask of the location of the pupil boundary, and
%iris_image is the image from which the iris pattern is being extracted.

Pupil_center = round(pupil_stats.Centroid);
[rows, cols] = size(pupil_mask);

%plots each half of pupillary boundary separately
[left_edge, right_edge, top_edge, bottom_edge] = get_pupil_edges(pupil_mask, pupil_stats);

right_side = right_edge;
fake_left_side = pupil_stats.Centroid(1) – (right_edge – pupil_stats.Centroid(1));

left_side = left_edge;
fake_right_side = pupil_stats.Centroid(1) + (pupil_stats.Centroid(1) – left_edge);

p_s1 = pupil_stats;
p_s2 = pupil_stats;

p_s1.MajorAxisLength = right_side – fake_left_side;
p_s1.MinorAxisLength = bottom_edge – top_edge;

p_s2.MajorAxisLength = fake_right_side – left_side;
p_s2.MinorAxisLength = bottom_edge – top_edge;

pupil_mask = add_pupil_boundaryL(pupil_mask, p_s1, top_edge, bottom_edge);
pupil_mask = add_pupil_boundaryR(pupil_mask, p_s2, top_edge, bottom_edge);

%mex localthresh.c
thresh_image = ~localthresh(double(iris_image), 45, 4);
thresh_image = bwmorph(thresh_image, ‘open’);

v_test = thresh_image(:,pupil_center(1));
h_test = thresh_image(pupil_center(2),☺;

%find iris edges
[left, right, top, bottom] = get_iris_edges(h_test, v_test, left_edge, ...
                        right_edge, top_edge, bottom_edge);
```

```
%calculate new major and minor axis for iris (different than pupil)
iris_stats = pupil_stats;
iris_stats.MajorAxisLength = right – left;
iris_stats.MinorAxisLength = bottom – top;
iris_stats.Centroid = [floor((right – left)/2 + left) ...
                    floor((bottom – top)/2 + top)];

% test to see if bounds are actual iris boundaries
[left, right, top, bottom, iris_stats] = test_iris_edges2(left, right, ...
   top, bottom, pupil_stats, iris_stats, h_test, v_test, left_edge, ...
   right_edge, top_edge, bottom_edge);

%% ADD ELLIPTICAL BOUNDARY %%
right_side = right;
fake_left_side = iris_stats.Centroid(1) – (right – iris_stats.Centroid(1));

left_side = left;
fake_right_side = iris_stats.Centroid(1) + (iris_stats.Centroid(1) – left);

i_s1 = iris_stats;
i_s2 = iris_stats;

i_s1.MajorAxisLength = right_side – fake_left_side;
i_s1.MinorAxisLength = bottom – top;

i_s2.MajorAxisLength = fake_right_side – left_side;
i_s2.MinorAxisLength = bottom – top;

iris_mask = zeros(rows, cols);
iris_mask = add_iris_boundaryL(iris_mask, i_s1, top, bottom);
iris_mask = add_iris_boundaryR(iris_mask, i_s2, top, bottom);

%Rotate the mask to adjust to fit the tilt of the iris in the image
orient = 90;

if abs(pupil_stats.MajorAxisLength – pupil_stats.MinorAxisLength) > 7
   orient = pupil_stats.Orientation;
end

left_bound = round(pupil_stats.Centroid(1) – pupil_stats.MajorAxisLength);
right_bound = round(pupil_stats.Centroid(1) + pupil_stats.MajorAxisLength);
top_bound = round(pupil_stats.Centroid(2) – pupil_stats.MajorAxisLength);
bot_bound = round(pupil_stats.Centroid(2) + pupil_stats.MajorAxisLength);
```

```
%%ROTATES PUPIL MASK
sub_mask = pupil_mask(top_bound:bot_bound, left_bound:right_bound);
sub_mask = imrotate(sub_mask, -(90-orient), 'nearest', 'crop');
pupil_mask = zeros(rows, cols);
pupil_mask(top_bound:bot_bound, left_bound:right_bound) = sub_mask;

iris_stats = pupil_stats;
iris_stats.MajorAxisLength = right – left;
iris_stats.MinorAxisLength = bottom – top;
iris_stats.Centroid = [floor((right – left)/2 + left) ...
                floor((bottom – top)/2 + top)];

rotate_length = iris_stats.MinorAxisLength;
if iris_stats.MajorAxisLength > rotate_length
   rotate_length = iris_stats.MajorAxisLength;
end

left_bound = round(iris_stats.Centroid(1) – rotate_length/2 – 5);
right_bound = round(iris_stats.Centroid(1) + rotate_length/2 + 5);
top_bound = round(iris_stats.Centroid(2) – rotate_length/2 – 5);
bot_bound = round(iris_stats.Centroid(2) + rotate_length/2 + 5);

if left_bound < 6 || right_bound > 634 || top_bound < 6 || bot_bound > 474
   final_mask = pupil_mask + iris_mask;
else
   sub_mask2 = iris_mask(top_bound:bot_bound, left_bound:right_bound);
   sub_mask2 = imrotate(sub_mask2, -(90-orient), 'nearest', 'crop');
   iris_mask2 = zeros(rows, cols);
   iris_mask2(top_bound:bot_bound, left_bound:right_bound) = sub_mask2;

   final_mask = pupil_mask + iris_mask2;
end

y = final_mask;
stats = p_s1;
```

```
function [ymean,ystd,yvar]=localstats2(x,nsize)
%
%   function [ymean,ystd,yvar]=localstats2(x,nsize)
%
%   This function takes in an input 2D array x and outputs two 2D arrays:
%   the first is an array of the local means in nsize x nsize
%   neighborhoods, and the second is the local std deviations.

[r c]=size(x);
ymean=zeros(size(x));
ystd=zeros(size(x));
yvar=zeros(size(x));
for i=1:r
   for j=1:c
      z=extract_ives(x,I,j,nsize);
      ymean(I,j)=mean(z(☺);
      ystd(I,j)=std(z(☺);
      yvar(I,j)=var(z(☺);
   end
end
```

```
/*
 * localstd.c – Computes the local standard deviation about each pixel in an nsize x nsize
 *neighborhood. The nsize should be an odd number.
 *
 * MATLAB usage: y=localstd(x,nsize)
 *
 */

#include <math.h>
#include “mex.h”

#define ROUND(A) ((fabs(A)) >= ((int)A+0.5) ? ((int)A+1.0) : ((int)A))  /* this is the rounding
operation for + ints */
#define MIN(A,B) (A) < (B) ? (A)☹B)
#define MAX(A,B) (A) > (B) ? (A)☹B)

void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
{
  int rows,cols,nsize;

  int I,j,ii,jj,N,halfsize,index;
  int rowstart,rowstop,colstart,colstop;

  double D,*y,*yy,*n,*outArray;
  double ave,sdev,s;
  double *ptr;

  if (mxIsDouble(prhs[0]))

ptr=(double *)mxGetPr(prhs[0]);
else
mexErrMsgTxt(“Error: in function \”localstd\”, input matrix must be double.”);

  /* Do some error checking */
  if (nrhs != 2)
     mexErrMsgTxt(“Not enough input arguments.”);
   else if (nlhs > 1)
     mexErrMsgTxt(“Too many output arguments.”);


  /* Create a matrix for the return argument */
  rows=mxGetN(prhs[0]);   /* note: because MATLAB is column ordered, interchange rows &
cols */
  cols=mxGetM(prhs[0]);
  nsize=mxGetScalar(prhs[1]);
  if (floor(nsize/2)== (double)nsize/2)
     mexErrMsgTxt(“Error: in function \”localstd\”, neighborhood size must be odd.”);
```

```
  N=nsize*nsize;
/* # elements in the neighborhood */
  halfsize=(nsize-1)/2;

/* # pixels on either side of a pixel, defines neighborhood */

  n = (double *) mxMalloc(N*sizeof(double));

/* create storage for output */

  /*y=mxMalloc(rows*cols*sizeof(double)); */

   plhs[0]=mxCreateDoubleMatrix(cols,rows,mxREAL);
   y=mxGetPr(plhs[0]);

  /*yy=mxCreateDoubleMatrix(rows,cols,mxREAL);
  y=mxGetPr(yy);*/
  /* compute local standard deviation */
  for (i=0;i<rows;i++)
  for (j=0;j<cols;j++)
  {

/*printf("loop, [%d][%d]\n",I,j);*/

index=0;

rowstart=MAX(0,i-halfsize);

rowstop=MIN(rows-1,i+halfsize);

colstart=MAX(0,j-halfsize);

colstop=MIN(cols-1,j+halfsize);

/*

printf("here, rowstart=[%d], rowstop=[%d], colstart=[%d],
colstop[%d]\n",rowstart,rowstop,colstart,colstop);

*/

for (ii=rowstart;ii<=rowstop;ii++)

for (jj=colstart;jj<=colstop;jj++)

n[index++]=ptr[ii*cols+jj];
```

```
        ave=sdev=0.0;
        for (ii=0;ii<index;ii++)
ave += n[ii];

ave /= index;
        for (ii=0;ii<index;ii++)
{
s=n[ii]-ave;

sdev += s*s;
}

sdev /= (index-1);
sdev=sqrt(sdev);

y[i*cols+j]=sdev;

/*
  printf("y[%d][%d] = ? y[%d]\n",I,j,i*cols+j);
*/
  }
/*
plhs[0]=mxCreateDoubleMatrix(rows,cols,mxREAL);
outArray = mxGetPr(plhs[0]);
memcpy(outArray,y,rows*cols*sizeof(double));
mxFree(n);
*/
}
```

```
/*
 * localthresh.c – Thresholds an image based on a local mean about each pixel in an nsize x nsize
 *neighborhood. The nsize should be an odd number. C is a constant.
 *T = localmean – C
 * MATLAB usage: y=localthresh(x,nsize,C)
 *
 *
 *Written by Robert Ives, US Naval Academy, 21 Oct 04
 */

#include <math.h>
#include “mex.h”

#define ROUND(A) ((fabs(A)) >= ((int)A+0.5) ? ((int)A+1.0) : ((int)A))  /* this is the rounding
operation for + ints */
#define MIN(A,B) (A) < (B) ? (A)☹B)
#define MAX(A,B) (A) > (B) ? (A)☹B)

void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
{
  int rows,cols,nsize;

  int I,j,ii,jj,N,halfsize,index;
  int rowstart,rowstop,colstart,colstop;

  double D,*y,*yy,*n,*outArray;
  double ave,s,C,T;
  double *ptr;

  if (mxIsDouble(prhs[0]))

ptr=(double *)mxGetPr(prhs[0]);
else
mexErrMsgTxt(“Error: in function \”localstd\”, input matrix must be double.”);

  /* Do some error checking */
  if (nrhs != 3)
     mexErrMsgTxt(“Not enough input arguments.”);
   else if (nlhs > 1)
     mexErrMsgTxt(“Too many output arguments.”);

  /* Create a matrix for the return argument */
  rows=mxGetN(prhs[0]);   /* note: because MATLAB is column ordered, interchange rows &
cols */
  cols=mxGetM(prhs[0]);
  nsize=mxGetScalar(prhs[1]);
  C=mxGetScalar(prhs[2]);
```

```
  if (floor(nsize/2)== (double)nsize/2)
     mexErrMsgTxt("Error: in function \"localstd\", neighborhood size must be odd.");

  N=nsize*nsize;

/* # elements in the neighborhood */
  halfsize=(nsize-1)/2;

/* # pixels on either side of a pixel, defines neighborhood */

  n = (double *) mxMalloc(N*sizeof(double));
 /* create storage for output */

  /*y=mxMalloc(rows*cols*sizeof(double)); */

   plhs[0]=mxCreateDoubleMatrix(cols,rows,mxREAL);
   y=mxGetPr(plhs[0]);

  /*yy=mxCreateDoubleMatrix(rows,cols,mxREAL);
  y=mxGetPr(yy);*/
  /* compute local standard deviation */
  for (i=0;i<rows;i++)
                                                                          for
(j=0;j<cols;j++)
  {

/*printf("loop, [%d][%d]\n",I,j);*/

index=0;

rowstart=MAX(0,i-halfsize);

rowstop=MIN(rows-1,i+halfsize);

colstart=MAX(0,j-halfsize);

colstop=MIN(cols-1,j+halfsize);


for (ii=rowstart;ii<=rowstop;ii++)

for (jj=colstart;jj<=colstop;jj++)
n[index++]=ptr[ii*cols+jj];

        ave=0.0;
        for (ii=0;ii<index;ii++)
```

```
ave += n[ii];

ave /= index;
        y[i*cols+j]=0;

T=ave-C;

if (ptr[i*cols+j] > T)
          y[i*cols+j]=1;
                    }

/*
plhs[0]=mxCreateDoubleMatrix(rows,cols,mxREAL);
outArray = mxGetPr(plhs[0]);
memcpy(outArray,y,rows*cols*sizeof(double));
mxFree(n);
*/

}
```

```
function [first, last] = middle_bottom_band(b_bound, bottom_edge)

%This function takes a vertical vector with the thresholded iris data and
%returns the first and last location of the next band of 'ones'.
%
%   usage: [first, last] = middle_bottom_band(b_bound, bottom_edge)
%
%where b_bound is the logical array beneath the iris and extending to the
%end of the image, and bottom_edge is the next potential end of the band of
%'ones'. It is a one dimensional array that is focused about the center
%pixel of the iris.

Counter = 0;

if sum(b_bound(☺) == 0
   first = bottom_edge;
   last = bottom_edge;
   return;
end

while b_bound(1) ~= 1
   b_bound = b_bound(2:length(b_bound));
   counter = counter + 1;
end

first = bottom_edge + counter;

while b_bound(1) == 1 && length(b_bound) >= 2
   b_bound = b_bound(2:length(b_bound));
   counter = counter + 1;
end

if sum(b_bound(☺) == 0 || length(b_bound) == 1
   last = bottom_edge + counter;
   return;
end

last = bottom_edge + counter;
```

```
function [first, last] = middle_left_band2(l_bound);

%This function takes a horizontal vector with the thresholded iris data and
%returns the first and last location of the next band of 'ones'.
%
%   usage: [first, last] = middle_left_band2(l_bound)
%
%where l_bound is the logical array beneath the iris and extending to the
%end of the image. It is a one dimensional array that is focused about the
%center pixel of the iris.

Left_edge = length(l_bound);

if sum(l_bound(☺) == 0
   first = left_edge;
   last = left_edge;
   return;
end

while l_bound(length(l_bound)) ~= 1
   l_bound = l_bound(1:length(l_bound)-1);
end

first = length(l_bound);

while l_bound(length(l_bound)) == 1 && length(l_bound) >= 2
   l_bound = l_bound(1:length(l_bound)-1);
end

if sum(l_bound(☺) == 0 || length(l_bound) == 1
   last = length(l_bound);
   return;
end

last =length(l_bound);
```

```
function [first, last] = middle_right_band2(r_bound, right_edge)

%This function takes a horizontal vector with the thresholded iris data and
%returns the first and last location of the next band of ‘ones’.
%
%   usage: [first, last] = middle_right_band2(r_bound, right_edge)
%
%where r_bound is the logical array beneath the iris and extending to the
%end of the image and right_edge is the next potential edge of the band of
%’ones’. It is a one dimensional array that is focused about the center
%pixel of the iris.

Counter = 0;

if sum(r_bound(☺) == 0
   first = right_edge;
   last = right_edge;
   return;
end

while r_bound(1) ~= 1
   r_bound = r_bound(2:length(r_bound));
   counter = counter + 1;
end

first = right_edge + counter;

while r_bound(1) == 1 && length(r_bound) >= 2
   r_bound = r_bound(2:length(r_bound));
   counter = counter + 1;
end

if sum(r_bound(☺) == 0 || length(r_bound) == 1
   last = right_edge + counter;
   return;
end

last = right_edge + counter;
```

```
function [first, last] = middle_top_band3(t_bound);

%This function takes a vertical vector with the thresholded iris data and
%returns the first and last location of the next band of 'ones'.
%
%   usage: [first, last] = middle_top_band3(t_bound)
%
%where b_bound is the logical array beneath the iris and extending to the
%end of the image. It is a one dimensional array that is focused about the
%center pixel of the iris.


Top_edge = length(t_bound);

if sum(t_bound(☺) == 0
   first = top_edge;
   last = top_edge;
   return;
end

while t_bound(length(t_bound)) ~= 1
   t_bound = t_bound(1:length(t_bound)-1);
end

first = length(t_bound);

while t_bound(length(t_bound)) == 1 && length(t_bound) >= 2
   t_bound = t_bound(1:length(t_bound)-1);
end

if sum(t_bound(☺) == 0 || length(t_bound) == 1
   last = length(t_bound);
   return;
end

last =length(t_bound);
```

```
function y = norim(x)

%This function normalizes an image to integer values in the range [0,255].
%
%       usage: y = norim(x)
%
%where x is the image desired to be normalized.

D = double(x);

%Divide the array by the largest value in the array
y = d – min(d(☺);
y = y/max(y(☺);

%Multiply array y by 255 to put on [0,255] range
y = uint8(y*255);
```

```
function [y, largest] = pupil_morph2(bitplane_iris);

%This function performs the necessary morphological operations of a given
%least significant iris bit plane, and it returns the logical pupil mask
%for that image.
%
% usage: [y, largest] = pupil_morph2(bitplane_iris)
%
%where bitplane_iris is the least significant bit plane of the iris image.
%The iris image should have been fed through the 'adjusted_bitzero()'
%function prior to use for proper results.  The image must be a 640x480
%iris image.

Bw1 = bitplane_iris;
[cols, rows] = size(bw1);

bw_pupil = zeros(cols, rows);

for I = 125:355              %isolation used to reduce high values caused
   for j = 150:490           %by lighting – focus on pupil area
      bw_pupil(I,j) = bw1(I,j);
   end
end

%initial erosion and dilation to remove specularities
bw_pupil = bwmorph(bw_pupil, 'open');

L = bwlabel(bw_pupil);
stats = regionprops(L, 'Centroid', 'MajorAxisLength', 'MinorAxisLength', 'Area', 'PixelList',
'Orientation');
[A B] = size(stats);

%%% FIND LARGEST BLOCK OF ONES!
Largest = stats(1);
for I = 2:A
   if stats(i).Area > largest.Area
      largest = stats(i);
   end
end

blank = zeros(cols, rows);
for I = 1:length(largest.PixelList)
   blank(largest.PixelList(I, 2), largest.PixelList(I, 1)) = 1;
end

y = blank;
```

```
function R = right_edge_test(r, h, r_e, old)

%This function takes the current location of the right of the iris, the
%horizontal vector holding the thresholded iris data, the right edge of the
%iris, and the former value of the right edge. It compares the next
%potential iris edge to the current iris edge.  If the new value is within
%the default distance (N=20 pixels), the old value is replaced with the
%new. Otherwise, the old value is returned.  This function is used
%recursively.
%
%   usage: R = left_edge_test(r, h, r_e, old)
%
%where r is the current location of the right side of the iris, h is the
%horizontal vector holding the thresholded iris data, r_e is the right edge
%of the pupil, and old is the former value of the right edge used for
%comparison.

R = r; N = 20;

r_e_new = r;
r_bound = h((r_e_new) : length(h));
[R_new_front, R_new_back] = middle_right_band2(r_bound, r_e_new);

if R_new_front > R + N | R_new_front == old
   R = r;
else
   R = R_new_back;
   R = right_edge_test(R, h, r_e, R);
end
```

```
function varargout = Segmentation_GUI(varargin)

% SEGMENTATION_GUI M-file for Segmentation_GUI.fig
%      SEGMENTATION_GUI, by itself, creates a new SEGMENTATION_GUI or raises the
%      existing singleton*.
%
%      H = SEGMENTATION_GUI returns the handle to a new SEGMENTATION_GUI or the
%      handle to the existing singleton*.
%
%      SEGMENTATION_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SEGMENTATION_GUI.M with the given input
%      arguments.
%
%      SEGMENTATION_GUI('Property','Value',...) creates a new SEGMENTATION_GUI or
%      raises the existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Segmentation_GUI_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Segmentation_GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help Segmentation_GUI

% Last Modified by GUIDE v2.5 17-Feb-2005 14:16:53

% Begin initialization code – written by The MathWorks, Inc. – DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Segmentation_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Segmentation_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code – DO NOT EDIT
```

```
% --- Executes just before Segmentation_GUI is made visible.
Function Segmentation_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Segmentation_GUI (see VARARGIN)

% Choose default command line output for Segmentation_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Segmentation_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Turn off initial axes1 and axes2 for GUI execution
set(handles.axes1, 'HandleVisibility', 'ON');
axes(handles.axes1);
axis off;
title(' ');
set(handles.axes1, 'HandleVisibility', 'OFF');

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
axis off;
title(' ');
set(handles.axes2, 'HandleVisibility', 'OFF');

set(handles.text4, 'String', ' ');
set(handles.numtruepixel, 'String', ' ');
set(handles.text6, 'String', ' ');
set(handles.nummaskpixel, 'String', ' ');
set(handles.text10, 'String', ' ');
set(handles.lowqual, 'String', ' ');
set(handles.text12, 'String', ' ');
set(handles.upperqual, 'String', ' ');
set(handles.text15, 'String', ' ');
set(handles.text18, 'String', ' ');
set(handles.topqual, 'String', ' ');
set(handles.text16, 'String', ' ');
set(handles.numcommonpixel, 'String', ' ');

mex localthresh.c
```

```
% --- Outputs from this function are returned to the command line.
Function varargout = Segmentation_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in loadraw.
Function loadraw_Callback(hObject, eventdata, handles)
% hObject    handle to loadraw (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.text4, 'String', ' ');
set(handles.numtruepixel, 'String', ' ');
set(handles.text6, 'String', ' ');
set(handles.nummaskpixel, 'String', ' ');
set(handles.text10, 'String', ' ');
set(handles.lowqual, 'String', ' ');
set(handles.text12, 'String', ' ');
set(handles.upperqual, 'String', ' ');
set(handles.text15, 'String', ' ');
set(handles.text18, 'String', ' ');
set(handles.topqual, 'String', ' ');
set(handles.text16, 'String', ' ');
set(handles.numcommonpixel, 'String', ' ');

global iris_image;
filename = get(handles.filename, 'String');

iris_image = imread(filename);

set(handles.axes1, 'HandleVisibility', 'ON');
axes(handles.axes1);
gimage(norim(iris_image)), axis image;
axis off;
set(handles.axes1, 'HandleVisibility', 'OFF');

global truth_mask;
global iris_image;

truth_mask = get_mask(iris_image);
```

```
% --- Executes on button press in segment.
Function segment_Callback(hObject, eventdata, handles)
% hObject    handle to segment (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global iris_image2;
global iris_mask;
global stats;

set(handles.text4, 'String', ' ');
set(handles.numtruepixel, 'String', ' ');
set(handles.text6, 'String', ' ');
set(handles.nummaskpixel, 'String', ' ');
set(handles.text10, 'String', ' ');
set(handles.lowqual, 'String', ' ');
set(handles.text12, 'String', ' ');
set(handles.upperqual, 'String', ' ');
set(handles.text15, 'String', ' ');
set(handles.text18, 'String', ' ');
set(handles.topqual, 'String', ' ');
set(handles.text16, 'String', ' ');
set(handles.numcommonpixel, 'String', ' ');

set(handles.text15, 'String', 'Segmenting...');
pause(0.01);

[iris_mask, stats] = iris_segmentation(iris_image2);

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
temp = uint8(bwmorph(iris_mask, 'dilate', 1));
gimage(norim(norim(uint8(temp)) + iris_image2)), axis image;
axis off;
set(handles.axes2, 'HandleVisibility', 'OFF');

set(handles.text15, 'String', ' ');

% --- Executes on button press in Reset.
Function Reset_Callback(hObject, eventdata, handles)
% hObject    handle to Reset (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.axes1, 'HandleVisibility', 'ON');
axes(handles.axes1);
cla reset;
```

```
axis off;
title(‘ ‘);
set(handles.axes1, ‘HandleVisibility’, ‘OFF’);

set(handles.axes2, ‘HandleVisibility’, ‘ON’);
axes(handles.axes2);
cla reset;
axis off;
title(‘ ‘);
set(handles.axes2, ‘HandleVisibility’, ‘OFF’);

set(handles.text4, ‘String’, ‘ ‘);
set(handles.numtruepixel, ‘String’, ‘ ‘);
set(handles.text6, ‘String’, ‘ ‘);
set(handles.nummaskpixel, ‘String’, ‘ ‘);
set(handles.text10, ‘String’, ‘ ‘);
set(handles.lowqual, ‘String’, ‘ ‘);
set(handles.text12, ‘String’, ‘ ‘);
set(handles.upperqual, ‘String’, ‘ ‘);
set(handles.text15, ‘String’, ‘ ‘);
set(handles.text18, ‘String’, ‘ ‘);
set(handles.topqual, ‘String’, ‘ ‘);
set(handles.text16, ‘String’, ‘ ‘);
set(handles.numcommonpixel, ‘String’, ‘ ‘);

% --- Executes during object creation, after setting all properties.
Function filename_CreateFcn(hObject, eventdata, handles)
% hObject    handle to filename (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    empty – handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
If ispc
   set(hObject,’BackgroundColor’,’white’);
else
   set(hObject,’BackgroundColor’,get(0,’defaultUicontrolBackgroundColor’));
end

% --- Executes on button press in calculate.
Function calculate_Callback(hObject, eventdata, handles)
% hObject    handle to calculate (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global iris_image;
global iris_image2;
```

```
global truth_mask;
global iris_mask;
global stats;

%fill mask
n = 4;
temp_mask = ones(480, 640);
while sum(~temp_mask(☺) == 0
   temp_mask = bwmorph(iris_mask, ‘dilate’, n);
   temp_mask = imfill(temp_mask, [1 1]);
   location = round([stats.Centroid(2) stats.Centroid(1)]);
   temp_mask = imfill(temp_mask, location);
   n = n + 1;
end

iris_mask = ~temp_mask;
iris_mask = bwmorph(iris_mask, ‘dilate’, n-2);

set(handles.axes2, ‘HandleVisibility’, ‘ON’);
axes(handles.axes2);
gimage(norim(norim(uint8(iris_mask)) + iris_image2)), axis image;
axis off;
set(handles.axes2, ‘HandleVisibility’, ‘OFF’);

%%
combo = truth_mask & iris_mask;
num_common_pixels = sum(combo(☺);
num_true_pixels = sum(truth_mask(☺);
num_mask_pixels = sum(iris_mask(☺);
num_error_pixels = num_mask_pixels – num_common_pixels;
if num_error_pixels < 0
   num_error_pixels = 0;
end

low = (num_common_pixels – 0.1 * num_error_pixels) / num_true_pixels;
mid = (num_common_pixels – 0.4 * num_error_pixels) / num_true_pixels;;
top = (num_common_pixels – 0.7 * num_error_pixels) / num_true_pixels;;

set(handles.text4, ‘String’, ‘Number of True Iris Pixels:’);
set(handles.numtruepixel, ‘String’, num_true_pixels);
set(handles.text6, ‘String’, ‘Number of Mask Iris Pixels:’);
set(handles.nummaskpixel, ‘String’, num_mask_pixels);
set(handles.text10, ‘String’, ‘10% Quality Bound:’);
set(handles.lowqual, ‘String’, low);
set(handles.text12, ‘String’, ‘40% Quality Bound:’);
set(handles.upperqual, ‘String’, mid);
set(handles.text18, ‘String’, ‘70% Quality Bound:’);
```

```
set(handles.topqual, 'String', top);
set(handles.text16, 'String', 'Number of Common Pixels:');
set(handles.numcommonpixel, 'String', num_common_pixels);


% --- Executes on button press in loadiris2.
Function loadiris2_Callback(hObject, eventdata, handles)
% hObject    handle to loadiris2 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.text4, 'String', ' ');
set(handles.numtruepixel, 'String', ' ');
set(handles.text6, 'String', ' ');
set(handles.nummaskpixel, 'String', ' ');
set(handles.text10, 'String', ' ');
set(handles.lowqual, 'String', ' ');
set(handles.text12, 'String', ' ');
set(handles.upperqual, 'String', ' ');
set(handles.text15, 'String', ' ');
set(handles.text18, 'String', ' ');
set(handles.topqual, 'String', ' ');
set(handles.text16, 'String', ' ');
set(handles.numcommonpixel, 'String', ' ');

global iris_image2;

filename2 = get(handles.edit2, 'String');
iris_image2 = imread(filename2);

set(handles.axes2, 'HandleVisibility', 'ON');
axes(handles.axes2);
gimage(norim(iris_image2)), axis image;
axis off;
set(handles.axes2, 'HandleVisibility', 'OFF');

% --- Executes during object creation, after setting all properties.
Function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved – to be defined in a future version of MATLAB
% handles    empty – handles not created until after all CreateFcns called

if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function [L,R,T,B,I] = test_iris_edges2(l, r, t, b, p_stats, i_stats, h,...
                                        v, l_e, r_e, t_e, b_e)

%This function tests for and returns the limbic boundaries of the iris
%given initial starting conditions.
%
% usage: [L,R,T,B,I] = test_iris_edges2(l, r, t, b, p_stats, i_stats, h,...
%                                       v, l_e, r_e, t_e, b_e)
%
%where l is the initial left edge, r is the initial right edge, t is the
%initial top edge, and b is the initial bottom edge of the iris.  P_stats
%and i_stats are the 1x1 structs that contain the statistics for the pupil
%and iris respectively. H is the horizontally thresholded iris data vector
%and v is the vertically thresholded iris data vector. L_e is the left edge
%of the pupil, r_e is the right edge of the pupil, t_e is the top edge of
%the pupil, and b_e is the bottom edge of the pupil.

L = left_edge_test(l, h, l_e, 0);

R = right_edge_test(r, h, r_e, 0);

T = top_edge_test(t, v, t_e, 0);

B = bottom_edge_test(b, v, b_e, 0);


I = i_stats;
I.MajorAxisLength = R – L;
I.MinorAxisLength = B – T;
I.Centroid = [floor((R – L)/2 + L) floor((B – T)/2 + T)];
```

```
function T = top_edge_test(t, v, t_e, old)

%This function takes the current location of the top of the iris, the
%vertical vector holding the thresholded iris data, the top edge of the
%iris, and the former value of the top edge. It compares the next
%potential iris edge to the current iris edge.  If the new value is within
%the default distance (N=20 pixels), the old value is replaced with the
%new. Otherwise, the old value is returned.  This function is used
%recursively.
%
%   usage: T = top_edge_test(t, v, t_e, old)
%
%where t is the current location of the top of the iris, v is the
%vertical vector holding the thresholded iris data, t_e is the top edge
%of the pupil, and old is the former value of the top edge used for
%comparison.

T = t; N = 20;

t_e_new = t;
t_bound = v(1 : (t_e_new));
[T_new_front, T_new_back] = middle_top_band3(t_bound);
if T_new_front < T – N | T_new_front == old
   T = t;
else
   T = T_new_back;
   T = top_edge_test(T, v, t_e, T);
end
```

Appendix B: Experimental Data

iris index:		10% Bound	40% Bound	70% Bound		Variation
1		0.9725	0.9371	0.9018		0.0354
2		0.9302	0.8776	0.8251		0.0526
3		0.9545	0.9309	0.9074		0.0236
4		0.7064	0.7033	0.7002		0.0031
5		0.9187	0.9046	0.8904		0.0142
6		0.9340	0.9326	0.9312		0.0014
7		0.9568	0.9033	0.8498		0.0535
8		0.8626	0.8170	0.7713		0.0457
9		0.9701	0.9123	0.8544		0.0579
10		0.9073	0.8649	0.8226		0.0424
11		0.9505	0.8676	0.7846		0.0830
12		0.8887	0.8295	0.7702		0.0593
13		0.8443	0.8434	0.8425		0.0009
14		0.8653	0.7283	0.5913		0.1370
15		0.9544	0.9123	0.8701		0.0422
16		0.9322	0.8913	0.8504		0.0409
17		0.8570	0.6818	0.5065		0.1753
18		0.8895	0.6577	0.4258		0.2319
19		0.9151	0.9124	0.9098		0.0026
20		0.5022	0.4977	0.4933		0.0044
21		0.9339	0.8940	0.8541		0.0399
22		0.9496	0.9349	0.9202		0.0147
23		0.9406	0.9099	0.8792		0.0307
24		0.8948	0.7730	0.6512		0.1218
25		0.8993	0.8709	0.8425		0.0284
26		FS	FS	FS		FS
27		FS	FS	FS		FS
28		0.7113	0.7074	0.7035		0.0039
29		0.6368	0.6180	0.5991		0.0189
30		0.9121	0.9017	0.8913		0.0104
31		0.7585	0.7515	0.7446		0.0069
32		0.8829	0.7362	0.5896		0.1467
33		0.9489	0.9454	0.9418		0.0036
34		0.9401	0.8943	0.8484		0.0459
35		0.9466	0.8384	0.7302		0.1082
36		0.9357	0.9201	0.9045		0.0156
37		0.9470	0.9078	0.8687		0.0392
38		0.7222	0.6440	0.5657		0.0783
39		0.6997	0.6667	0.6337		0.0330
40		0.8615	0.8138	0.7661		0.0477
41		0.9476	0.9391	0.9306		0.0085
42		0.6194	0.6181	0.6169		0.0012
43		0.8993	0.8374	0.7755		0.0619
44		0.9517	0.8610	0.7702		0.0908
45		0.8580	0.7164	0.5748		0.1416

46		0.6760	0.6427	0.6093		0.0334
47		0.9514	0.9395	0.9277		0.0119
48		0.9684	0.9589	0.9494		0.0095
49		0.7860	0.6244	0.4628		0.1616
50		0.8498	0.6964	0.5431		0.1534
51		0.6311	0.5598	0.4885		0.0713
52		0.9152	0.9145	0.9138		0.0007
53		0.9265	0.9245	0.9224		0.0021
54		FS	FS	FS		FS
55		0.9599	0.8817	0.8035		0.0782
56		0.9525	0.9161	0.8797		0.0364
57		0.9067	0.9054	0.9042		0.0012
58		0.9376	0.8793	0.8210		0.0583
59		0.9394	0.9243	0.9092		0.0151
60		0.8848	0.8838	0.8827		0.0011
61		0.7726	0.6691	0.5655		0.1036
62		0.8869	0.8814	0.8759		0.0055
63		0.9268	0.9228	0.9187		0.0041
64		0.9371	0.9361	0.9350		0.0011
65		0.9723	0.9343	0.8963		0.0380
66		0.8723	0.7351	0.5979		0.1372
67		0.9045	0.8202	0.7360		0.0843
68		0.5846	0.5830	0.5814		0.0016
69		0.9288	0.7928	0.6568		0.1360
70		0.9530	0.8597	0.7663		0.0934
71		0.9348	0.8592	0.7835		0.0757
72		0.7520	0.7439	0.7359		0.0081
73		0.9459	0.8772	0.8084		0.0688
74		0.7449	0.7217	0.6985		0.0232
75		0.9157	0.8392	0.7628		0.0765
76		0.9101	0.9046	0.8991		0.0055
77		0.9127	0.8407	0.7687		0.0720
78		0.7180	0.7082	0.6983		0.0098
79		0.9432	0.9038	0.8645		0.0394
80		FS	FS	FS		FS
81		0.8833	0.8770	0.8706		0.0063
82		0.6833	0.6753	0.6673		0.0080
83		0.9416	0.8558	0.7699		0.0859
84		0.9285	0.8883	0.8480		0.0403
85		0.8879	0.7553	0.6227		0.1326
86		0.9497	0.9391	0.9285		0.0106
87		0.9244	0.8561	0.7878		0.0683
88		0.7724	0.7524	0.7323		0.0201
89		0.7098	0.7068	0.7039		0.0030
90		0.6652	0.6386	0.6119		0.0267
91		0.9659	0.9543	0.9427		0.0116
92		0.9515	0.9438	0.9361		0.0077
93		0.9496	0.9420	0.9344		0.0076
94		0.9312	0.8485	0.7658		0.0827

95		0.9668	0.9605	0.9542		0.0063
96		0.9236	0.9187	0.9138		0.0049
97		0.9476	0.9416	0.9356		0.0060
98		FS	FS	FS		FS
99		0.4879	0.4446	0.4014		0.0433
100		0.8874	0.7193	0.5512		0.1681
101		0.9507	0.9225	0.8944		0.0282
102		0.4753	0.4251	0.3749		0.0502
103		0.5367	0.5325	0.5283		0.0042
104		0.9426	0.9022	0.8618		0.0404
105		FS	FS	FS		FS
106		0.9206	0.8978	0.8749		0.0229
107		0.9695	0.9187	0.8679		0.0508
108		0.9028	0.8720	0.8413		0.0308
109		0.8437	0.8380	0.8322		0.0057
110		0.9376	0.9368	0.9359		0.0009
111		0.9255	0.9245	0.9234		0.0011
112		0.9648	0.9392	0.9137		0.0256
113		0.9570	0.9167	0.8768		0.0401
114		0.7569	0.7210	0.6852		0.0359
115		0.9420	0.9351	0.9281		0.0069
116		0.9245	0.9197	0.9149		0.0048
117		0.9564	0.9516	0.9469		0.0048
118		0.8931	0.8853	0.8775		0.0078
119		0.8779	0.8735	0.8692		0.0044
120		0.9237	0.9154	0.9071		0.0083
121		0.5666	0.5344	0.5023		0.0322
122		0.9021	0.7805	0.6588		0.1217
123		0.9360	0.7795	0.6229		0.1566
124		0.9712	0.9296	0.8879		0.0417
125		0.9634	0.9232	0.8831		0.0402
126		0.9124	0.8619	0.8113		0.0506
127		0.8835	0.8689	0.8544		0.0146
128		0.9109	0.9093	0.9077		0.0016
129		0.9250	0.8678	0.8106		0.0572
Average		0.8719	0.8276	0.7832		0.0443

Table 1: Orthogonal iris segmentation quality bound data

angle index		10% Bound	40% Bound	70% Bound		Variation
1		0.7102	0.6487	0.5571		0.0766
2		0.7063	0.7061	0.7059		0.0002
3		0.7364	0.7359	0.7354		0.0005
4		0.7189	0.7184	0.7179		0.0005
5		0.7396	0.7389	0.7382		0.0007
6		0.6660	0.6653	0.6646		0.0007
7		0.7209	0.7201	0.7193		0.0008
8		0.8920	0.8912	0.8903		0.0009

9		0.6489	0.6479	0.6469		0.0010
10		0.7102	0.7091	0.7080		0.0011
11		0.6538	0.6519	0.6501		0.0019
12		0.5946	0.5927	0.5908		0.0019
13		0.8795	0.8776	0.8756		0.0019
14		0.8895	0.8875	0.8854		0.0021
15		0.6368	0.6347	0.6326		0.0021
16		0.8873	0.8851	0.8829		0.0022
17		0.6402	0.6377	0.6353		0.0025
18		0.8806	0.8780	0.8754		0.0026
19		0.8642	0.8614	0.8586		0.0028
20		0.8445	0.8410	0.8375		0.0035
21		0.8940	0.8903	0.8866		0.0037
22		0.8517	0.8467	0.8417		0.0050
23		0.8002	0.7950	0.7898		0.0052
24		0.8754	0.8695	0.8635		0.0059
25		0.8796	0.8731	0.8666		0.0065
26		0.8210	0.8144	0.8078		0.0066
27		0.6288	0.6220	0.6152		0.0068
28		0.6543	0.6467	0.6391		0.0076
29		0.8695	0.8608	0.8522		0.0087
30		0.6937	0.6849	0.6760		0.0088
31		0.8709	0.8610	0.8510		0.0100
32		0.7137	0.7036	0.6936		0.0101
33		0.8727	0.8626	0.8525		0.0101
34		0.6211	0.6105	0.6000		0.0106
35		0.7586	0.7480	0.7375		0.0106
36		0.8901	0.8790	0.8679		0.0111
37		0.8481	0.8369	0.8257		0.0112
38		0.6415	0.6297	0.6179		0.0118
39		0.8594	0.8476	0.8357		0.0119
40		0.8195	0.8075	0.7954		0.0121
41		0.8325	0.8202	0.8078		0.0124
42		0.7948	0.7819	0.7691		0.0129
43		0.9176	0.9046	0.8916		0.0130
44		0.8117	0.7986	0.7854		0.0132
45		0.8002	0.7869	0.7736		0.0133
46		0.8230	0.8095	0.7961		0.0135
47		0.7345	0.7210	0.7075		0.0135
48		0.8368	0.8223	0.8078		0.0145
49		0.8890	0.8742	0.8594		0.0148
50		0.8260	0.8110	0.7959		0.0151
51		0.8887	0.8732	0.8577		0.0155
52		0.8244	0.8078	0.7913		0.0166
53		0.7917	0.7749	0.7581		0.0168
54		0.5566	0.5395	0.5223		0.0172
55		0.8029	0.7856	0.7683		0.0173
56		0.6539	0.6357	0.6175		0.0182
57		0.5515	0.5326	0.5138		0.0189

58		FS	FS	FS		FS
59		0.8547	0.8356	0.8164		0.0192
60		0.8511	0.8311	0.8112		0.0200
61		0.7571	0.7370	0.7168		0.0202
62		0.7694	0.7487	0.7281		0.0207
63		0.7001	0.6788	0.6576		0.0213
64		0.7999	0.7779	0.7560		0.0220
65		0.9680	0.9457	0.9235		0.0223
66		0.7568	0.7342	0.7116		0.0226
67		0.8501	0.8269	0.8038		0.0232
68		0.7518	0.7283	0.7047		0.0236
69		0.7722	0.7478	0.7235		0.0244
70		0.9048	0.8803	0.8558		0.0245
71		0.8157	0.7905	0.7652		0.0253
72		0.8313	0.8059	0.7805		0.0254
73		0.7595	0.7339	0.7084		0.0256
74		0.7611	0.7351	0.7091		0.0260
75		0.9327	0.9063	0.8798		0.0265
76		0.8424	0.8156	0.7889		0.0268
77		0.8225	0.7951	0.7678		0.0274
78		0.7631	0.7354	0.7076		0.0278
79		0.7354	0.7076	0.6799		0.0278
80		0.8718	0.8438	0.8159		0.0280
81		0.8765	0.8485	0.8205		0.0280
82		0.7407	0.7125	0.6843		0.0282
83		0.8609	0.8327	0.8044		0.0283
84		0.8953	0.8668	0.8383		0.0285
85		0.8697	0.8411	0.8126		0.0286
86		0.7479	0.7193	0.6907		0.0286
87		0.8179	0.7893	0.7606		0.0287
88		0.7607	0.7320	0.7033		0.0287
89		0.7677	0.7380	0.7084		0.0297
90		0.9467	0.9163	0.8859		0.0304
91		0.7586	0.7279	0.6971		0.0308
92		0.8835	0.8527	0.8218		0.0309
93		0.9606	0.9297	0.8987		0.0310
94		0.8424	0.8105	0.7786		0.0319
95		0.7684	0.7362	0.7041		0.0322
96		0.7657	0.7335	0.7012		0.0323
97		0.9214	0.8891	0.8568		0.0323
98		0.7695	0.7369	0.7044		0.0326
99		0.7175	0.6843	0.6511		0.0332
100		0.9283	0.8946	0.8609		0.0337
101		0.8803	0.8458	0.8112		0.0346
102		0.7258	0.6905	0.6552		0.0353
103		0.9340	0.8987	0.8634		0.0353
104		0.7080	0.6726	0.6373		0.0354
105		0.9436	0.9076	0.8716		0.0360
106		0.9657	0.9293	0.8928		0.0365

107		0.9250	0.8883	0.8516		0.0367
108		0.9517	0.9149	0.8782		0.0368
109		0.9511	0.9139	0.8767		0.0372
110		0.8888	0.8516	0.8143		0.0373
111		0.9400	0.9026	0.8654		0.0373
112		0.8914	0.8541	0.8168		0.0373
113		0.9513	0.9139	0.8765		0.0374
114		0.9214	0.8832	0.8449		0.0383
115		0.9564	0.9179	0.8793		0.0386
116		0.9528	0.9141	0.8754		0.0387
117		0.8757	0.8368	0.7980		0.0389
118		0.4878	0.4482	0.4085		0.0397
119		0.9774	0.9377	0.8980		0.0397
120		0.9661	0.9263	0.8865		0.0398
121		0.9402	0.8992	0.8582		0.0410
122		0.8979	0.8560	0.8140		0.0420
123		0.8479	0.8057	0.7636		0.0422
124		0.9402	0.8980	0.8558		0.0422
125		0.8795	0.8371	0.7947		0.0424
126		0.8736	0.8308	0.7880		0.0428
127		0.9309	0.8879	0.8449		0.0430
128		0.9681	0.9248	0.8815		0.0433
129		0.8620	0.8186	0.7752		0.0434
130		0.9575	0.9136	0.8696		0.0440
131		0.9612	0.9171	0.8730		0.0441
132		0.8316	0.7873	0.7430		0.0443
133		0.8923	0.8478	0.8033		0.0445
134		0.9081	0.8634	0.8187		0.0447
135		0.9407	0.8957	0.8506		0.0451
136		0.9040	0.8588	0.8137		0.0452
137		0.9647	0.9191	0.8736		0.0456
138		0.6637	0.6180	0.5723		0.0457
139		0.9333	0.8871	0.8409		0.0462
140		0.8812	0.8348	0.7885		0.0464
141		0.6286	0.5817	0.5349		0.0469
142		0.4974	0.4501	0.4028		0.0473
143		0.7126	0.6650	0.6175		0.0476
144		0.9406	0.8924	0.8443		0.0482
145		0.4872	0.4390	0.3909		0.0482
146		FS	FS	FS		FS
147		0.9024	0.8538	0.8051		0.0487
148		0.7369	0.6875	0.6381		0.0494
149		0.6720	0.6225	0.5731		0.0495
150		0.7294	0.6798	0.6302		0.0496
151		0.7327	0.6828	0.6330		0.0499
152		0.8780	0.8276	0.7772		0.0504
153		0.7105	0.6601	0.6097		0.0504
154		0.6770	0.6266	0.5762		0.0504
155		0.4834	0.4329	0.3824		0.0505

156		0.9427	0.8908	0.8389		0.0519
157		0.6930	0.6411	0.5892		0.0519
158		0.9390	0.8866	0.8342		0.0524
159		0.7433	0.6908	0.6383		0.0525
160		0.7264	0.6728	0.6192		0.0536
161		0.6926	0.6387	0.5849		0.0539
162		0.9412	0.8872	0.8333		0.0540
163		0.4892	0.4348	0.3804		0.0544
164		FS	FS	FS		FS
165		0.7241	0.6693	0.6144		0.0549
166		0.6812	0.6263	0.5713		0.0550
167		0.9446	0.8891	0.8337		0.0555
168		0.9451	0.8891	0.8332		0.0560
169		0.9395	0.8834	0.8274		0.0561
170		0.8981	0.8411	0.7841		0.0570
171		0.8165	0.7578	0.6991		0.0587
172		0.9043	0.8900	0.7856		0.0594
173		0.8724	0.8129	0.7535		0.0595
174		0.8729	0.8121	0.7513		0.0608
175		0.9487	0.8878	0.8269		0.0609
176		0.8905	0.8295	0.7684		0.0611
177		0.6589	0.5973	0.5357		0.0616
178		0.8976	0.8353	0.7730		0.0623
179		0.9012	0.8384	0.7757		0.0628
180		0.9494	0.8857	0.8220		0.0637
181		0.9232	0.8591	0.7950		0.0641
182		0.9381	0.8739	0.8097		0.0642
183		0.9172	0.8516	0.7860		0.0656
184		0.8582	0.7923	0.7264		0.0659
185		0.6556	0.5897	0.5238		0.0659
186		0.8807	0.8148	0.7489		0.0659
187		0.9415	0.8759	0.8096		0.0660
188		0.7234	0.6571	0.5907		0.0664
189		0.6590	0.5922	0.5253		0.0669
190		0.9140	0.8471	0.7803		0.0669
191		0.6493	0.5824	0.5154		0.0670
192		0.6700	0.6023	0.5345		0.0678
193		0.8267	0.7589	0.6911		0.0678
194		0.9448	0.8769	0.8091		0.0679
195		0.9051	0.8370	0.7689		0.0681
196		0.6820	0.6139	0.5457		0.0682
197		0.8684	0.7999	0.7314		0.0685
198		0.6533	0.5845	0.5158		0.0688
199		0.9145	0.8455	0.7766		0.0690
200		0.9297	0.8607	0.7917		0.0690
201		0.9012	0.8318	0.7624		0.0694
202		0.9436	0.8740	0.8043		0.0697
203		0.6710	0.6011	0.5311		0.0700
204		FS	FS	FS		FS

205		0.8970	0.8263	0.7556		0.0707
206		0.8889	0.8174	0.7458		0.0716
207		0.8762	0.8046	0.7329		0.0717
208		0.8986	0.8257	0.7528		0.0729
209		0.6745	0.6010	0.5275		0.0735
210		0.8940	0.8204	0.7468		0.0736
211		0.8631	0.7883	0.7135		0.0748
212		0.8593	0.7844	0.7094		0.0750
213		0.8394	0.7643	0.6893		0.0751
214		0.6735	0.5981	0.5226		0.0755
215		0.8860	0.8104	0.7348		0.0756
216		0.8407	0.7624	0.6841		0.0783
217		0.8417	0.7632	0.6847		0.0785
218		0.8605	0.7817	0.7029		0.0788
219		0.6498	0.5686	0.4875		0.0812
220		0.6574	0.5761	0.4949		0.0813
221		0.8671	0.7855	0.7039		0.0816
222		0.9425	0.8603	0.7781		0.0822
223		0.9386	0.8563	0.7740		0.0823
224		0.8883	0.8049	0.7216		0.0834
225		0.8789	0.7944	0.7098		0.0846
226		0.8951	0.8095	0.7238		0.0857
227		0.9573	0.8708	0.7844		0.0865
228		0.8695	0.7789	0.6884		0.0906
229		0.8527	0.7621	0.6715		0.0906
230		0.6956	0.6040	0.5124		0.0916
231		0.9201	0.8206	0.7211		0.0995
232		0.9095	0.8044	0.6993		0.1051
233		0.8678	0.7472	0.6265		0.1207
234		0.8933	0.7698	0.6462		0.1236
235		0.8732	0.7494	0.6257		0.1238
236		0.8355	0.6634	0.4913		0.1721
Average		0.8208	0.7792	0.7383		0.0412

Table 2: Non-orthogonal iris segmentation quality bound data

CONSENT and INFORMATION FORM

Biometric Dataset Collection

Introduction: I, _____, have been invited to participate in this research study which has been explained to me and is being conducted in the Electrical Engineering Department at the United States Naval Academy.

Purposes of the Study: The purpose of this study is to research techniques in biometric signal processing.

Description of Procedures: This study involves capturing my fingerprints, iris patterns, voice, and 2D and 3D facial image, using several biometric scanners. Any of the above mentioned biometrics may or may not be captured depending on the availability of devices and personnel.

Risks and Discomforts: I should not experience any inconvenience or discomfort beyond that typically associated with my picture being taken, finger or hand put on a surface, or reading audibly. There are no known or expected risks from participating in this study.

Benefits: I understand that this study is not expected to be of direct benefit to me, but the knowledge gained may be of benefit to others.

Financial Considerations: I will incur no costs for participating in this study, and I will not be paid to participate in this study.

Confidentiality: I understand that any information about me obtained as a result of my participation in this research will be kept as confidential as legally possible. In any publication that results from this research my name, my facial images, or any information from which I might be obviously identified will not be published unless my consent is obtained.

Voluntary Participation: Participation in this study is voluntary. I have been given the opportunity to ask questions about the research, and I have received answers concerning areas I did not understand.

You are making a decision whether or not to participate. Your signature indicates that you have read and understand the information presented above, that you have decided to participate, and that you consent to the study as described.

Participant Signature: _____

Participant Name (printed): _____Date: _____

Investigator Name (printed): _____

Investigator Signature: _____

IRIS PATTERN EXTRACTION USING BIT PLANES AND STANDARD DEVIATIONS

Bradford Bonney, Robert Ives, Delores Etter, Yingzi Du
Electrical Engineering Department
U.S. Naval Academy
Annapolis, MD 21402

ABSTRACT

Iris recognition has been shown to be very accurate for human identification. In this paper, we develop a technique for iris pattern extraction utilizing the least significant bit-plane: the least significant bit of every pixel in the image. Through binary morphology applied to the bit-plane, the pupillay boundary of the iris is determined. The limbic boundary is identified by evaluating the standard deviation of the image intensity along the vertical and horizontal axes. Because our extraction approach restricts localization techniques to evaluating only bit-planes and standard deviations, iris pattern extraction is not dependent on circular edge detection. This allows for an expanded functionality of iris identification technology by no longer requiring a frontal view, which leads to the potential for off-angle iris recognition technology. Initial results show that it is possible to fit a close elliptical approximation to an iris pattern by using only bit-planes and standard deviations for iris localization.

1. INTRODUCTION

The iris is the round, pigmented tissue that lies behind the cornea [1]. The patterns within the iris are very unique to each person, and even the left eye is unique from the right eye [2]. Compared with other biometric features such as face and fingerprint, iris patterns are more stable and reliable [3, 4].

Since Ophthalmologists Flom and Safir first noted the uniqueness of the iris patterns in 1987 [5], various algorithms have been proposed for iris recognition [1, 6-12], which include the quadrature 2D Gabor wavelet method [1], the Laplacian parameter approach [8], zero-crossings of the one-dimensional (1D) wavelet [9], the independent component analysis (ICA) approach [10], Gabor filtering and wavelet transform [11], and the texture analysis using multi-channel [12]. Recently, Du *et al.* designed a local texture analysis algorithm to calculate the local variances of iris images and generate a one-dimensional iris signature [6, 7], which relaxed the requirement of entire whole iris for identification and recognition [7]. However, all of these algorithms assume that a circular iris pattern has been successfully extracted from an image

In practice, the iris pattern must be extracted from the image prior to analysis. Currently, iris recognition systems require a cooperative subject [3]. Both commercial systems that utilize Daugman's algorithm [13] and other separately developed iris recognition techniques like the one-dimensional approach developed by Du *et al.* [6, 7] rely on this supposition to detect the iris pattern using circular edge detection. As an iris image is rotated away from the normal to the imaging device, these systems develop complications in determining the iris pattern. They are unable to successfully locate the iris pattern in order to proceed to recognition and matching. Our initial results show that it is possible to fit a close elliptical approximation to both orthogonal and non-orthogonal iris patterns in order to extract them from digital images, which can potentially be adapted for non-orthogonal iris recognition.

2. MORPHOLOGICAL OPERATIONS

Erosion and dilation are two morphological operations that are very useful in processing binary images. Erosion and dilation, as depicted in Fig 1 below,

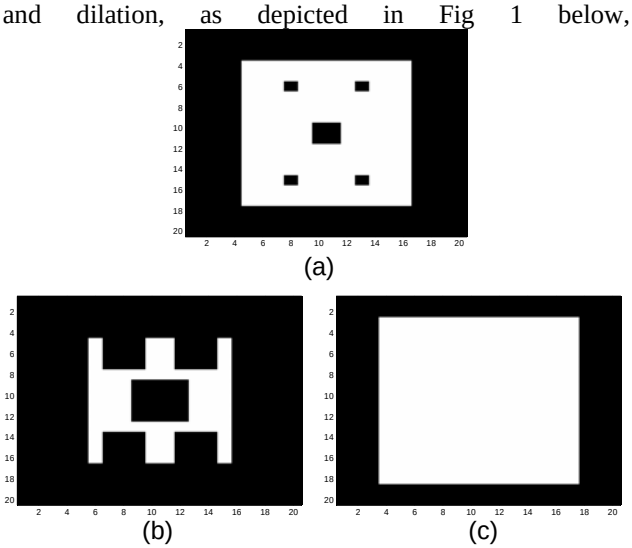


Figure 1: (a) Original binary image **(b)** Image (a) eroded by a 3x3 structuring element **(c)** Image (a) dilated by a 3x3 structuring element

allow groupings of *ones*, represented by white pixels, to be enlarged or shrunk to produce resulting images that either fill grouping gaps or remove small groupings of *ones* as necessary.

Erosion can best be described as a mathematical operator that shrinks groupings of *ones* in binary images. A structuring element, or kernel, represented by a matrix of *ones* and *zeros*, is passed through the original image. During erosion, the resultant image is the set of all structuring element origin locations where the translated structuring element has no overlap with the background of the original image [14]. Ultimately, small groupings of *ones* and thin lines are removed from the image.

On the other hand, dilation is the mathematical operator that expands groupings of *ones*, filling 'gaps' or 'holes' in a binary image. The dilation of an image by a structuring element results in an image consisting of all the structuring element origin locations where the reflected and translated structuring element overlaps at least some portion of the image [14]. By repeatedly dilating a binary image, black space (*zeros)* within groupings of *ones* is removed in order to achieve a single homogenous region.

3. PUPILLARY BOUNDARY DETECTION

To find the pupillary boundary, an original digital iris image is first adjusted and the least significant bit-plane is removed for analysis. The first step towards achieving a homogenous region that can be identified as the pupil is to set the image values below 60 and above 240 equal to 255. This sets the pupil to binary '1' in all bit-planes as seen in Fig. 2.

The purpose for adjusting values above 240 to 255 is to reduce the effect of specularities that may be present in the pupil.

Bit-plane 0, the least significant bit-plane, is used to determine the pupillary boundary because it not only provides a relatively homogenous region that is easily identifiable as the pupil, but also because it is fast and easy to extract from the original image. After performing modulo division by two on the image, the resultant is the least significant bit-plane. The borders of the bit-plane are removed to minimize the effects of near-infrared glare. Figs. 3 shows a series of morphological operations using 3x3 and 7x7 kernels to erode and dilate the binary image until a single homogeneous region remains.

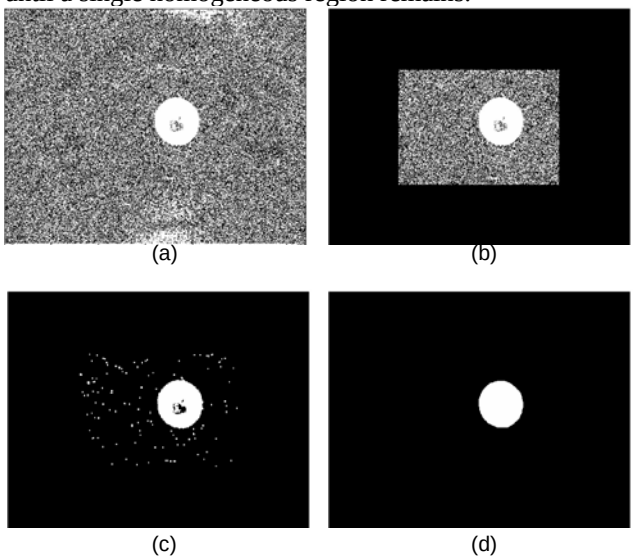


Figure 3: (a) Least significant bit-plane **(b)** Bit-plane zero with borders removed **(c)** Initial erosion and dilation using 3x3 kernel **(d)** Final homogenous pupil mask

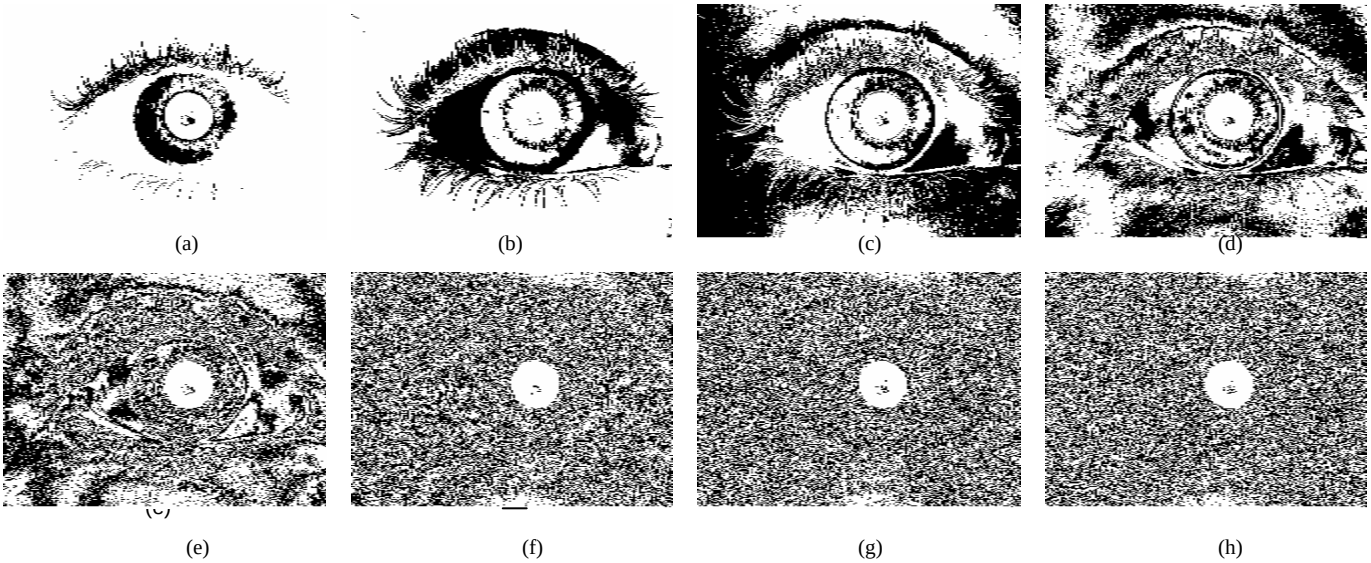


Figure 2: **(a)** Most significant bit-plane 7 **(b)** Bit-plane 6 **(c)** Bit-plane 5 **(d)** Bit-plane 4 **(e)** Bit-plane 3 **(f)** Bit-plane 2 **(g)** Bit-plane 1 **(h)** Least significant bit-plane 0

Once the final homogenous pupil mask is found through binary morphology, the end points in the cardinal directions – N, E, S, W – are identified. Equation (1) is used to determine the elliptical curve,

$$\frac{x^2}{b^2}+\frac{y^2}{a^2}=1, \qquad (1)$$

where b and a are the major and minor axes, and x and y are the row and column coordinates of the ellipse. The elliptical curve is fit to these points, with each half of the ellipse being calculated independently of the other as seen in Fig 4.

Figure 4: (a) N,S,E,W points of pupillary boundary **(b)** Elliptical curve fit through cardinal points defining pupillary boundary

4. LIMBIC BOUNDARY DETECTION

Once the pupillary boundary has been successfully determined, the limbic boundary must then be isolated. To find the limbic boundary, the division between the iris and the sclera [15], we use standard deviation windows to calculate local standard deviations in the vertical and horizontal directions. The resulting standard deviation windows are thresholded in order to produce a binary image. By eroding and dilating these standard deviation windows, a single row or column vector is obtained as seen in Fig 5. These vectors are then used to determine the location of the limbic boundary.

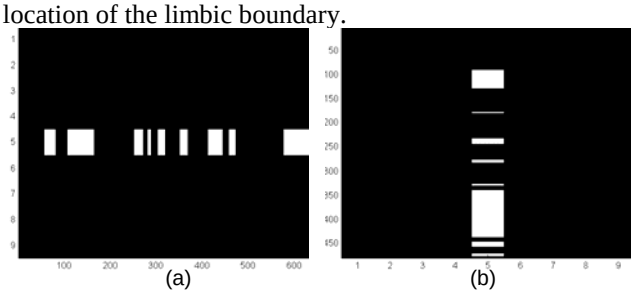


Figure 5: (a) horizontal local standard deviation vector **(b)** vertical local standard deviation vector

In order to determine the limbic boundaries from the horizontal and vertical standard deviation vectors, the pupillary boundary coordinates in the cardinal directions must be known. These coordinates in each of their respective vectors – E, W in the horizontal direction and N, S in the vertical direction – provide a starting location for the search of the limbic boundary because the limbic boundary may not be located on the interior of the pupillary boundary.

To automatically isolate the limbic boundary using standard deviation windows, a continual guess-and-check method is used. The pupil boundary in one of the cardinal directions is used as an initial starting location. For example, to find the left most limbic boundary, starting at the left most pupillary boundary and moving in a direction away from the pupil, we scan left within the standard deviation band until we reach a grouping of *ones*. This first point is marked as a potential limbic boundary. In this example, the scan continues left until the next grouping of *ones* is located. If the next indication of a potential limbic boundary is reasonable, that location is marked as the new limbic boundary potential. A reasonable limbic boundary marker is one that does not appear after a fifty pixel duration of *zeros*. These *zeros* represent the either the uniformity of the sclera, the white part of the eye, or the constant skin tones above and below the eyelashes. Fig 6 demonstrates the automatic limbic boundary determination and testing features.

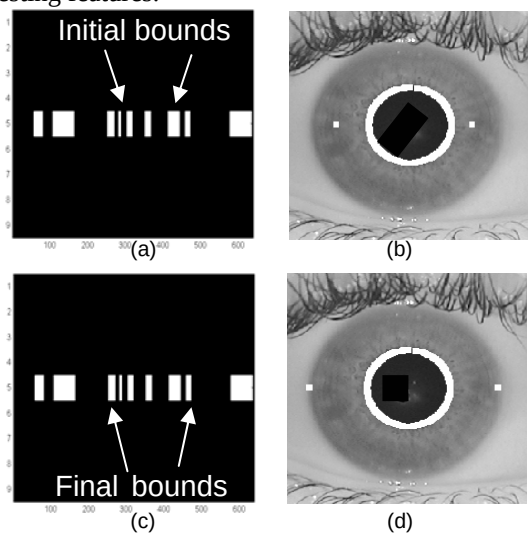


Figure 6: (a) Horizontal local standard deviation vector with initial boundaries highlighted **(b)** Eye image with initial horizontal boundaries indicated **(c)** Vertical local standard deviation vector with final limbic boundaries highlighted **(d)** Eye image with final horizontal boundaries indicated

Once both the horizontal and vertical limbic boundaries have been determined, Eqn 1 is used to calculate the elliptical curve of each half of the iris. Again, each half of the ellipse is calculated independently of the other, and the resulting area between the pupillary boundary and the limbic boundary forms the iris mask, as seen in Fig 7.

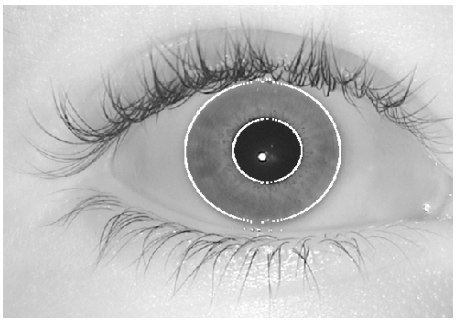


Figure 7: Eye image with final pupillary and limbic boundaries shown

5. EXPERIMENTAL RESULTS

The Institute of Automation, Chinese Academy of Science, CASIA, [16] and the United States Naval Academy, USNA, biometrics laboratory databases were used for algorithm testing. Performance of the bit-plane and standard deviation window process was divided into three categories based on the amount of iris pattern within the mask, as well as fit of the elliptical boundaries. Fig 8 shows examples of each category and an approximate fit used for categorizing output. Because of the difficulties in determining the limbic boundary, three distinct output categories emerged. Category 1 images, Figs 8(a) and 8(b),

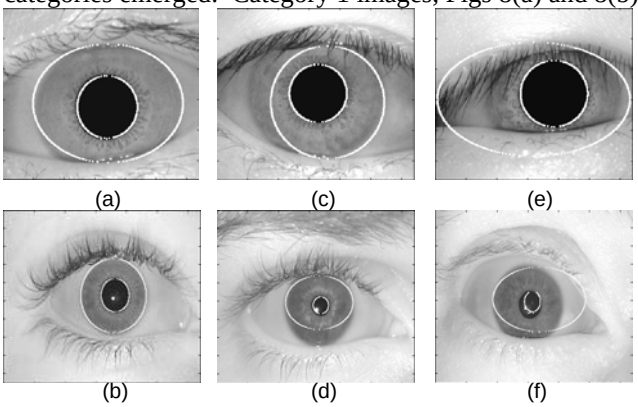


Figure 8: (a) and **(b)** Category 1 iris localization **(c)** and **(d)** Category 2 iris localization **(e)** and **(f)** Category 3 iris localization

are defined by output that correctly identifies and extracts the iris pattern. Category 2 images, Figs 8(c) and 8(d), are those images in which the limbic boundary does not encompass the full iris pattern. This is caused primarily by a lack of distinction between the limbic boundary and the sclera. In the standard deviation windows, no significant variation in pixel intensity occurred to alert the presence of the boundary. Category 3 images, Figs 8(e) and 8(f), are the remaining images. Output from these categories typically traversed the image far beyond the limbic boundary. Causes for the inability to identify these images can be attributed to large occlusions of the eyelids or eyelashes, or intense glare from the near infrared illumination of the surrounding skin. Table 1 shows results for iris extraction of the CASIA and USNA databases.

	Category 1 (Excellent)	Category 2 (Fair)	Category 3 (Poor)
CASIA	63	29	16
USNA	78	18	8

Table 1: *Experimental results of bit-plane and standard deviation approach to iris pattern extraction.*

For initial CASIA testing, the second of seven images was used for all 108 subjects. When testing the USNA images, one image from 104 different eyes was used. In all cases for the CASIA database images and in all but two cases for the USNA database images, the pupil was successfully isolated. Limbic boundary detection was the distinguishing factor in determining performance of this method of iris extraction.

Difficulties arose due to excessive eyelash interference at the limbic boundaries, causing large groupings of *ones* in the vertical standard deviation windows. These large groupings can be accredited to the static threshold that was used in calculating the standard deviation windows, a threshold that was determined experimentally to yield the best overall results. Experimentally, the best threshold to use varies over a forty integer value range. The use of a dynamic threshold based on individual image characteristics will most likely improve the results of the vertical and horizontal standard deviation windows. To initially compensate for these groupings of *ones* that fall within the potential area for limbic boundaries, if a grouping of fifty or more *ones* is encountered in the vertical direction, the center point of the grouping is selected as the limbic boundary.

The response of successful iris pattern extraction from the USNA database images was substantially better than the CASIA database. Reasons for the discrepancy in performance include unknown preprocessing to CASIA database images, and less extreme iris pattern occlusion due to eyelids and eyelashes in the USNA database. Also, the resolution on the CASIA database images is less than the USNA database images. The pixel vales appear to vary less between the limbic boundary and the sclera in the CASIA database images. Improving the limbic boundary detection modules to account for the increased shadowing with a dynamic threshold should improve overall system performance.

6. CONCLUSIONS

All of the statistical results found in Table 1 were conducted with iris images in which the user was looking directly at the imaging device. The purpose for

approaching the issue of iris pattern extraction without the assumption that patterns are circular is to allow for the iris extraction of non-orthogonally captured iris images. Initial tests show that with some additional modifications to the limbic boundary detection, this approach is applicable to non-orthogonal iris images as seen in Fig 9.

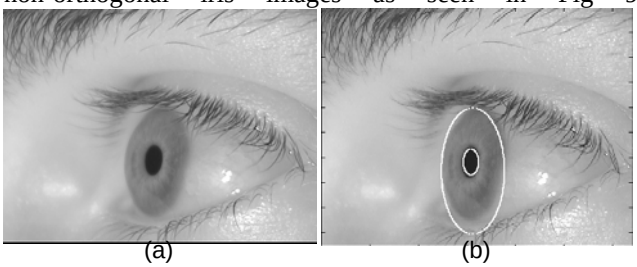


Figure 9: (a) Non-orthogonal image **(b)** Resulting elliptical iris mask

Because the iris image in Fig 9 is rotated away from the normal to the imaging device, current commercial systems develop complications extracting the iris pattern. Initial results show that it is possible to fit a close elliptical approximation to non-orthogonal iris patterns using bit-planes and standard deviation windows, which can potentially be adapted for off-angle iris recognition.

7. ACKNOWLEDGEMENT

Portions of the research in this paper use the CASIA iris image database [16] collected by Institute of Automation, Chinese Academy of Sciences. The authors would like to thank LT Robert Schultz, USN, from the U.S. Naval Academy for use of his MATLAB GUI for iris image collection.

8. REFERENCES

[1] J. Forrester, A. Dick, P. McMenamin, and W. Lee, *The Eye: Basic Sciences in Practice*, W B Saunder, London, 2001.

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

[3] Y. Du, R. W. Ives, and D. M. Etter, "Iris Recognition", a chapter on biometrics, *the Electrical Engineering Handbook*, 3rd Edition, Boca Raton, FL: CRC Press, 2004 (in press).

[4] J.D.Woodward, N.M.Orlans, and P.T.Higgins, *Biometrics*, the McGraw-Hill Company, California, U.S.A, 2002.

[5] L. Flom and A. Safir, United States Patent No. 4,641,349 (issued February 3, 1987), *Iris Recognition System*, Washington D.C.: U.S. Government Printing Office.

[6] Y. Du, R. W. Ives, D. M. Etter, T. B. Welch, and C.-I Chang, "One Dimensional Approach to Iris Recognition", *Proceedings of SPIE Volume 5404*, pp. 237-247, Apr., 2004.

[7] Y. Du, R. W. Ives, D. M. Etter, and T. B. Welch, "Use of One-Dimensional Iris Signatures to Rank Iris Pattern Similarities", submitted to *Optical Engineering*, 2004.

[8] R.P. Wildes, J.C. Asmuth, G.L. Green, S.C. Hsu, R.J. Kolczynski, J.R. Matey, and S.E. McBride, "A Machine Vision System for Iris Recognition", *Mach. Vision Application*, Vol. 9, 1-8, 1996.

[9] W.W. Boles and B. Boashash, "A Human Identification Technique Using Images of the Iris and Wavelet Transform", *Signal Processing, IEEE Transactions on*, Vol. 46, No. 4, 1998.

[10] Y.-P. Huang, S.-W. Luo, E.-Y. Chen, "An Efficient Iris Recognition System", *Proceedings of the First International Conference on Machine Learning and Cybernetics*, pp. 450-454, 2002.

[11] Y. Zhu, T. Tan, and Y. Wnag, "Biometric Personal Identification Based on Iris Patterns", *Pattern Recognition, 15th International Conference on*, Vol. 2 , pp. 801 –804, 2000.

[12] L. Ma, Y. Wang, and T. Tan, "Iris Recognition Using Circular Symmetric Filters", *Pattern Recognition, 16th International Conference on* , Vol. 2 , pp. 414-417, 2002 .

[13] Daugman, John G. "Biometric Personal Identification System Based on Iris Analysis", U.S. Patent 5,291,560. 1994.

[14] Gonzalez, Rafael C., R. E. Woods, and Steven L. Eddins. "Digital Image Processing using MATLAB." Prentice Hall, Upper Saddle River, NJ. 20044.

[15] A. J. Bron, R. C. Tripathi, and B. J. Tripathi. "Wolff's Anatomy of the Eye and Orbit, 8th Edition." Chapman and Hall Medical, London. 1997.

[16] CASIA Iris Image Database, http://www.sinobiometrics.com

ANALYSIS OF PARTIAL IRIS RECOGNITION USING A 1-D APPROACH

Yingzi Du, Bradford Bonney, Robert Ives, Delores Etter, Robert Schultz

Electrical Engineering Department
U.S. Naval Academy
Annapolis, MD 21402

ABSTRACT

Iris recognition has been shown to be very accurate for human identification. In this paper, we investigate the performance of the use of a partial iris for recognition. A partial iris identification system based on a one-dimensional approach to iris identification is developed. The experiment results shows that a more distinguishable and individually unique signal is found in the inner rings of the iris. The results also show that it is possible to use only a portion of the iris for human identification.

1. INTRODUCTION

The iris is the round, pigmented tissue that lies behind the cornea [1]. The patterns within the iris are very unique to each person, and even the left eye is unique from the right eye [2]. Compared with other biometric features such as face and fingerprint, iris patterns are more stable and reliable [11,12].

Since ophthalmologists Flom and Safir first noted the uniqueness of the iris patterns in 1987 [3], various algorithms have been proposed for iris recognition [1, 4-10], which include the quadrature 2D Gabor wavelet method [1], the Laplacian parameter approach [6], zero-crossings of the one-dimensional (1D) wavelet [7], the independent component analysis (ICA) approach [8], Gabor filtering and wavelet transform [9], and the texture analysis using multi-channel Gabor filtering and wavelet transform [10]. Recently, Du *et al.* designed a local texture analysis algorithm to calculate the local variances of iris images and generate a one-dimensional iris signature [4, 5], which relaxed the requirement of a significant portion of the iris for identification and recognition [5].

Currently, iris recognition systems require a cooperative subject [11]. Partial iris recognition algorithms would be very important in designing systems, where capturing the entire iris may not be feasible.

In this paper, we investigate the accuracy of using a partial iris for identification. In addition, we also investigate which portion of the iris has most distinguishable patterns. A partial iris identification system based on a one-dimensional approach to iris identification system is developed. The experimental results show that it is possible to use only a partial iris image for human identification.

2. ONE DIMENSIONAL APPROACH TO IRIS IDENTIFICATION

In this paper, the one dimensional approach proposed by Du *et al.* [4, 5] is used as a technique for recognition analysis. Here, we briefly introduce the one-dimensional approach.

Fig. 1 shows the one-dimensional iris identification system architecture, which includes the Preprocessing Module, the Mask Generation Module, the Local Texture Pattern (LTP) Module, the Iris Signature Generation Module, the Enrollment Module, the Iris Signature Database and the Iris Identification Module.

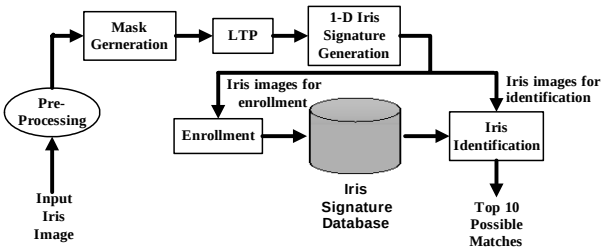


Figure 1. One-dimensional Iris Identification System Architecture [4, 5].

The Preprocessing Module finds the pupillary boundary, the limbic boundary, the eyelids, and the eyelashes in the input raw iris image. In addition, the iris image is transformed to the polar coordinates from rectangular coordinates in this step. At this point in the processing, an image is created which each row represents a concentric circle of iris pixels.

The Mask Generation Module isolates the iris pixels and normalizes the distance between the limbic boundary and the pupillary boundary to a constant $\tilde{L}$ pixels. Here

we select $\tilde{L}$ = 65. In this way, we achieve resolution invariance.

The LTP Module generates the local iris patterns by using overlapped windows to calculate the local variances.

The Iris Signature Generation Module builds a one-dimensional signature for each iris image by averaging the LTP values of each row. If more than 65% of the pixels in a row are non-iris, the signature value for that row is set to be 0.

The Enrollment Module averages multiple iris signatures generated from the same iris to create the one-dimensional iris signature template for later identification.

The Iris Signature Database collects the one-dimensional iris signatures and stores them in the database for further identification.

The Iris Identification Module matches the iris signature generated from a newly input iris image with the enrolled iris signatures inside the database. The matching score is based on the *Du* measurement [5]. The output of this module is the 10 closest matches from the database.

The merit of this one-dimensional LTP method is that it relaxes the requirement of using a major part of the iris, which can enable partial iris recognition. In addition, this approach generates a list of possible matches instead of only the best match. In this way, the users could potentially identify the iris image by another level of analysis.

3. PARTIAL IRIS IDENTIFICATION ANALYSIS

In our partial iris identification analysis, we used part of the iris image to generate the one-dimensional signatures. Fig. 2 shows the system architecture for generating the partial iris and the partial iris signature for iris identification, which includes the Partial Iris Generation Module, the Preprocessing and Mask Generation Module, the LTP Module, the 1-D Partial Iris Signature Generation Module, the Iris Signature Database Module, and the Iris Identification Module.

Fig. 3(a) is an example iris image. The Partial Iris Generation Module will select a portion of the iris based on a particular experiment. In our experiments, we analyzed three different kinds of partial iris images:

- Left-to-Right: The "Left-to-Right" model gradually exposes the iris beginning at the left limbic boundary and concluding at the right limbic boundary. (Fig. 3(b)).
- Outside-to-Inside: The "Outside-to-Inside" model starts at the outer limbic boundary and gradually exposes the iris pattern in concentric rings moving toward the pupil. (Fig. 3(c)).
- Inside-to-Outside: The "Inside-to-Outside" model gradually exposes concentric rings beginning at the pupillary boundary and concluding at the limbic boundary. (Fig. 3(d)).

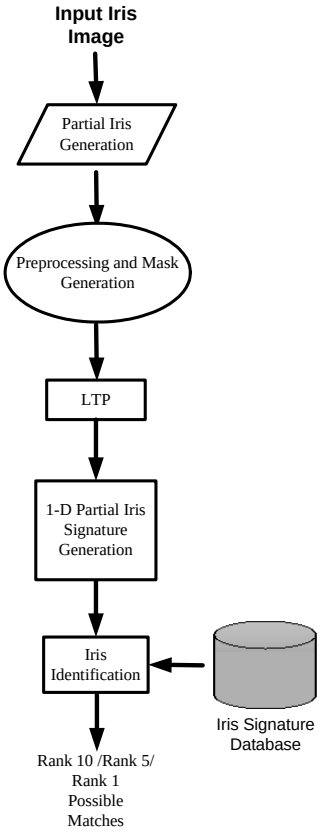


Figure. 2. Partial Iris Identification System

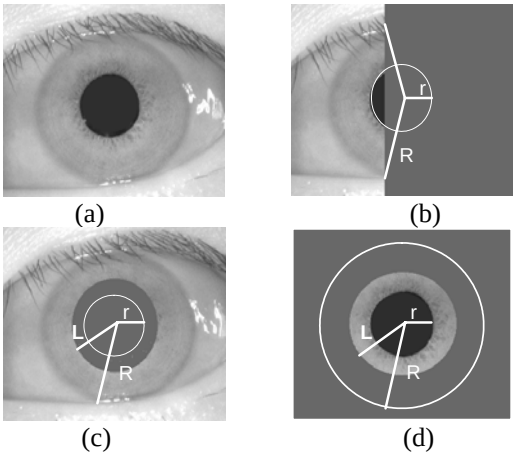


Figure 3. An example of generated partial iris images. (a) The original iris image, (b) Left-to-Right, (c) Outside-to-Inside, (d) Inside-to-Outside. (r, R, and L are pupil, limbic, and partial radius respectively.)

The percentage of the iris used in the identification is calculated differently for these three different approaches.

For Fig. 3(b), the percentage is calculated by: $\text{Partial percentage}=\frac{\text{Area of the Partial Iris}}{\text{Total Area of the Iris}}\times 100\%$; for Fig. 3(c), $\text{Partial percentage}=\frac{\text{R-L}}{\text{R-r}}\times 100\%$; for Fig. 3(d), $\text{Partial percentage}=\frac{\text{L-r}}{\text{R-r}}\times 100\%$ (r, R, and L are defining in Fig. 3).

Depending on the percentage of the iris image used, it may be hard to detect the pupil, the limbic boundary, the eyelids and eyelashes. The Preprocessing and Mask Generation Module will use the information retrieved from the entire iris image to generate the normalized mask for the partial iris image.

The LTP Module generates the local iris patterns by using overlapped windows to calculate the local variances and is similar to that of the 1D Iris Identification System in Fig. 1.

The 1-D Partial Iris Signature Generation Module builds a one-dimensional signature for each partial iris image by averaging the available LTP values of each row.

The Iris Identification Module matches the partial iris signature with the iris signatures inside the database, contained in the Database Module. Here, the iris signatures inside the database are also cut to match the length of the partial iris signature. The matching score is based on the *Du* measurement. The output of this module is the rank 10 closest matches from the database, the rank 5 closest matches, or the rank 1 closest match. (rank 10/rank 5/rank 1 means the matches following in top 10/ top 5/top 1 rank.)

4. EXPERIMENTAL RESULTS

In our experiment, the iris images from the CASIA iris image database [14] were used. It contains 756 iris images from 108 different eyes (each eye has 7 iris images). As stated in [5], 2 sets of iris images are not used because of insufficient iris patterns or unclear iris pattern. Overall, 742 iris images from 106 different eyes are used in the experiment.

In our experiment, the accuracy rate for partial iris recognition is defined as:

$$\text{Accuracy rate}=\frac{\text{Number of Correctly Identified Iris Images}}{\text{Total Number of Iris Images in the Test}}\times 100\% \quad (1)$$

Here "the correctly identified iris images" means the correctly identified iris images in the rank 10/5/1 ranks. The testing results coincide with intuition; as more of the iris pattern is available for analysis, the probability of correct match increases.

Fig. 4 shows the iris identification results for the "Left-to-Right" model. On the left side of the curve, the accuracy rate increases gradually and consistently between approximately 45% of iris pattern exposure. The curve is fairly flat until over 55% of iris pattern exposure, where the curve starts increasing again. The curve remaining steady approximately between 45%-55%, corresponds to regions covered by the eyelids and eyelashes. The reflection points of the curves are around 50%, the center of the iris. As the pattern is exposed, and the pupil is revealed, less distinguishable patterns are added to the image due to the relative area the pupil occupies as compared to the iris in the central vertical band of the eye. Once the pupil is fully exposed and more of the iris pattern is again added to the image, the percent chance that the test case matches the original increases, as expected. By slowly increasing the amount of iris area exposed in the "Left-to-Right" models, only a smaller relative area (of iris pattern) is exposed in the central vertical band (45%-55%), limiting its relative effectiveness in aiding in identification.

The performance of partial iris identification from the "Outside-to-Inside" Model is shown in Fig. 5, while the curves for the "Inside-to-Outside" model are shown in Fig. 6. In Fig. 5, the accuracy rate increases, and there is no obvious "knee in the curve". However, in Fig. 6, the accuracy rate increases much more dramatically than the other methods, and as a result, the "knee" for this model is located at approximately 35% of iris pattern exposure.

By setting a threshold for acceptance at a 95% accuracy rate (for rank 10 matching), the "Outside-to-Inside" model requires at least 65% of the iris pattern to be present. Conversely, only 40% on the iris pattern needs to be exposed for the "Inside-to-Outside" model to achieve the same accuracy rate. These experimental results support the conjecture that a more distinguishable and individually unique signal is found in the inner rings of the iris. As one traverses to the limbic boundary of the iris, the pattern becomes less defined, and ultimately less useful in determining identity. For the "Left-to-Right" model, 80% of the iris pattern would be necessary for a 95% accuracy rate for rank 10 matching. In the "Left-to-Right" model, each point of the one-dimensional signature is affected by the portion of the exposed iris, while for the "Inside-to-Outside" and "Outside-to-Inside" models, the points of the generated one-dimensional signature for the partial iris are either very similar to those of the original iris or zeroed out.

From Figs. 4-6, we can also see that using only 40% of the iris can achieve more than 85% accuracy rate for rank 10, 70% accuracy rate for Rank 5, and 45% for Rank 1. It shows that the partial iris recognition is promising for use in human identification using a rank 10/5 technique. However, it would be very challenging to use in human verification (rank 1).

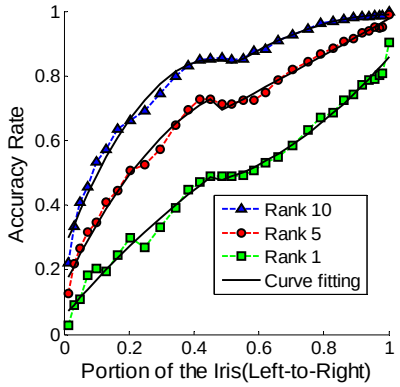


Figure 4. The performance of partial iris identification for "Left-to-Right" Model.

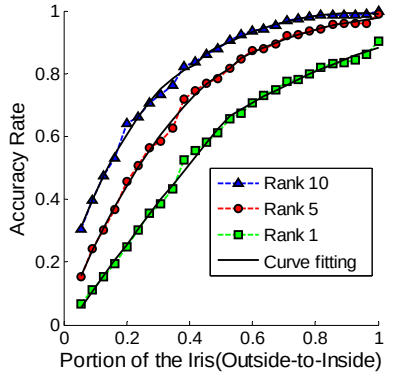


Figure 5. The performance of partial iris identification from the "Outside-to-Inside" Model

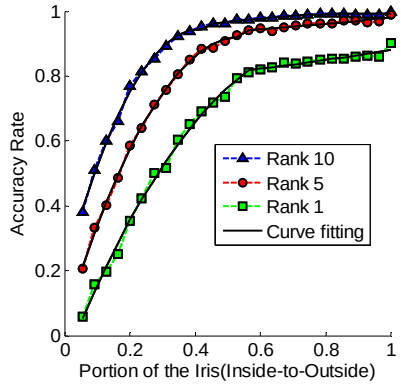


Figure 6. The performance of iris identification from the "Inside-to-Outside" Model

5. CONCLUSIONS

In this paper, the performance of partial iris identification is analyzed using the One-Dimensional LTP Approach. The experiment results show that a more distinguishable and individually unique signal is found in the inner rings of the iris. Also, as expected, the experimental results show that the eyelids and eyelashes detrimentally affect the iris recognition result. By slowly increasing the amount of iris area exposed in the "Left-to-Right" models, only a smaller relative area (of iris pattern) is exposed in the central vertical band, limiting its relative effectiveness in aiding in identification.

Finally, the experimental results show that a partial iris image can be used for human identification.

6. ACKNOWLEDGEMENT

Portions of the research in this paper use the CASIA iris image database [14] collected by Institute of Automation, Chinese Academy of Sciences. This work has been sponsored in part by the National Security Agency.

7. REFERENCES

[1] J. Forrester, A. Dick, P. McMenamin, and W. Lee, *The Eye: Basic Sciences in Practice*, W B Saunder, London, 2001.

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

[3] L. Flom and A. Safir, United States Patent No. 4,641,349 (issued February 3, 1987), *Iris Recognition System*, Washington D.C.: U.S. Government Printing Office.

[4] Y. Du, R. W. Ives, D. M. Etter, T. B. Welch, and C.-I Chang, "One Dimensional Approach to Iris Recognition", *Proceedings of SPIE Volume 5404*, pp. 237-247, Apr., 2004.

[5] Y. Du, R. W. Ives, D. M. Etter, and T. B. Welch, "Use of One-Dimensional Iris Signatures to Rank Iris Pattern Similarities", submitted to *Optical Engineering*, 2004.

[6] R.P. Wildes, J.C. Asmuth, G.L. Green, S.C. Hsu, R.J. Kolczynski, J.R. Matey, and S.E. McBride, "A Machine Vision System for Iris Recognition", *Mach. Vision Application*, Vol. 9, 1-8, 1996.

[7] W.W. Boles and B. Boashash, "A Human Identification Technique Using Images of the Iris and Wavelet Transform", *Signal Processing, IEEE Transactions on*, Vol. 46, No. 4, 1998.

[8] Y.-P. Huang, S.-W. Luo, E.-Y. Chen, "An Efficient Iris Recognition System", *the First International Conference on Machine Learning and Cybernetics*, pp. 450-454, 2002.

[9] Y. Zhu, T. Tan, and Y. Wnag, "Biometric Personal Identification Based on Iris Patterns", *Pattern Recognition, 15th International Conference on*, Vol. 2 , pp. 801 –804, 2000.

[10] L. Ma, Y. Wang, and T. Tan, "Iris Recognition Using Circular Symmetric Filters", *Pattern Recognition, 16th International Conference on* , Vol. 2 , pp. 414-417, 2002 .

[11] Y. Du, R. W. Ives, and D. M. Etter, "Iris Recognition", a chapter on biometrics, *the Electrical Engineering Handbook*, 3rd Edition, Boca Raton, FL: CRC Press, 2004 (in press).

[12] J.D.Woodward, N.M.Orlans, and P.T.Higgins, *Biometrics*, the McGraw-Hill Company, California, U.S.A, 2002.

[13]Y. Du, R. W. Ives, D. M. Etter, and T. B. Welch, "One-dimensional Iris Signature for Identification", U.S. Patent Pending, (Navy case number: 96,365), 2004.

[14] CASIA Iris Image Database, http://www.sinobiometrics.com

ANALYSIS OF PARTIAL IRIS RECOGNITION

Yingzi Du, Robert Ives, Bradford Bonney, Delores Etter
Electrical Engineering Department, U.S. Naval Academy, Annapolis, MD, USA 21402

ABSTRACT

In this paper, we investigate the accuracy of using a partial iris image for identification and determine which portion of the iris has the most distinguishable patterns. Moreover, we compare these results with the results of Du *et. al.* using the CASIA database. The experimental results show that it is challenging but feasible to use only a partial iris image for human identification.

Keywords: 1D iris identification, partial iris, iris recognition

1. INTRODUCTION

The iris (Fig. 1) is a protected internal organ behind the cornea which gives color to the eye [1]. Ophthalmologists Flom and Safir first noted that the iris is very unique for each person and remains unchanged after the first year of human life [2]. For each person, the left eye is distinctive from the right eye [2]. In 1987, they described a manual approach for iris recognition based on visible iris features. In 1994, Daugman invented the first automatic iris recognition system [3]. Since then, various algorithms have been proposed for iris recognition [3-11], which include Daugman's quadrature 2D Gabor wavelet method [3] and a one-dimensional iris recognition approach [4, 5, 11] by Du *et. al*.

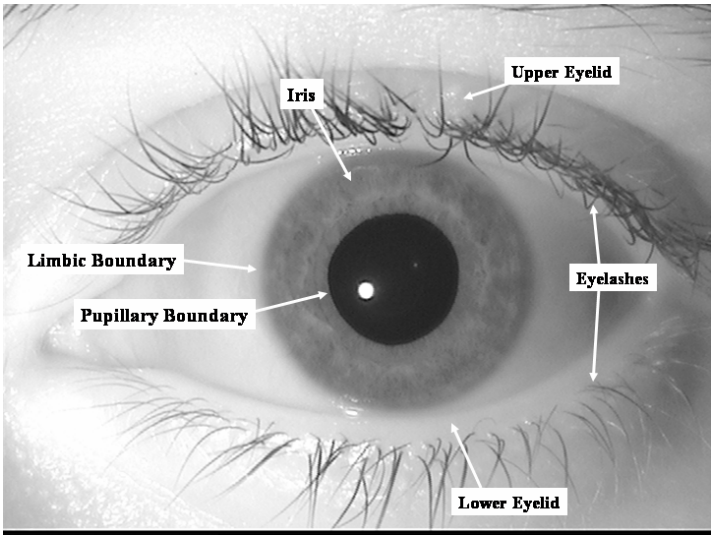


Figure 1: An iris image.

Currently, iris recognition systems require a cooperative subject [12]. Partial iris recognition algorithms would be very important in surveillance applications where capturing the entire iris may not be feasible. Little research has been performed in this area.

In this paper, we investigate the accuracy of using a partial iris image for identification and determine which portion of the iris has the most distinguishable patterns. Moreover, we compare these results against with the results of Du *et. al.* using the CASIA database [13] reported in [14]. The experimental results show that it is challenging but feasible to use only a partial iris image for human identification.

2. PARTIAL IRIS GENERATION

To analyze the partial iris recognition performance, we generated a collection of partial iris images from full iris images. For our experiments, we generated four different kinds of partial iris images. Fig, 2 provides an example, with Fig. 2(a) being the original full iris image. From this image, we created the following:

- Left-to-Right: The “Left-to-Right” model gradually exposes the iris beginning at the left limbic boundary and concluding at the right limbic boundary (Fig. 2(b)).
- Right-to-Left: The “Right-to-Left” model gradually exposes the iris beginning at the right limbic boundary and concluding at the left limbic boundary (Fig. 2(c)).
- Radial Outside-to-Inside: The “Radial Outside-to-Inside” model starts radially at the outer limbic boundary and gradually exposes the iris pattern in concentric rings moving toward the pupil (Fig. 2(d)).
- Radial Inside-to-Outside: The “Radial Inside-to-Outside” model gradually exposes concentric rings beginning radially at the pupillary boundary and concluding at the limbic boundary. (Fig. 2(e)).

The percentage of the iris patterns used in the identification is calculated by:

$$\text{Partial percentage} = \frac{\text{Area of the Partial Iris}}{\text{Total Area of the Iris}} \times 100\% \quad (1)$$

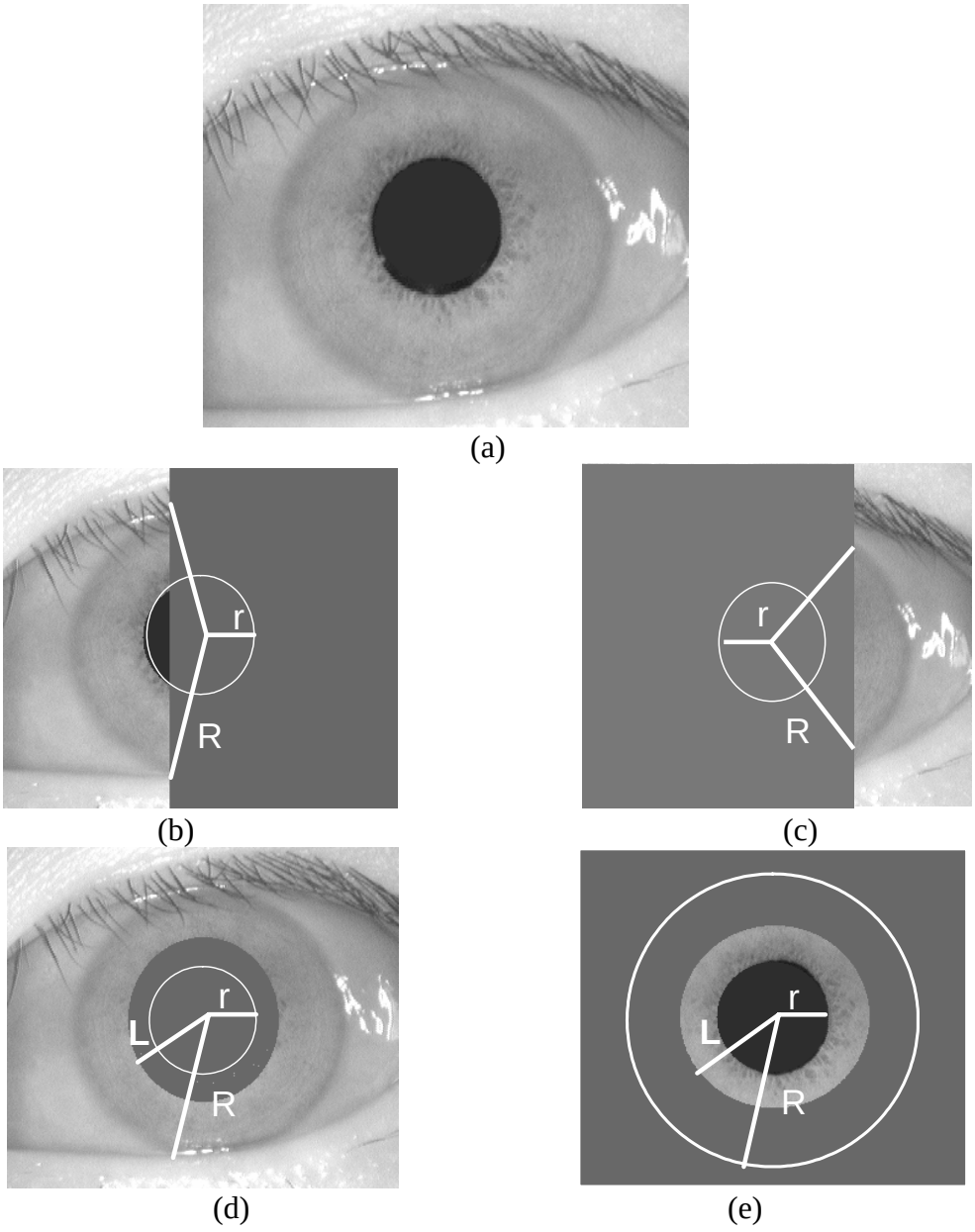


Figure 2. An example of generated partial iris images. (a) The original iris image, (b) Left-to-Right, (c) Right-to-Left, (d) Radial Outside-to-Inside, (e) Radial Inside-to-Outside. (r, R, and L are pupil, limbic, and partial radius respectively.)

With the partial iris images generated in Fig. 2, we can analyze four different kinds of situations:

- Tear Duct-to-Outside: The "Tear Duct-to-Outside" model gradually exposes the iris beginning at the near tear duct side and concluding at the far duct side. For the subject's left eye, This corresponds to the "Left-to-Right" model; for the subject's right eye, it would be the "Right-to-Left" Model.
- Outside-to-Tear Duct: The "Outside-to-Tear Duct" model moves in the inverse direction of the "Tear Duct-to-Outside" model.

- Radial Outside-to-Inside: Uses the “Outside-to-Inside” model for analysis.
- Radial Inside-to-Outside: Uses the “Radial Inside-to-Outside” model for analysis.

3. 1D IRIS IDENTIFICATION ALGORITHM

Fig. 3 shows the 1D Iris Identification System, which is used to analyze potential iris recognition. This algorithm is explained in detail in [4], and the functionality of the block diagram of Fig. 3 is briefly described in the following.

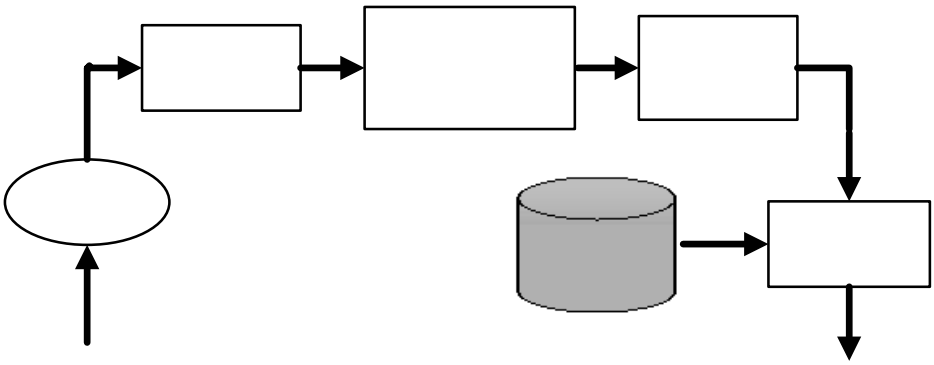


Figure 3. 1D Iris Identification System

The Preprocessing Module finds the pupillary boundary, the limbic boundary, the eyelids, and the eyelashes in the input raw iris image. The Mask Generation Module isolates the iris pixels and normalizes the distance between the limbic boundary and the pupillary boundary to a constant pixel size. The LTP Module generates the local iris patterns by using overlapped windows to calculate the local variances. The Iris Signature Generation Module builds a one-dimensional signature for each iris image by averaging the LTP values of each row. The Iris Signature Database stores the one-dimensional iris signatures in the database. The Iris Identification Module matches the iris signature generated from a newly input iris image with the enrolled iris signatures in the database. The matching score is based on the Du measurement [5]. The output of this module is the ten closest matches from the database.

The merit of this one-dimensional method is that it relaxes the requirement of using a major portion of the iris, which can enable partial iris recognition. In addition, this approach generates a list of possible matches instead of only the best match. In this way, the users could potentially identify the iris by another level of analysis.

The partial iris images are used to produce the iris pattern (signature). For a partial iris image, depending on the percentage of the iris image used it would be very difficult or even impossible to detect the pupil, the limbic boundary, the eyelids and eyelashes. The purpose of the paper is to analyze the partial iris identification performance. Therefore, in this system, we first preprocess the input raw full iris image to identify the iris area and determine pupil center, pupil radius, and limbic radius. In addition, eyelids and eyelashes are detected.

Ma
Gener

4. EXPERIMENTAL RESULTS

In our database, we have collected 1520 iris images from 106 different eyes. These iris images include those with contact lens and eyeglasses. In this analysis, we only use iris images from bare eyes (iris images without eyeglasses or contact lens). In addition, blurred iris images were eliminated from the experiment. Overall 818 iris images were used, 395 from left eyes and 423 from right eyes.

In this experiment, the accuracy rate for partial iris recognition is defined as:

$$\text{Accuracy rate} = \frac{\text{Number of Correctly Identified Iris Images}}{\text{Total Number of Iris Images Tested}} \times 100\% \tag{4}$$

Here "the correctly identified iris images" means the algorithm correctly placed the iris images within the top 10, or top 5, or top 1 (also called rank 10, rank 5 or rank 1). The testing results coincide with intuition; as more of the iris pattern is available for analysis, the probability of a correct match increases.

Fig. 6 shows the iris identification results for the "Tear Duct-to-Outside" model. Here, the Rank 10 and Rank 5 curves increase sharply until approximately 35% of iris pattern exposure, which is the reflection point of the curves. After this point, the two curves increase very slowly. However, the Rank 1 curves increases gradually and consistently throughout the exposing of the iris patterns. From Fig. 6, we find that exposure of 30% of the iris patterns is good enough to achieve over 95% accuracy for a Rank 10 system and over 90% accuracy for a Rank 5 system; while accurate identification (Rank 1) needs far more information.

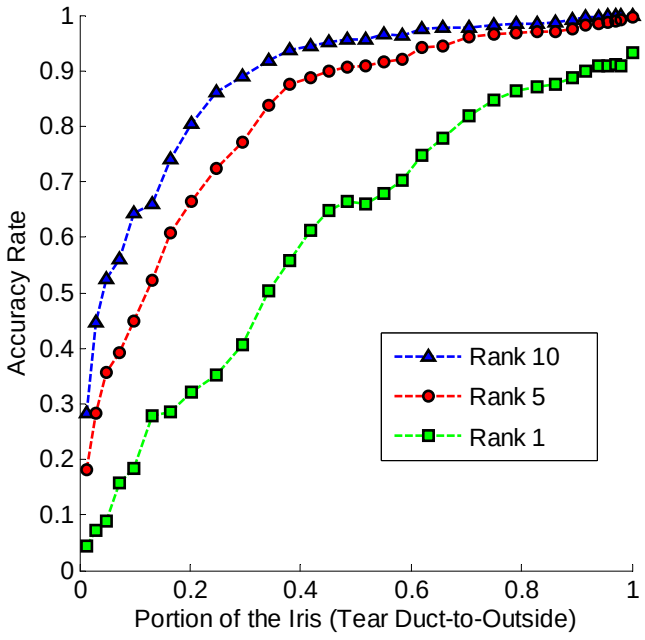


Figure 6. Partial iris identification performance for the "Tear Duct-to-Outside" model.

Fig. 7 shows the iris identification results for the "Outside-to-Tear Duct" model. In Fig. 7, the curves increase gradually and consistently until approximately 40% of iris pattern exposure. The curves remain fairly flat between approximately 40%-60%, correspondingly to regions covered by the eyelids and eyelashes. Once the pupil is fully exposed and more of the iris pattern is again added to the image, the accuracy again increases, as expected.

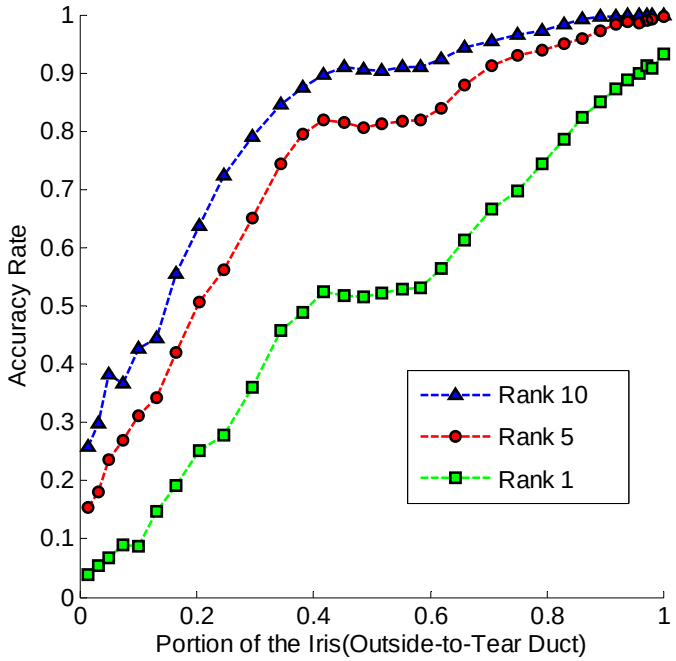
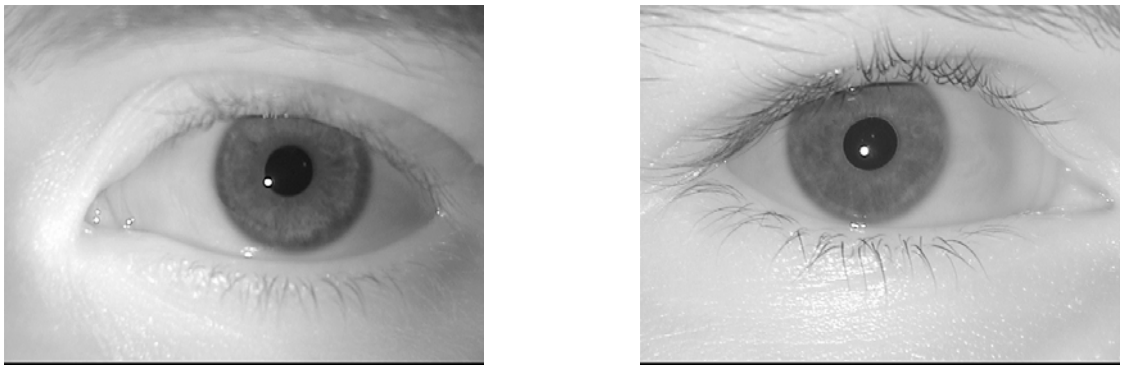


Figure 7. Partial iris identification performance for "Outside-to-Tear Duct" model.

Comparing Fig. 6 and Fig. 7, the Tear Duct-to-Outside model uses a smaller portion of the iris pattern to achieve the same accuracy rate as that of the "Outside-to-Tear Duct" model. For example, to achieve a 90% accuracy rate in the Rank 10 system, the "Tear Duct-to-Outside" model needs 25% while the Outside-to-Tear Duct model needs 45%. For 50% of iris pattern exposed, the "Tear Duct-to-Outside" model can achieve 70% identification (Rank 1) accuracy while the "Outside-to-Tear Duct" model can only achieve 50% accuracy.

The differences between these two models are reasonable and expected. They result from the shape of the eyelids. The eyelids tend to cover more of the Outside half than the Tear Duct side (Fig. 8). From the above analysis, we see that using these iris patterns to do partial identification is more challenging but feasible by using a Rank 10 or Rank 5 system.



(a) A left eye (b) A right eye

Figure 8. The shape of the eyelids

Because the iris images in the CASIA database do not label the left or right eye and it cannot always be visually determined (some eye images are clipped in the left and right side), we cannot compare the Tear Duct-to-Outside and Outside-to-Tear Duct models. Du *et al.* has used the "Left-to-Right" model to analyze the CASIA database [14]. The "Left-to-Right" model can be looked on as an average of the "Tear Duct-to-Outside" model and the "Outside-to-Tear Duct" model. In the CASIA database, the curve remained steady between approximately 45%-55% exposure. This observation matches the simulation results using our own database.

The performance of partial iris identification for the "Radial Inside-to-Outside" Model is shown in Fig. 9, while the curves for the "Radial Outside-to-Inside" model are shown in Fig. 10. In Fig. 9, the accuracy rate increases much more dramatically than the other methods, and as a result, the "knee" for this model is located at approximately 20% of iris pattern exposure. In Fig. 10, the accuracy rate increases quickly up to 20%, then increases at a slower rate.

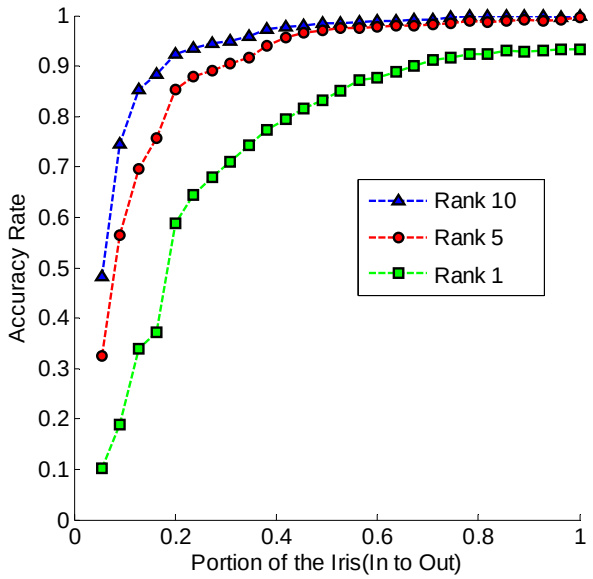


Figure 9. Partial iris identification performance for "Radial Inside-to-Outside" model.

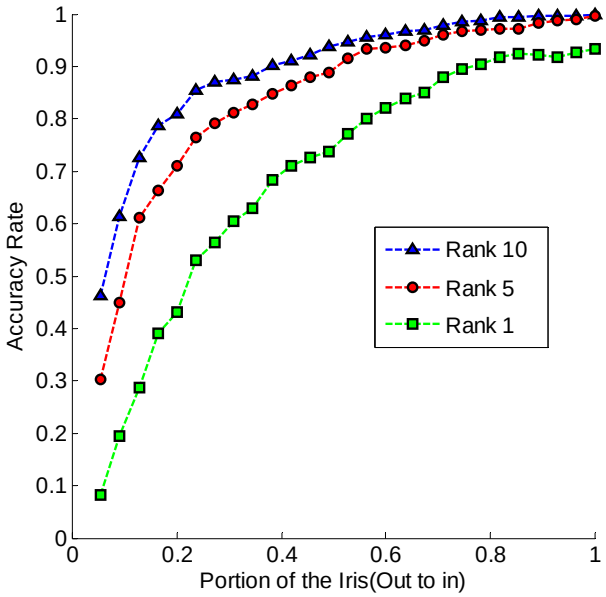


Figure 10. Partial iris identification performance for "Radial Outside-to-Inside" model.

By setting a threshold for acceptance at a 95% accuracy rate (for rank 10 matching), the "Radial Outside-to-Inside" model requires at least 60% of the iris pattern to be present. Conversely, only 25% on the iris pattern needs to be exposed for the "Radial Inside-to-Outside" model to achieve the same accuracy rate. These experimental results support the conjecture that a more distinguishable and individually unique signal is found in the inner rings of the iris.

In all cases (Figs. 6,7,9,10), with 40% of the iris, a 90% accuracy rate can be achieved for rank 10, a 80% accuracy rate for Rank 5, and a 45% accuracy rate for Rank 1. It shows that the partial iris recognition is promising for use in human identification using a rank 10/5 technique. However, it did not perform well enough for rank 1 identification

5. CONCLUSIONS

In this paper, the performance of partial iris recognition is analyzed. The experimental results show that a more distinguishable and individually unique signal is found in the inner rings of the iris. Also, as expected, the experimental results show that the eyelids and eyelashes detrimentally affect the iris recognition result. For surveillance, it is more likely that the eye (away from the tear duct) would be captured. This is the more challenging scenario but the results show that it is still feasible. Finally, the results show that a partial iris image can be used for human identification using rank 5 or rank 10 systems.

ACKNOWLEDGEMENT

This work has been sponsored in part by the National Security Agency.

REFERENCES

[1] J. Forrester, A. Dick, P. McMenamin, and W. Lee, *The Eye: Basic Sciences in Practice*, W B Saunder, London, 2001.

[2] L. Flom and A. Safir, United States Patent No. 4,641,349 (issued February 3, 1987), *Iris Recognition System*, Washington D.C.: U.S. Government Printing Office.

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

[4] Y. Du, R. W. Ives, D. M. Etter, T. B. Welch, and C.-I Chang, "One Dimensional Approach to Iris Recognition", *Proceedings of SPIE Volume 5404*, pp. 237-247, Apr., 2004.

[5] Y. Du, R. W. Ives, D. M. Etter, and T. B. Welch, "Use of One-Dimensional Iris Signatures to Rank Iris Pattern Similarities", submitted to *Optical Engineering*, 2004.

[6] R.P. Wildes, J.C. Asmuth, G.L. Green, S.C. Hsu, R.J. Kolczynski, J.R. Matey, and S.E. McBride, "A Machine Vision System for Iris Recognition", *Mach. Vision Application*, Vol. 9, 1-8, 1996.

[7] W.W. Boles and B. Boashash, "A Human Identification Technique Using Images of the Iris and Wavelet Transform", *IEEE Transactions on Signal Processing*, Vol. 46, No. 4, 1998.

[8] Y.-P. Huang, S.-W. Luo, E.-Y. Chen, "An Efficient Iris Recognition System", *the First International Conference on Machine Learning and Cybernetics*, pp. 450-454, 2002.

[9] Y. Zhu, T. Tan, and Y. Wnag, "Biometric Personal Identification Based on Iris Patterns", *15th International Conference on Pattern Recognition*, Vol. 2 , pp. 801 –804, 2000.

[10] L. Ma, Y. Wang, and T. Tan, "Iris Recognition Using Circular Symmetric Filters", *16th International Conference on Pattern Recognition*, Vol. 2 , pp. 414-417, 2002 .

[11] Y. Du, R. W. Ives, D. M. Etter, and T. B. Welch, "One-dimensional Iris Signature for Identification", U.S. Patent Pending, (Navy case number: 96,365), 2004.

[12] Y. Du, R. W. Ives, and D. M. Etter, "Iris Recognition", a chapter on biometrics, *the Electrical Engineering Handbook*, 3rd Edition, Boca Raton, FL: CRC Press, 2004 (in press).

[13] CASIA Iris Image Database, http://www.sinobiometrics.com

[14] Y. Du, B. Bonney, R. W. Ives, D. M. Etter, and R. Schultz, "Partial Iris Recognition Using a 1-D Approach: Statistics and Analysis," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2005.

Effect of Image Compression on Iris Recognition

Robert W. Ives, Bradford L. Bonney, Delores M. Etter
EE Department, U.S. Naval Academy, Annapolis, MD 21402

Abstract *– Iris recognition is a proven, accurate means to identify people. Commercial iris recognition systems are currently employed to allow passengers in some airports to be rapidly processed through security, to allow access to secure areas, and for secure access to computer networks. With the growing employment of iris recognition systems and associated research to support this, the need for large databases of iris images is growing. If required storage space is not adequate for these images, compression is an alternative. It allows a reduction in the space needed to store these iris images, although it may be at a cost in some amount of information lost in the process. This paper investigates the effects of image compression on iris recognition. Compression is performed using JPEG2000, and the iris recognition algorithm used is based on several methods, including the Daugman algorithm.*

Keywords *– Iris recognition, Daugman, Hamming distance, JPEG2000.*

I. INTRODUCTION

Biometric identification or verification of identity is currently a very active field of research. Many applications that require some degree of confidence concerning the personal identification of the people involved, such as banking, computer network access or physical access to a secure facility, are moving away from the use of paper or plastic identity cards, or alpha-numeric passwords. These systems are too easy to defeat. A higher degree of confidence can be achieved by using unique physical and/or behavioral characteristics to identify a person; this is biometrics.

In order to use biometrics for identification, the biometric data must be collected by some means from the individuals in question. In some cases, this may be a costly and time-consuming process, and the data obtained is valuable and must be protected. Additionally, data collections can create an inordinate amount of data that puts a strain on the available storage. To alleviate this problem, one available option is compression. In many applications where compression is required, but no loss of information is acceptable (such as monetary transactions or some medical applications), lossless compression is necessary; that is, compression without loss of information.

There are many lossless compression algorithms available that work best on certain types of data, such as predictive coding for one-dimensional waveform data and string coding for text. For imagery, JPEG2000 and lossless-JPEG have demonstrated very good lossless compression performance with most types of imagery. Unfortunately, lossless compression has a major drawback in that the reduction in file size is on the order of only 1.5:1 to 3:1 for many types of imagery. On the other hand, these algorithms can readily compress data further if some loss of information is tolerable. It is up to the user of the data to determine how much loss of information is acceptable.

The iris (see Fig. 1) is the colored portion of the eye that surrounds the pupil. Its combination of pits, striations, filaments, rings, dark spots and freckles that is evident under near-infrared (NIR) light make for a very accurate means of biometric identification [1]. Its uniqueness is such that even the left and right eye of the same individual is very different.

In this paper, we investigate the effects of lossy compression on the ability of an iris recognition system to accurately identify individuals. The performance is evaluated by means of the change in Hamming distances between IrisCodes using an iris recognition implementation based on several algorithms, including the Daugman algorithm [1]. Typically, a database for an iris recognition system does not contain actual iris images, but rather it stores a binary file that represents each processed iris, such as Daugman's IrisCode, stored as 512 bytes per eye. The size of such a database may not necessarily be prohibitive. However, we do not propose compressing this template data, but instead the original images from which they were created. We seek to compress the original imagery because it is this data that is valuable, and serves as training and testing imagery for the development of new algorithms. Its importance became apparent to the authors as we began to collect our own iris database, which is discussed in the next section.

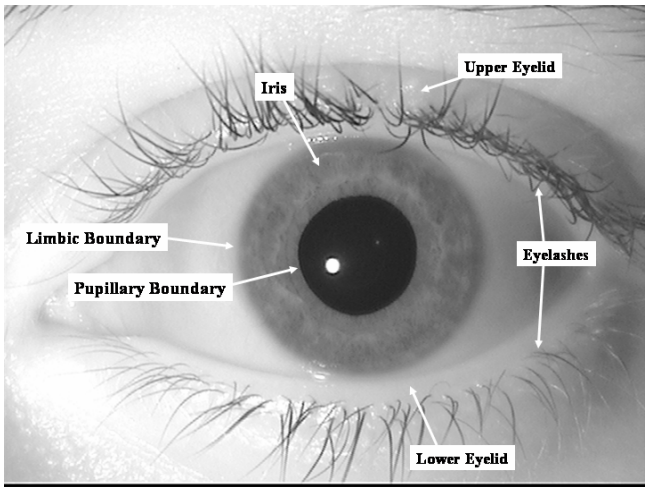


Figure 1: An Example Near-Infrared Iris Image

Compression has been investigated and used in some biometric applications, such as the FBI standard for fingerprint compression [2]-[3], or using MPEG compression [4]-[5] for video that may be used in facial recognition applications. There has been some limited research in the area of iris image compression [6], but this was compression applied to IrisCodes, not iris images. Here, we address the issue of compression applied to the iris imagery itself.

II. JPEG2000

JPEG2000 is the new compression standard published by the Joint Photographic Experts Group [7]. It employs state-of-the art compression techniques based on wavelet technology. Like the previous JPEG standard, it allows for both lossless and lossy compression of imagery. Lossy compression means that some information is lost in the process, and the amount of information lost is dependent on the algorithm used for compression, as well as the amount of compression desired (that is, the size of the compressed file).

JPEG2000 offers some advanced features, such as region-of-interest (ROI) coding, where the user could identify regions of the image that should be compressed to a higher quality than the surroundings. ROI coding might prove advantageous in iris image compression, since it would allow the iris itself to be compressed with less loss of information than other areas of the image that are not used in recognition. For this research, both lossless and lossy compression of iris images were tested using the default parameters and options for JPEG2000. JPEG2000 was implemented using Win32 executable code freely available from Kakadu Software [8].

Fig. 2 displays an original iris image before and after compression to 20:1 using JPEG-2000. The original image was collected with the LG IrisAccess 3000 system. Comparing the original and the compressed image closely will reveal some detectable differences, primarily in the areas of high detail in the original image where compression artifacts or smoothing is noted. Statistically, the two images are not very different; the maximum difference between the two images is 26 gray levels, and the overall average difference is 0.056328 with a standard deviation of 2.951321. Overall, JPEG-2000 does a good job of maintaining the detail information even up to a compression of 20:1.

III. IRIS RECOGNITION ALGORITHM

Commercial iris recognition systems today use the algorithm developed by John Daugman [1]. This patented algorithm is not available for free use, so an alternative for research purposes can be found in the implementation created by Libor Masek [9]. This algorithm follows the Daugman algorithm to some extent, but also incorporates parts of other reported algorithms. Most notably, the MATLAB code is freely available [10].

The Masek algorithm differs from the Daugman algorithm in several areas. This includes the use of the Hough transform

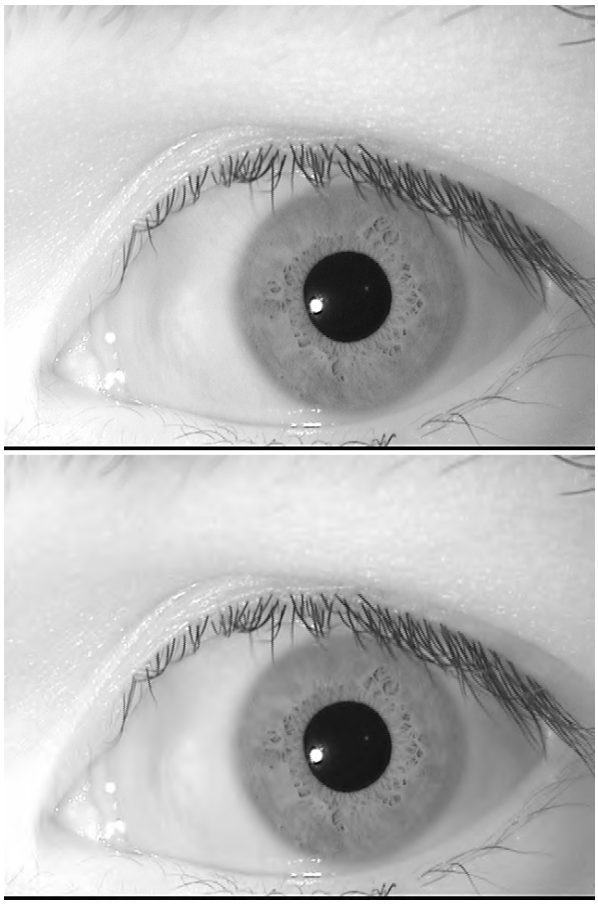


Figure 2: An original iris image (top) and compressed to 20:1 with JPEG-2000 (bottom).

to detect the circular inner iris boundary (the pupil) and outer iris boundary and its use of Log-Gabor wavelets vice Gabor wavelets for feature coding. When an image is input to the algorithm, the output is composed of two parts: the phase-code bits that represent the distinct patterns within the iris; and a mask which represents the locations of iris pattern bits which are used to compare irises, as opposed to noise that is present in the image among the iris patterns (such as eyelashes, glare, etc.), but should not be used for comparison.

IV. METHODOLOGY

The images used in this research come from two sources. First, we used the Chinese Academy of Sciences (CASIA) iris database [11]. This is composed of images of 108 different eyes, with 7 images of each eye (totaling 756 iris images). These images are 320x280 8-bit bitmapped images (.bmp), each occupying 92,160 bytes on a hard drive. A second database was comprised of images collected using the video output of our lab's LG IrisAccess 3000. This video was fed to a Matrox Meteor II frame grabber installed in a Dell Dimension 4600 desktop computer. Using the MATLAB Image Acquisition toolbox, the video was piped directly into

a graphical user interface (GUI) that runs on MATLAB 7.0. Using this GUI, the user is set to capture video at 10 frames/sec for one second (these numbers are based on our desire to capture 10 images of the same iris). Each frame becomes a 640x480 8-bit bitmapped image (.bmp), and occupies 309,248 bytes on the hard drive.

Performance was measured by observing the effect on fractional Hamming distances between the IrisCodes from the original and decompressed images, computed using the Masek algorithm [9]. The fractional Hamming distance (HD) between IrisCodes A and B is defined as:

$$HD = \frac{\left\| (\text{code A} \otimes \text{code B}) \cap \text{mask A} \cap \text{mask B} \right\|}{\left\| \text{mask A} \cap \text{mask B} \right\|}. \quad (1)$$

The $\otimes$ operator is the Boolean XOR operation to detect disagreement between the pairs of phase code bits in the two IrisCodes (code A and code B), and mask A and B identify the values in each IrisCode that are not corrupted by artifacts such as eyelids/eyelashes and specularities. The $\cap$ operator is the Boolean AND operator. The $\| \cdot \|$ operator is used to sum the number of "1" bits within its argument. The denominator of (1) ensures that only the phase-code bits that matter are included in the calculation, after any artifacts are discounted. This serves as a measure of recognition performance, as it is the fractional Hamming distance that determines if identification has been made. A value of HD = 0 indicates a perfect match between the IrisCodes, while typically a Hamming distance of ≤ 0.32 allows identification with high confidence and is here used as a threshold for recognition.

We compressed 44 images from each database using lossy JPEG-2000 to compression ratios of 4:1, 6:1, 8:1, 10:1, then 11:1, 12:1, etc. up to 20:1 and their IrisCodes were created. As mentioned in Section II, when using JPEG-2000, the default compression parameters were selected; the only option chosen was the desired bit rate (i.e., compression ratio). For each original iris image, there were 14 compressed versions, which populated each database with 660 images (44 x 15). To derive the performance results, each original iris image was compared against every other image in the database. This means that a total of 28,996 comparisons were made (659 x 44), of which 616 comparisons (44 x 14) were enrollee attempts (the irises should match) and 28,380 (44 x 43 x 15) were imposter attempts (the irises should not match).

V. RESULTS

To form a baseline regarding compression of iris images, JPEG-2000 was used first to compress the iris images without loss of information. Lossless compression allows exact reproduction of the original image from the compressed file. Depending on the algorithm used, the size of the compressed file will vary. In addition, different images will result in different compression attainable when using the same algorithm. The lossless compression results are summarized in Table 1 for each database. Here, the average lossless compression ratio achieved for the 44 images of each database are presented.

Table 1: Lossless Compression by Database

	CASIA	LGIris
Average Lossless CR Achieved	1.74	2.188

Lossy compression effects on recognition performance were evaluated using the False Acceptance Rate (FAR) and False Rejection Rate (FRR), with results summarized in Table 2. A value of HD $\leq$ 0.32 was used to determine whether a match had been made. In computing these values for the LGIris database, there were zero matches made out of 28,380 imposter attempts (images that should not have matched), but four false rejections out of the 616 enrollee attempts (images that should have matched). For the CASIA database, there were 38 false rejections and 0 false matches using the same number of attempts as for the LGIris database. The definition used in Table 2 for the FRR is defined in [12] as

$$(\%)FRR = \frac{\#\text{ of incidents of false rejections}}{\text{total \# of samples}} \times 100\% \quad (2)$$

and the FAR is defined as

$$(\%)FAR = \frac{\#\text{ of incidents of false acceptance}}{\text{total \# of samples}} \times 100\%. \quad (3)$$

In these formulas, for each database, the denominator is 28,996, as stated in Section IV.

Table 2: Recognition Performance Results by Database

	CASIA	LGIris
FAR (%)	0	0
FRR (%)	0.131	0.00138

Concerning the false rejection rate, it is important to note that three of the four false rejections in the LGIris database were associated with the same iris image, while in the CASIA database, three of the images resulted in 33 of the 38 false rejects. One of the CASIA images (image number 064_1_1) resulted in 14 false rejections in 14 attempts. This image is shown in Fig. 3.

Closer inspection of this image reveals some degree of blurring of the iris as well as capture artifacts (noticeable on the eyelashes), not to mention the occlusion of the iris by the upper eyelid and eyelashes. We attribute the poor results using this eye to the image quality. Overall, the quality of the LGIris database imagery is superior to the CASIA imagery.

Typical HD results using the LGIris database are illustrated in Table 3, here for an iris image labeled "Iris00001." The left column denotes the compression ratio

applied to test images to which the original Iris00001 is compared. The middle column displays the Hamming distance computed when the IrisCode for the original Iris00001 was compared against itself and also against compressed versions of itself. The right column was derived by comparing the original Iris00001 IrisCode with an uncompressed image of a different eye (referred to as "Iris00002"), as well as compressed versions of Iris00002.

Table 3: LGIris Database Hamming Distances (HD)

Compression Ratio	Iris 00001 (same eye)	Iris 00002 (different eye)
None	0	0.47078
4:1	0.04188	0.46853
6:1	0.14671	0.46335
8:1	0.12487	0.46339
10:1	0.12420	0.46480
11:1	0.08567	0.47090
12:1	0.08791	0.46882
13:1	0.12366	0.46975
14:1	0.12961	0.46978
15:1	0.13414	0.47019
16:1	0.10559	0.46953
17:1	0.11156	0.46841
18:1	0.11409	0.46666
19:1	0.15485	0.46533
20:1	0.19828	0.46798
Note: HD ≤ 0.32 determines recognition		

VI. CONCLUSIONS

From these results, JPEG-2000 has proven to be a very capable lossy compressor of NIR iris imagery. There was no effect on the false acceptance rate, and only a very slight effect on the false rejection rate. This is noteworthy, given the relatively high compression ratios these images were subjected to. Overall, this means that iris database storage could be reduced in size, possibly by a factor of 20 or even higher (since 20:1 was the limit of compression in this study), and have only a very minor affect on system performance. Further analysis of the false rejections is warranted, and research into how these results scale to a larger database is in progress.

As a state-of-the-art lossless compressor, compression of these iris images using lossless JPEG-2000 could reduce the required storage for a database to approximately ½ of its original size. This may be sufficient in some cases, but significant improvement can be achieved with lossy compression.

Further testing using JPEG-2000 is feasible and in progress to determine additional limits. One feature of JPEG-2000 that was not incorporated in this research was the use of regions of interest. A priori knowledge of a region of interest that should be preserved with less information loss should improve these results. For example, determination of the pupil's location, a relatively simple task in iris preprocessing, would allow identification of an area of interest such that the eye portion of the eye image could be preserved with better quality than surrounding areas (such as eyelids, forehead, etc.). In addition, other options of JPEG-2000, such as choice of wavelet filters can also be examined.

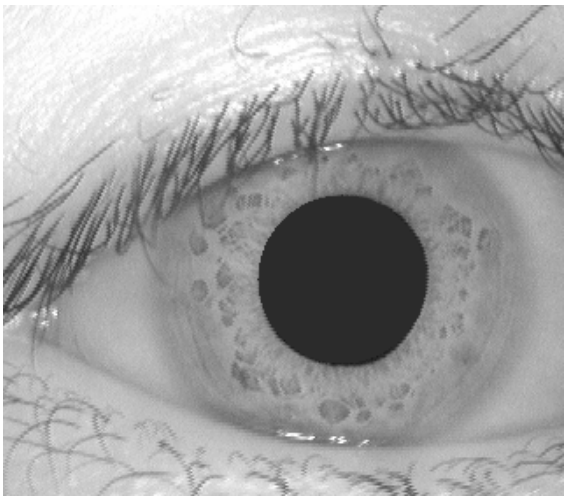


Figure 3: CASIA Image 064_1_1. This image resulted in 14 out of 14 false rejections when compressed.

ACKNOWLEDGEMENTS

Portions of this research used the CASIA iris image database [11] collected by Institute of Automation, Chinese Academy of Sciences. This work has been sponsored in part by the U.S. National Security Agency.

REFERENCES

[1] J. Daugman "How iris recognition works," *IEEE Trans. on Circuits and Systems for Video Technology.*, Vol. 14, No. 1, pp. 21-30.

[2] C. M. Brislawn, "The FBI Fingerprint Image Compression Specification," in *Wavelet Image and Video Compression*, P. N. Topiwala, Ed. Boston, MA: Kluwer, 1998, ch. 16, pp. 271–288, invited book chapter.

[3] J. N. Bradley and C. M. Brislawn, "Compression of fingerprint data using the wavelet vector quantization image compression algorithm," Los Alamos Nat'l. Lab, Tech. Report LA-UR-92-1507, Apr. 1992, FBI report.

[4] J. Daugman, "Face and gesture recognition: overview," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 675-676, 1997.

[5] H. Wang and S.F. Chang, "A highly efficient system for automatic face region detection in MPEG video," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 4, pp. 615-628, 1997.

[6] U. von Seelen, "IrisCode template compression and its effects on authentication performance," Presentation at the Biometrics Consortium Conference 2004, http://www.biometrics.org/bc2004/CD/PDF_PROCEEDINGS/Microsoft%20PowerPoint%20-%20Von SeelenBrief_Update.ppt%20%5BRead-Only%5D.pdf.

[7] The JPEG2000 Standard, http://www.jpeg.org/jpeg2000/index.html, Feb. 27, 2005.

[8] Kakadu Software, http://www.kakadusoftware.com/, Feb. 27, 2005.

[9] L. Masek, "Recognition of Human Iris Patterns for Biometric Identification," M. Thesis, The University of Western Australia, 2003, www.csse.uwa.edu.au/~pk/studentprojects/libor/LiborMasekThesis.pdf, Mar. 26, 2005.

[10] Libor Masek, Peter Kovesi. *MATLAB Source Code for a Biometric Identification System Based on Iris Patterns*. The School of Computer Science and Software Engineering, The University of Western Australia. 2003.
[11] CASIA Iris Image Database, http://www.sinobiometrics.com.
[12] John Woodward, Nicholas Orlans and Peter Higgins, *Biometrics: Identity Assurance in the Information Age*, New York, NY: McGraw-Hill, 2003.
[13] Y. Du, B. Bonney, R. W. Ives, D. M. Etter, and R. Schultz, “Partial Iris Recognition Using a 1-D Approach: Statistics and Analysis,” presented at the *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP), Philadelphia, PA, Mar 2005.
[14] Y. Du, R.W. Ives, D.M. Etter, T.B. Welch, and C.-I Chang, "One dimensional approach to iris recognition", *Proceedings of the SPIE*, pp. 237-247, Apr., 2004.