Iris Recognition

Collection Editor:
Dmitry Khabashesku

Iris Recognition

Collection Editor:
Dmitry Khabashesku

Authors:
David Carr
Dmitry Khabashesku
Bryan Lipinski
Paul Robichaux

Online:
< http://cnx.org/content/col10256/1.1/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: December 18, 2004
PDF generated: February 4, 2011
For copyright and attribution information for the modules contained in this collection, see p. 35.

Table of Contents

Chapter 1

Motivation behind Iris Detection[1]

1.1 Passwords are Bad

Pin numbers, email passwords, credit card numbers, and protected premises access numbers all have something in common. All of them are a key to your identity, and all of them can easily be stolen or guessed after reading the first few pages of "Identity Theft for Dummies".

Currently users have been encouraged to create strong passwords for every different domain. This leads to some logical problems. People tend to forget multiple, lengthy and varied passwords, therefore, they use one strong password for everything. This only allows the successful thief to gain access to all the protected information. The other option which follows is to carry a hard copy of each password which again can only be a reward for the quick pick-pocket.

Only recently have companies started to use biometric authentication to protect access to highly confidential assets. You may be familiar with some of the physical traits used by biometric authentication programs such as fingerprints and retinas. Other traits that can be measured include the voice, face, and the iris. For most people, these are only high-tech gadgets simulated in hollywood. However, this technology is very real and is currently being used in the private sector. One method of particular interest is the use of iris codes to authenticate users.

[1] This content is available online at <http://cnx.org/content/m12488/1.2/>.

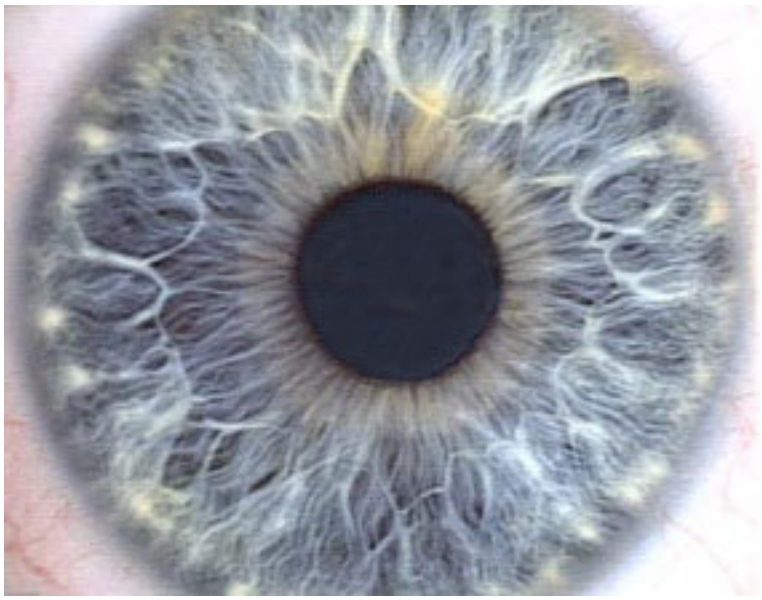


Figure 1.1: The Eye www.icdri.org[2]

1.2 Enter: Iris Detection

Iris detection is one one of the most accurate and secure means of biometric identification while also being one of the least invasive. Fingerprints of a person can be faked–dead people can come to life by using a severed thumb. Thiefs can don a nifty mask to fool a simple face recognition program. The iris has many properties which make it the ideal biometric recognition component.

The iris has the unique characteristic of very little variation over a life's period yet a multitude of variation between individuals. Irises not only differ between identical twins, but also between the left and right eye. Because of the hundreds of degrees of freedom the iris gives and the ability to accurately measure the textured iris, the false accept probability can be estimated at 1 in 10^31. Another characteristic which makes the iris difficult to fake is its responsive nature. Comparisons of measurements taken a few seconds apart will detect a change in iris area if the light is adjusted–whereas a contact lens or picture will exhibit zero change and flag a false input.

[2] http://www.icdri.org/biometrics/iris_biometrics.htm

Chapter 2

Iris Recognition: Detecting the Pupil[1]

2.1 Acquiring the Picture

Beginning with a 320x280 pixel photograph of the eye taken from 4 cenimeters away using a near infrared camera. The near infrared spectrum emphasizes the texture patterns of the iris making the measurements taken during iris recognition more precise. All images tested in this program were taken from the **Chinese Academy of Sciences Institute of Automation (CASIA)** iris database.

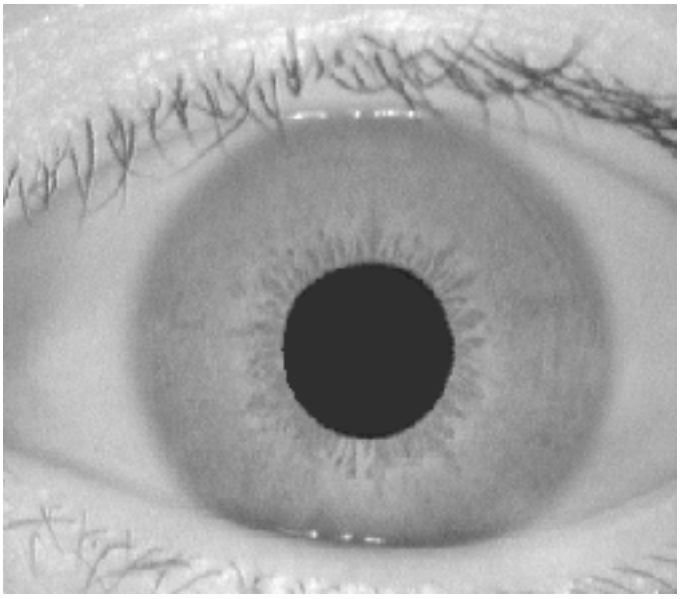


Figure 2.1: Near-infrared image of eye from CASIA Database

[1] This content is available online at <http://cnx.org/content/m12487/1.4/>.

2.2 Edge Detection

Since the picture was acquired using an infrared camera the pupil is a very distinct black circle. The pupil is in fact so black relative to everything else in the picture a simple edge detection should be able to find its outside edge very easily. Furthermore, the thresholding on the edge detection can be set very high as to ignore smaller less contrasting edges while still being able to retrieve the entire perimeter of the pupil.

The best edge detection algorithm for outlining the pupil is **canny edge detection**. This algorithm uses horizontal and vertical gradients in order to deduce edges in the image. After running the canny edge detection on the image a circle is clearly present along the pupil boundary.

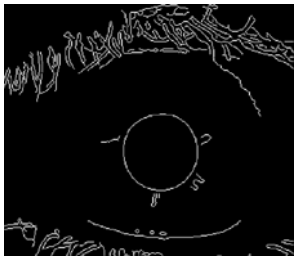


Figure 2.2: Canny edge detected image of the eye

2.3 Image Clean Up

A variety of other filters can be used in order decrease the extraneous data found in the edge detection stage. The first step in cleaning up the image is to dilate all the edge detected lines. By increasing the size of the lines nearby edge detected components are likely to coalesce into a larger line segment. In this way complete edges not fully linked by the edge detector can form. Thus the dilation will give us a higher probability that the perimeter of the pupil is a complete circle.

Knowing that the pupil is well defined more filters can be used without fear of throwing out that important information. Assuming the image is centered a filter can be used to fill in the circle defined by the pupil's perimeter. In this way we clearly define the entire area of the pupil. After this, a filter which simply throws out sections of connected pixels with an area below a threshold can be used effectively to throw out smaller disconnected parts of the image the edge detector found. Finally, any holes in the pupil caused by reflections or other distortions can be filled, by looking for sections of blank pixels with an area below a threshold. After this processing we achieve a picture that highlights the pupil area while being fairly clean of extraneous data.

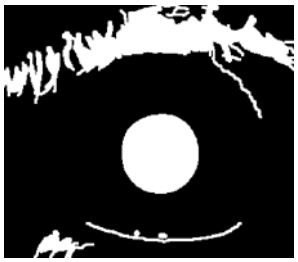


Figure 2.3: Image after final filters

2.4 Pupil Information Extraction

Having pre-processed the image sufficiently the extraction of the pupil center and radius can begin. By computing the euclidean distance from any non-zero point to the nearest zero valued point an overall spectrum can be found. This spectrum shows the largest filled circle that can be formed within a set of pixels. Since the pupil is the largest filled circle in the image the overall intensity of this spectrum will peak in it.

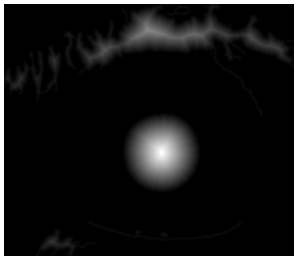


Figure 2.4: Image after computing minimal euclidean distance to non-white pixel

In the pupil circle the exact center will have the highest value. This is due to the simple fact that the center is the one point inside the circle that is farthest from the edges of the circle. Thus the maximum value must correspond to the pupil center, and furthermore the value at that maximum (distance from that point to nearest non-zero) must be equal to the pupil radius.

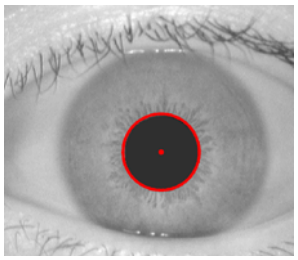


Figure 2.5: The original image of the eye with the pupil center and perimeter, found with the algorithm, highlighted

Chapter 3

Iris Recognition: Detecting the Iris[1]

3.1 Iris Detection

With the information on the pupil discovered the location of the iris can now begin. It is important to note that the pupil and iris are not concentric. Consequently, the pupil information does not help directly determine the same parameters for the iris. However, the pupil information does give a good starting point, the pupil center.

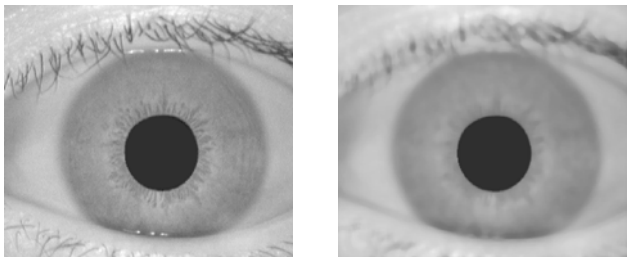
Most modern iris detection algorithms use random circles to determine the iris parameters. Having a starting point at the pupil, these algorithms guess potential iris centers and radii. They then integrate over the circumference in order to determine if it is on the border of the iris. While this is highly accurate the process can consume a lot of time. This module explains an alternate approach which is much more lightweight but with higher error rates.

3.2 Iris Radius Approximation

The first step in finding the actual iris radius is to find an approximation of the iris radius. This approximation can then be fine tuned to find the actual iris parameters. In order to find this approximation a single edge of the iris must be found. Knowing that eyes are most likely to be distorted in the top and bottom parts due to eyelashes and eyelids, the best choice for finding an unobstructed edge is along the horizontal line through the pupil center.

Having decided on where to attempt to detect the iris edge, the question of how to do it arises. It seems obvious that some type of edge detection should be used. It happens that for any edge detection it is a good idea to blur the image to subtract any noise prior to running the algorithm, but too much blurring can dilate the boundaries of an edge, or make it very difficult to detect. Consequently, a special smoothing filter such as the **median filter** should be used on the original image. This type of eliminates sparse noise while preserving image boundaries. The image may need to have its contrast increased after the median filter.

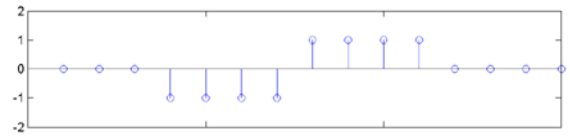
[1] This content is available online at <http://cnx.org/content/m12489/1.3/>.



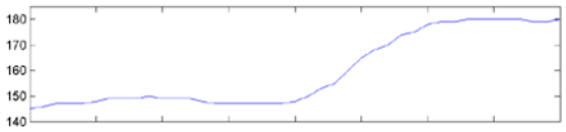
(a) (b)

Figure 3.1: The original image after running through a median filter. A median filter works by assigning to a pixel the median value of its neighnors.

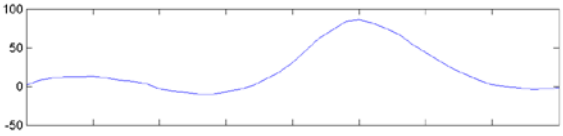
Now that the image is prepped the edge detection can be done. Since there is such a noticeable rising edge in luminescence at the edge of the iris, filtering with a **haar wavelet** should act as a simple edge detector. The area of interest is not just the single horizontal line through the iris, but the portion of that line to the left of the pupil. This is so that the rising luminescence from the transition from iris to white is the only major step.



(a)



(b)



(c)

Figure 3.2: By filtering the area of interest with a haar wavelet all rises in luminence are transformed into high valued components of the output. The sharpness of change in luminence affects the overall height of the component. (a) Haar Wavelet (b) The area of interest (c) The area of interest after filtering with the haar wavelet

The iris should represent the steepest luminence change in the area of interest. Consequently, this area of the image should correspond to the highest valued componenet of the the output from the filter. By finding this maximal value the edge of the iris to the right of the pupil should be found. It should be noted that since the iris may not be concentric with the pupil the distance from the pupil center to this edge may not correspond to the iris' radius.

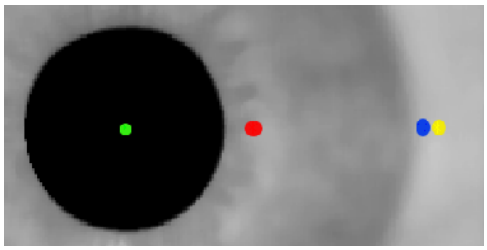
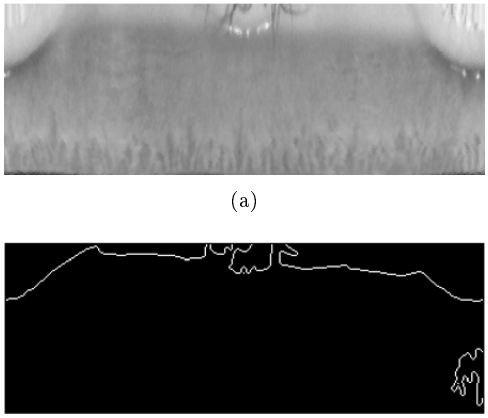


Figure 3.3: The green point is the pupil center found using the pupil detection techniques of part 1. The red point indicates the starting point of the area of interest. The blue point is the approximate radius found. The yellow point is the padded radius for use in finding the actual iris parameters.

3.3 Iris Translation

Having acquired an approximate radius, a small pad of this value should produce a circle centered on the pupil which contains the entire iris. Furthermore, with the perimeter of the pupil known, an annulus may be formed which should have the majority of it's area filled by the iris. This annulus can then be unrolled into cartestian coordinates through a straight discretized transformation. (r –> y, θ –> x) The details of this procedure are described in Step 3.

If the iris is perfectly centered on the pupil, the unrolled image should have a perfectly straight line along its top. However, if the iris is off center even a little this line will be wavy. The line represents the overall distance the iris is at from the pupil center. It is this line which will help to determine the iris' center and radius. Consequently, an edge detection algorith m must be run on the strip in order to determine the line's exact location. Once again **canny edge detection** is used. However, before the edge detection can be run the image should undergo some simple pre-processing to increase the contrast of the line. This will allow for a higher thresholding on the edge detection to eliminate extraneous data.



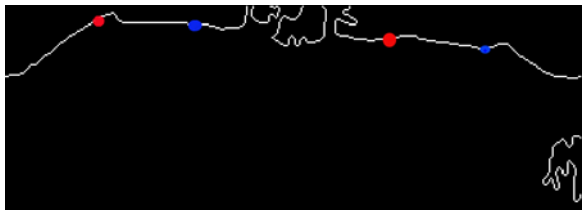
(a)

(b)

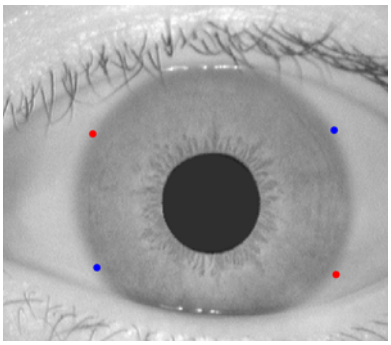
Figure 3.4: The unrolled iris before and after edge detection

3.4 Iris Information Extraction

In order to extrapolate the iris' center and radius, two chords of the actual iris through the pupil must be found. This can be easily accomplished with the information gained in the previous step. By examing points with x values on the strip offset by half of the length of the strip a chord of the iris is formed through the pupil center. It is important to pick the vectors for these chords so they are both maximally independent of each other, while also being far from areas where eyelids or eyelashes may interfere.



(a)



(b)

Figure 3.5: The points selected on the strip to form the chords of the iris through the pupil

The center of the iris can be computed by examining the shift vectors of the chords. By looking at both sides of a chord and comparing their lengths an offset can be computed. If the center was shifted by this vector it would equalize the two components of the chord. By doing this with both of the chords two different offset vectors can be computed. However, by just shifting the center through both of these vectors some components of shift will be overcompensated for due to the vectors not being orthogonal. Thus, the center should be shifted through the first vector, and the orthogonal component of the second to the first.

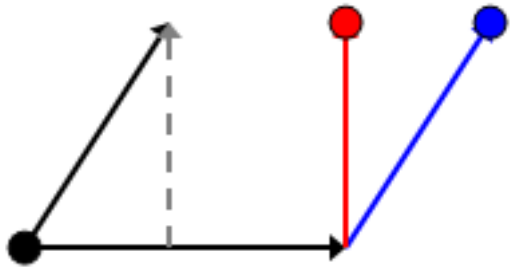


Figure 3.6: The change vectors (black) represent the shift of the pupil (black circle) in order to find the iris center. By just adding the vectors (blue vector) the result (blue circle) is offset by any vector the two change vectors share. Consequently, by adding the orthogonal component (gray vector) of one vector to the other (red vector), the actual iris center (red circle) is found.

The diameter of the iris can now be estimated by simply averaging the two diameters of the chords. While this is not a perfect estimate, that would require a single chord through the iris center, it is a very good approximation.

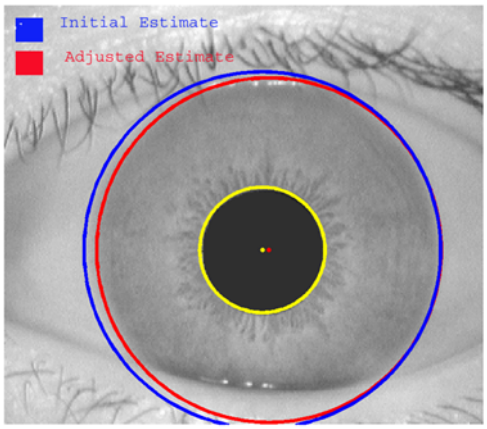


Figure 3.7: The pupil center and perimeter, along with the original estimate of the iris perimeter and the determined iris center and perimeter

3.5 Possible Improvements

Currently the algorithm only examines two static vectored chords. While these were chosen to work best within the maximal orthogonality and minimal eyelash/eyelid interference constraint there are still many cases where they do not work. It is possible for someone to have the entire upper half of their iris covered and still have the rest of the iris code generation work since less then half of the iris surface is needed for an identification. Of course, if half of the eye is shut the chords will not intersect the iris edge and no result

will occur. Instead of these static vectors for chords, an adaptive algorithm could be used which starts from the eye center and works progressively through angles to find points near the eyelid interference region. In this case the orthogonality would be maximized for that eye while still finding points on the iris edge.

Most modern iris detection algorithms produce what is known as an iris mask. This mask represents the portion of the iris obstructed by the eyelid or eyelash. This portion can then be ignored when doing the iris code comparison. In this way a user is not penalized for blocking a portion of their iris in the overall authentication. This takes advantage of the need for only a fraction of the iris for identification. The algorithm described in this module does not currently produce a mask. One way in which a mask could be produced is during the iris extrapolation. After unrolling the iris the presence of discontinuities in the line across the top the eyelids or eyelashes could be detected and a mask could be returned showing them. Another way a mask could be generated is through the running of a edge detector around the edge of the computed iris. In this way all sections of the perimeter which are messy could be found and masked.

Chapter 4

Iris Recognition: Unwrapping the Iris[1]

4.1 Why is unwrapping needed?

Image processing of the iris region is computationally expensive. In addition the area of interest in the image is a 'donut' shape, and grabbing pixels in this region requires repeated rectangular-to-polar conversions. To make things easier, the iris region is first unwrapped into a rectangular region using simple trigonometry. This allows the iris decoding algorithm to address pixels in simple (row,column) format.

4.2 Asymmetry of the eye

Although the pupil and iris circles appear to be perfectly concentric, they rarely are. In fact, the pupil and iris regions each have their own bounding circle radius and center location. This means that the unwrapped region between the pupil and iris bounding does not map perfectly to a rectangle. This is easily taken care of with a little trigonometry.

There is also the matter of the pupil, which grows and contracts its area to control the amount of light entering the eye. Between any two images of the same person's eye, the pupil will likely have a different radius. When the pupil radius changes, the iris stretches with it like a rubber sheet. Luckily, this stretching is almost linear, and can be compensated back to a standard dimension before further processing.

4.3 The unwrapping algorithm

In figure1, points Cp and Ci are the detected centers of the pupil and iris respectively. We extend a wedge of angle dθ starting at an angle θ, from both points Cp and Ci, with radii Rp and Ri, respectively. The intersection points of these wedges with the pupil and iris circles form a skewed wedge polygon P1 P2 P3 P4. The skewed wedge is subdivided radially into N blocks. and the image pixel values in each block are averaged to form a pixel (j,k) in the unwrapped iris image, where j is the current angle number and k is the current radius number.

[1] This content is available online at <http://cnx.org/content/m12492/1.3/>.

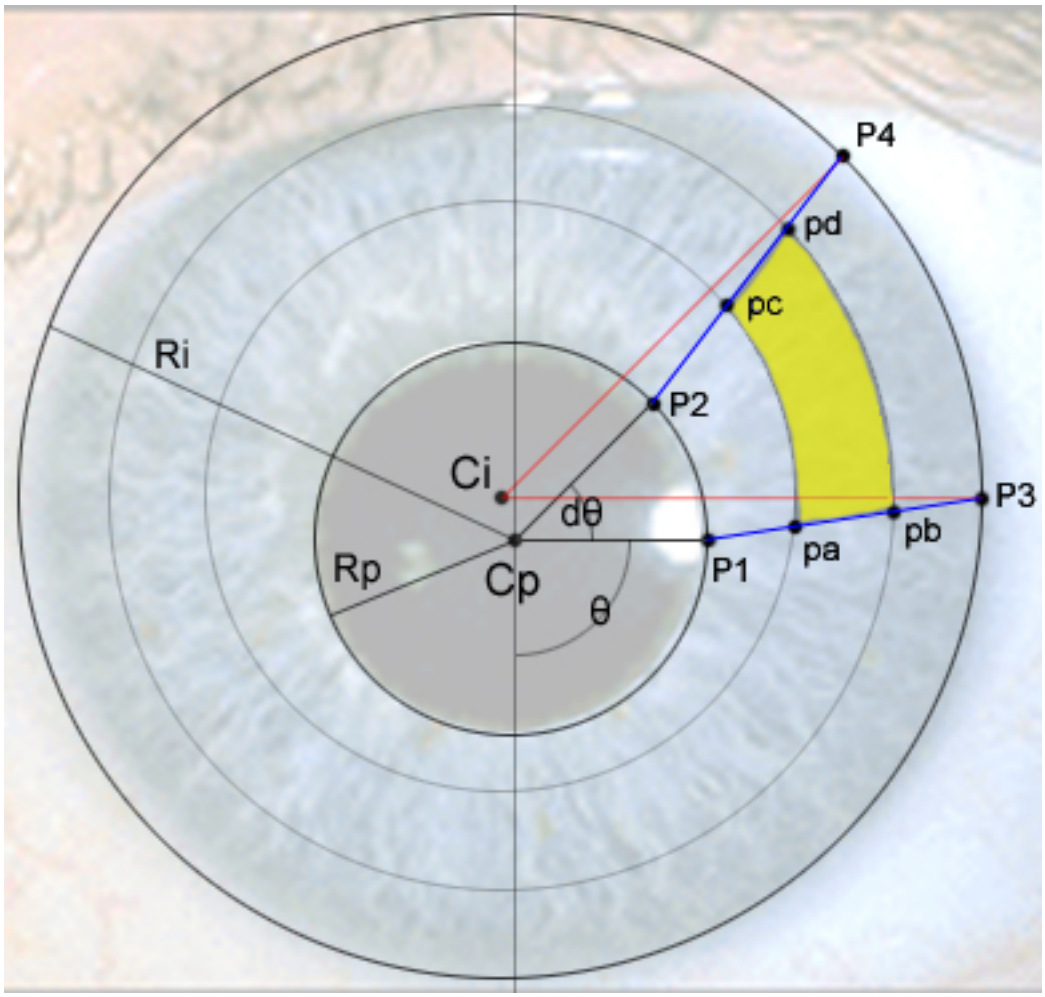


Figure 4.1: Algorithm for unwrapping the iris region.

For this project, the standard dimensions of the extracted iris rectangle are 128 rows and 8 columns (see Figure 4). This corresponds to N=128 wedges, each of angle $\frac{2\pi}{128}$, with each wedge divided radially into 8 sections. The equations below define the important points marked in Figure 1. Points Pa through Pd are interpolated along line segments P1-P3 and P2-P4.

$$P_1 = \mathrm{Cp} + \mathrm{Rp}\left(\cos\left(\theta\right) - \sin\left(\theta\right)\right)$$

$$P_2 = C_p + R_p\left(\cos\left(\theta + \mathrm{d}\theta\right) - \sin\left(\theta + \mathrm{d}\theta\right)\right)$$

$$P_3 = \mathrm{Ci} + \mathrm{Ri}\left(\cos\left(\theta\right) - \sin\left(\theta\right)\right)$$

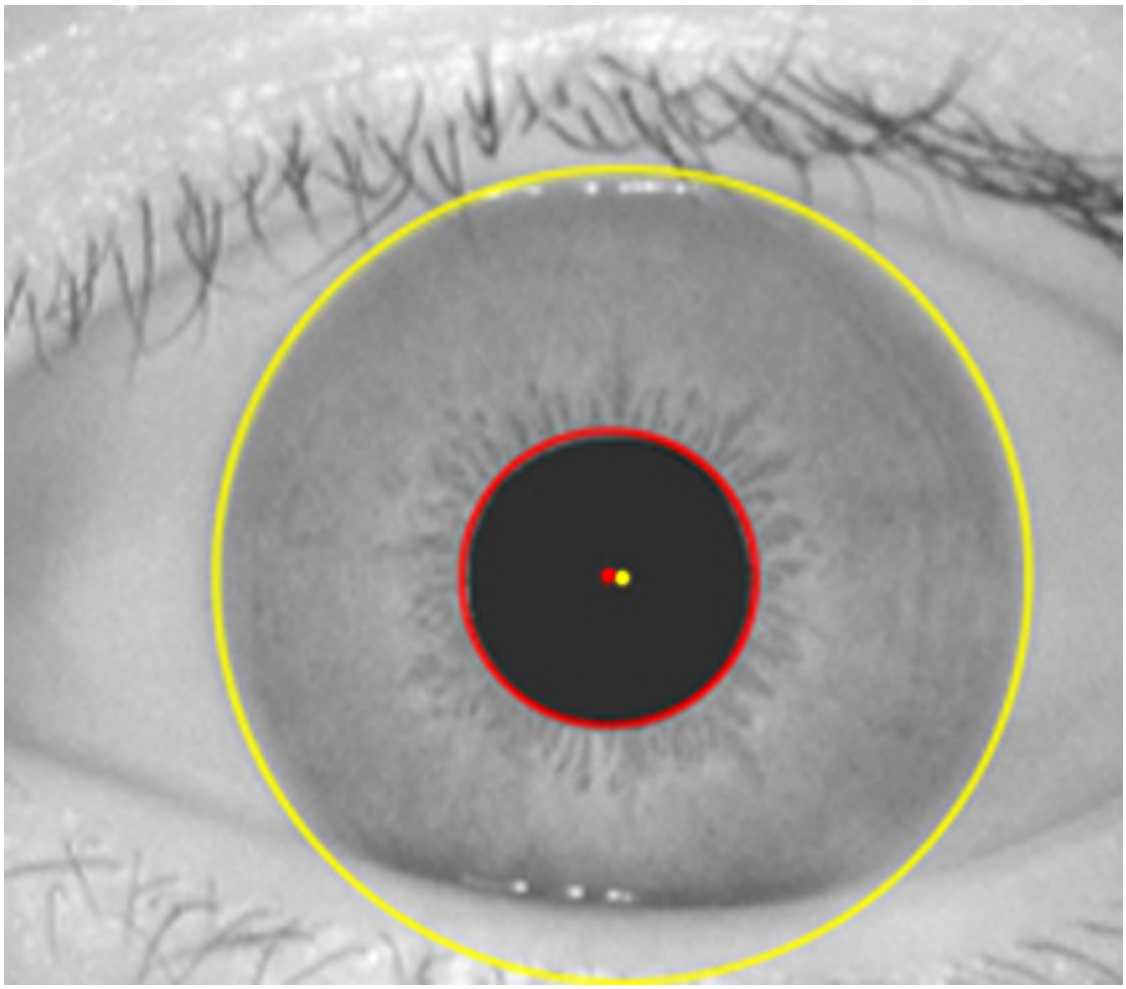
$$P_4 = C_i + R_i\left(\cos\left(\theta + \mathrm{d}\theta\right) - \sin\left(\theta + \mathrm{d}\theta\right)\right)$$

$$P_a = P_1\left(1 - \frac{k}{N}\right) + \frac{P_3 k}{N}$$

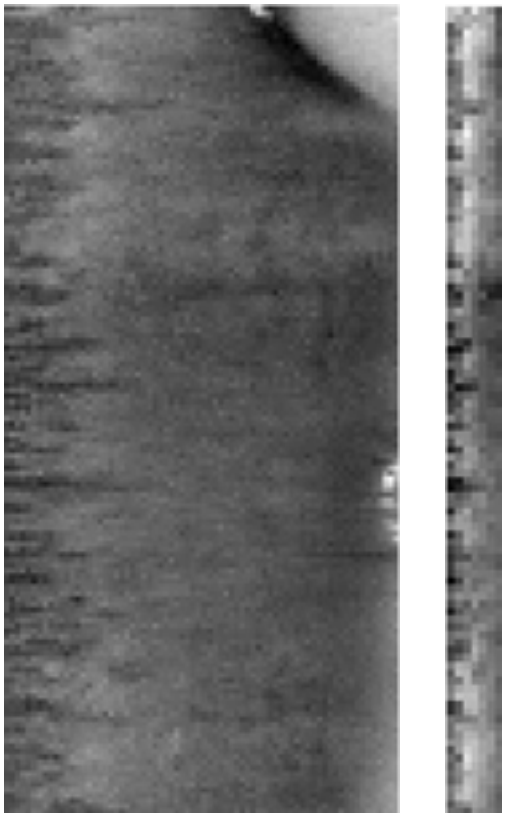
$$P_b = P_1\left(1 - \frac{k+1}{N}\right) + \frac{P_3\,(k+1)}{N}$$

$$P_c = P_2\left(1 - \frac{k}{N}\right) + \frac{P_4 k}{N}$$

$$P_d = P_2\left(1 - \frac{k+1}{N}\right) + \frac{P_4\,(k+1)}{N}$$



(a)



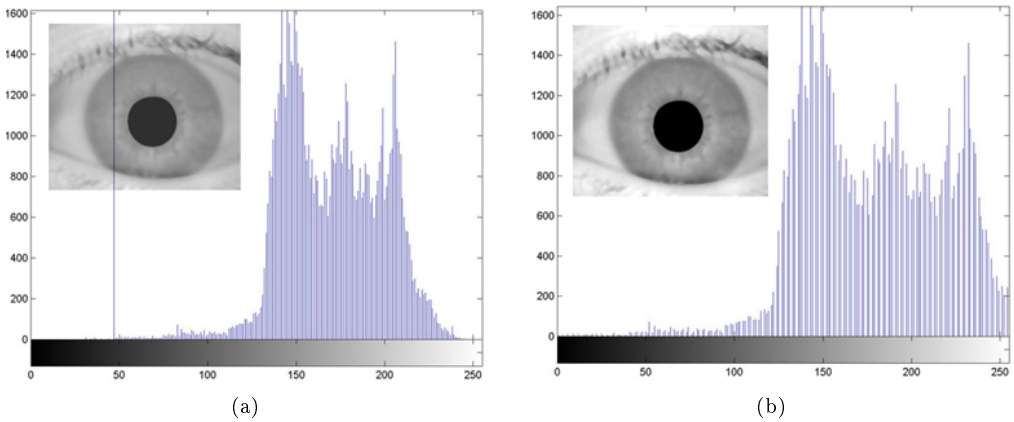
4.4 Masking of Extraneous Regions

Subfigure 2.2 demonstrates a high-resolution unwrapping. Note the large eyelid regions at the top and bottom of the image. These are the areas inside the iris circle that are covered by an eyelid. These regions do not contain any useful data and need to be discarded. One way to do this is to detect regions of the image that are unneeded and note the position of pixels within the region. Then, when the iris pattern is decoded and compared to another image, only regions that are marked "useful" in both images are considered.

A less robust method of ignoring the eyelid regions is to extract the inner 60% of the region between the pupil and iris. This assumes that an eyelid within this 50% will be detected before unwrapping and the image will be discarded. While simpler to implement, this method has the drawback that less iris data is extracted for comparison.

4.5 Contrast adjustment

Notice that subfigures 2.2 and 2.3 appear to be better contrasted than subfigure 2.1. These images have been equalized in contrast to maximize the range of luminance values in the iris image. This makes it numerically easier to encode the iris data. The equalizing is done by acquiring a luminance histogram of the image and stretching the upper and lower boundaries of the histogram to span the entire range of luminance values 0-255. Figure 3 demonstrates an example of this process.



(a) (b)

Figure 4.3: (3.1) Poor contrasted image and luminance histogram (2.2) Image transformed by stretching the luminance histogram.

Chapter 5

Iris Recognition: Gabor Filtering[1]

5.1 Gabor Wavelets

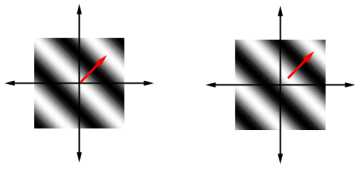
To understand the concept of Gabor filtering, we must first start with Gabor wavelets. Gabor wavelets are formed from two components, a complex sinusoidal carrier and a Gaussian envelope.

$$g(x,y) = s(x,y)\, w_r(x,y)$$

The complex carrier takes the form:

$$s(x,y) = e^{j(2\pi(u_0 x + v_0 y) + P)}$$

We can visualize the real and imaginary parts of this function seperately as shown in this figure.



(a) (b)

Figure 5.1: Image Source[2] (a) Real Part (b) Imaginary Part

The real part of the function is given by:

$$Re(s(x,y)) = cos(2\pi(u_0 x + v_0 y) + P)$$

and the imaginary:

$$Im(s(x,y)) = sin(2\pi(u_0 x + v_0 y) + P)$$

The parameters u_0 and v_0 represent the frequency of the horizontal and vertical sinusoids respectively. P respresents an arbitrary phase shift.

The second component of a gabor wavelet is its envelope. The resulting wavelet is the product of the sinusoidal carrier and this envelope. The envelope has a gaussian profile and is described by the following equation:

[1] This content is available online at <http://cnx.org/content/m12493/1.4/>.
[2] http://mplab.ucsd.edu/tutorials/pdfs/Gabor.pdf

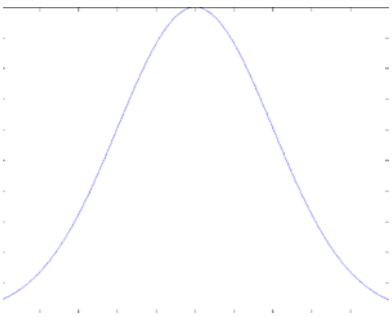


Figure 5.2: Gaussian Envelope

$$g(x,y) = Ke^{-\pi\left(a^2(x-x_0)_r^2+b^2(y-y_0)_r^2\right)}$$

where:

$$(x-x_0)_r = (x-x_0)\,cos(\theta) + (y-y_0)\,sin(\theta)$$

$$(y-y_0)_r = -(x-x_0)\,sin(\theta) + (y-y_0)\,cos(\theta)$$

The parameters used above are K - a scaling constant (a,b) - envelope axis scaling constants, θ - envelope rotation constant, (x_0,y_0) - Gausian envelope peak.

To put it all together, we multiply $s(x,y)$ by $w_r(x,y)$. This produces a wavelet like this one:

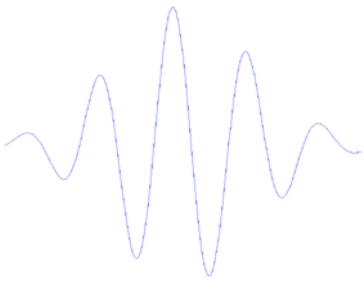
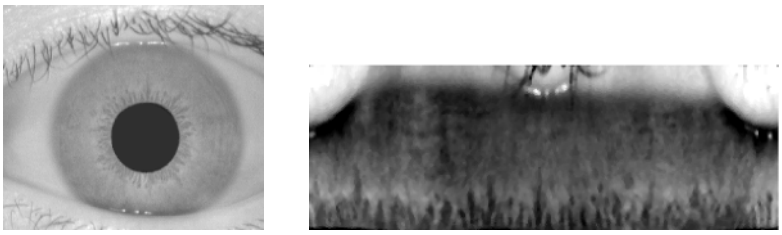


Figure 5.3: 1D Gabor Wavelet

5.2 Generating an Iris Code

Now that we have Gabor wavelets, lets do something interesting with them. Lets start with an image of an eye and then unroll it (map it to cartesian coordinates) so we have something like the following:



(a) (b)

Figure 5.4: (a) Image of Eye (b) "Unrolled" Iris

What we want to do is somehow extract a set of unique features from this iris and then store them. That way if we are presented with an unknown iris, we can compare the stored features to the features in the unknown iris to see if they are the same. We'll call this set of features an "Iris Code."

Any given iris has a unique texture that is generated through a random process before birth. Filters based on Gabor wavelets turn out to be very good at detecting patterns in images. We'll use a fixed frequency 1D Gabor filter to look for patterns in our unrolled image. First, we'll take a one pixel wide column from our unrolled image and convolve it with a 1D Gabor wavelet. Because the Gabor filter is complex, the result will have a real and imaginary part which are treated seperately. We only want to store a small number of bits for each iris code, so the real and imaginary parts are each quantized. If a given value in the result vector is greater than zero, a one is stored; otherwise zero is stored. Once all the columns of the image have been filtered and quantized, we can form a new black and white image by putting all of the columns side by side. The real and imaginary parts of this image (a matrix), the iris code, are shown here:

(a)

(b)

Figure 5.5: (a) Real Part of Iris Code (b) Imaginary Part of Iris Code

Now that we have an iris code, we can store it in a database, file or even on a card. What happens though if we want to compare two iris codes and decide how similar they are?

5.3 Comparing Iris Codes

The problem of comparing iris codes arises when we want to authenticate a new user. The user's eye is photographed and the iris code produced from the image. It would be nice to be able to compare the new code to a database stored codes to see if this user is allowed or to see who they are. To perform this task,

we'll attempt to measure the Hamming distance between two iris codes. The Hamming distance between any two equal length binary vectors is simply the number of bit positions in which they differ divided by the length of the vectors. This way, two identical vectors have distance 0 while two completely different vectors have distance 1. Its worth noting that on average two random vectors will differ in half their bits giving a Hamming distance of 0.5. The Hamming distance is mathematically defined in this equation:

$$D = \frac{A \oplus B}{\text{length(A)}}$$

In theory, two iris codes independently generated from the same iris will be exactly the same. In reality though, this doesn't happen vary often for reasons such as imperfect cameras, lighting or small rotational errors. To account for these slight inconsistencies, two iris codes are compared and if the distance between them is below a certain threshold we'll call them a match. This is based on the idea of statistical independance. The iris is random enough such that iris codes from different eyes will be statistically independent (ie: have a distance larger than the threshold) and therefore only iris codes of the same eye will fail the test of statisical independance. Empirical studies with millions of images have supported this assertion. In fact, when these studies used the threshold used in our method (.3) false positive rates fell below 1 in 10 million.

Chapter 6

Iris Recognition Results and Conclusions[1]

6.1 Iris Recognition Results

Our implementation of the iris recognition algorithm is broken up into several components, each of which has its strengths and weaknesses.

The pupil recognition algorithm appears to be 98% effective in detecting the pupil center when tested against a database of 50 images. This is due to the extremely uniform black color of the pupil and strong contrast to the iris and virtuall all other features in the image. Although the assumptions that the algorithm is founded break down in extreme cases, such as when there are other large black spots in the image, these cases can be detected by other means and discarded.

The iris detection algorithm proved to be rather hit-or-miss. In it's current form, it has a 50% success rate in detecting the iris correctly within an image of the eye. This large error lies mostly in the 'guessing' scheme used to make an initial prediction about the radius of the iris. This guessing scheme can be expanded to make higher precision, higher accuracy guesses at the expensive of algorithm execution speed. Also, the iris-sclera transition boundary of the eye can be more intensively processed by a multi-scale edge detection kernel (the implementation in this project uses a single-scale Haar wavelet kernel).

[1] This content is available online at <http://cnx.org/content/m12495/1.2/>.

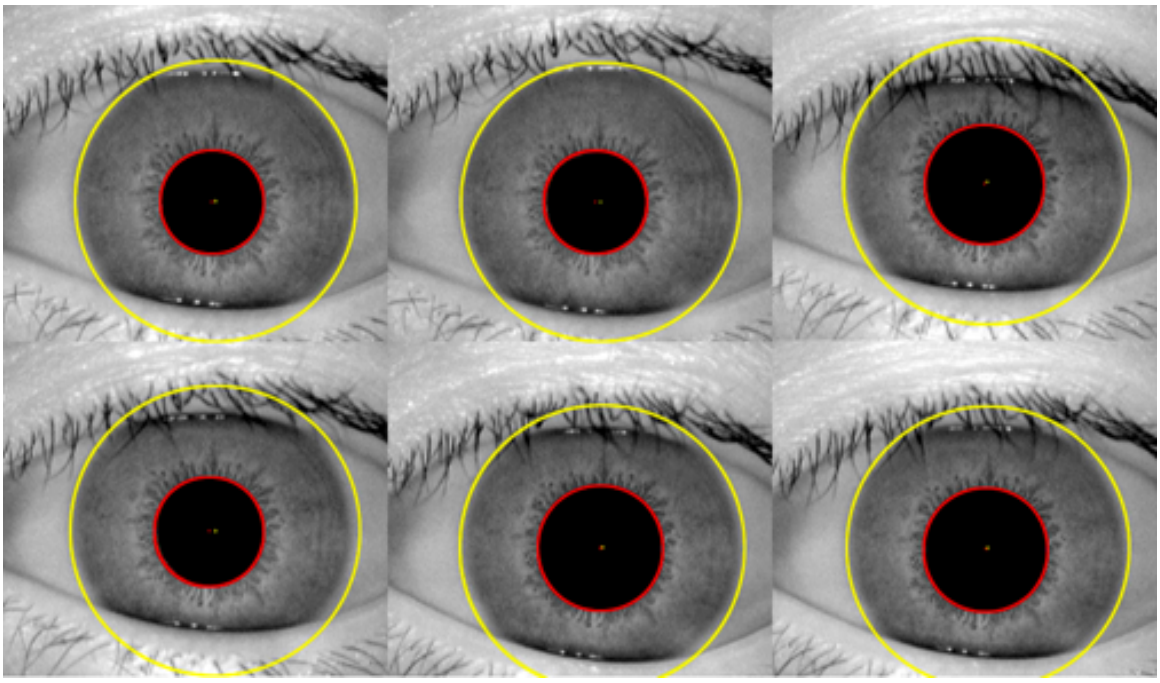


Figure 6.1: Six different images of the same eye. These were 100% verified to be the same eye for every unique combination of 2 images.

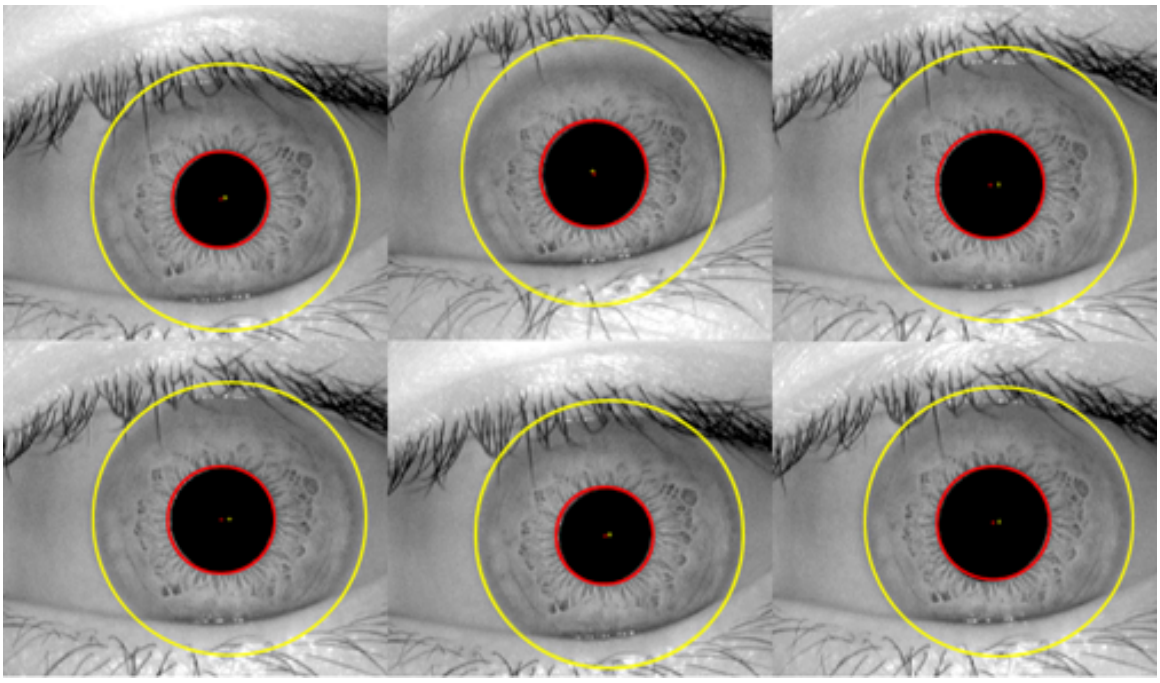


Figure 6.2: Six different images of the anothe eye. These were 80% verified to be the same eye for every unique combination of 2 images. The error is due to the presense of eyelids and eyelashes in the image.

The Gabor correlation algorithm proved to be very robust. Our implementation used a 1-D, single frequency, single phase Gabor wavelet to generate the iris codes. The threshold for statistical independence (Hamming Distance) between two iris codes was experimentally determined to be 0.3. During testing, we chose a set of images that reliably passed the other stages of the iris detection process (find pupil, find iris) so that we could test the functionality of the Gabor filter independent of errors accrued in earlier stages. Because of long computation times, small datasets of 6 images were compiled, and all combinations of two images within each dataset were compared to another. In one test (Figure 1), 100% of all combinations of 6 eyes of the same person registered as correctly positive. In another run of this same test on a different person (Figure 2), 80% of combinations of 6 eyes registered as correctly positive. This is due to the fact that our algorithm does not mask out regions of the iris obscured by eyelashes and eyelids. The second set of images had substantially more interference from eyelids, which produced errors in the Gabor codes. The first set of images is less prone to these defects, and so passes the tests better.

6.2 The Future of Iris Codes

As evident from the results, it is possible to create, relatively easily, an algorithm to detect and recognize irises to a calculated degree of confidence. In addition, after a little research (hint: google "**John Daugman**"), it is clear that more sophisticated algorithms exist that give zero false acceptance–something many other authentication techniques simply cannot deliver. One specific algorithm patented by Dr. Daugman, is currently the most accepted and widely used in iris code recognition systems. This embodiment of the algorithm uses robust algorithms in each part of the implementation (pupil and iris detection, masking, Gabor correlation), and has experimentally proven to be extremely accurate. In the largest deployment

of iris recognition systems, this algorithm does 3.8 billion comparisons a day in the United Arab Emirates (story here[2]).

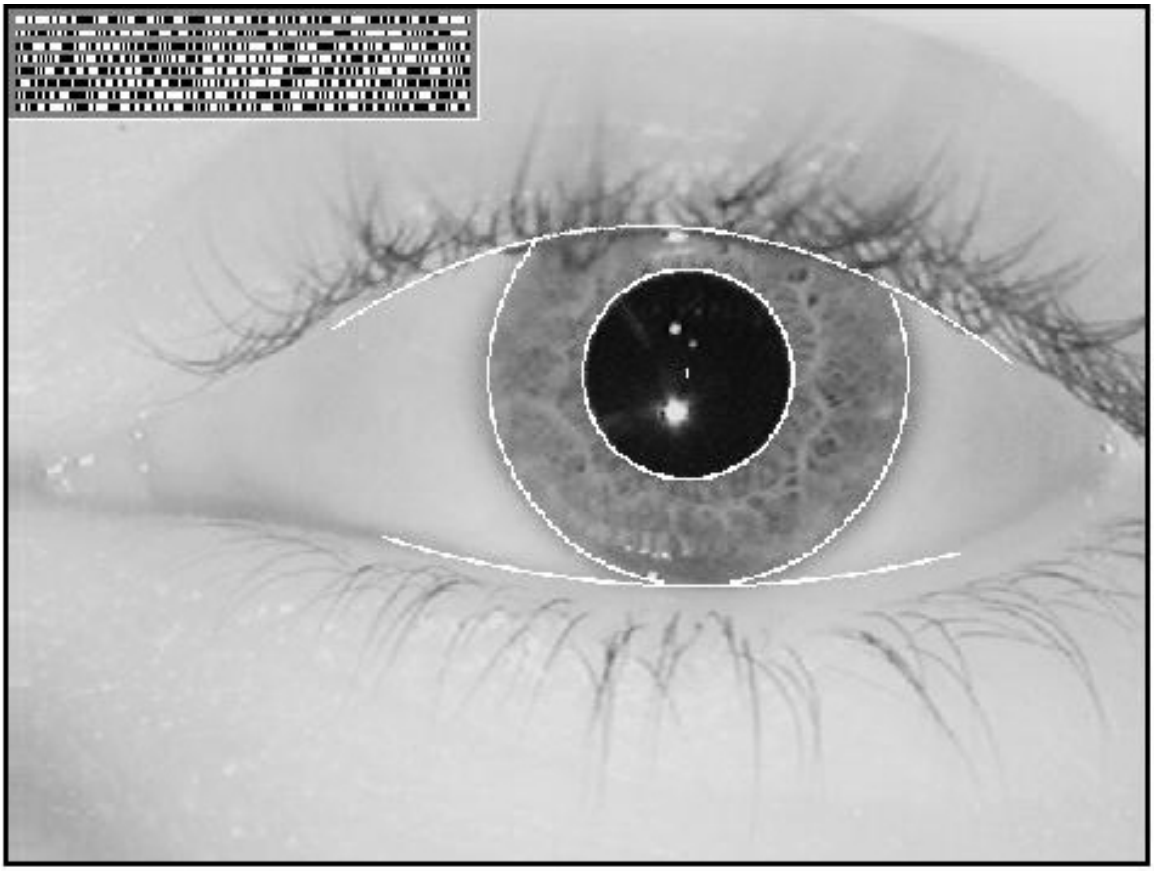


Figure 6.3: Iris Code by John Daugman check it out here[3]

Why, then, is iris recognition not more common? The answer seems simple: money. In order to implement iris recognition it is necessary to have a computer and an adjustable camera to accomodate people of differing stature. Obviously this is more costly on a large scale than encoded cards or memorized PIN's. However, as identity theft becomes more widespread and restricted access premises seek more powerful security solutions, iris recognition systems will become worth the cost.

[2] http://www.cl.cam.ac.uk/users/jgd1000/deployments.html
[3] http://www.cl.cam.ac.uk/users/jgd1000/iris_recognition.html

Chapter 7

Iris Recognition: The I Team[1]

7.1 Bryan Lipinski

Figure 7.1: Bryan Lipinski at work

[1] This content is available online at <http://cnx.org/content/m12496/1.2/>.

Bryan is a Jones College senior at Rice University. As far as the project, he had his hand in almost all the programming but mainly detecting the iris. Bryan loves to write computer code and calculate Fast Fourier Transforms as evident by his double major of computer science and electrical engineering. When he is not at his computer, he loves to hit the slopes back at home in Colorado. An expert snowboarder, he is proud of his most recent board purchase the Ice Slider 4000. After he graduates in the fall of 2004, he is headed to the rainy coast of Washington to fullfill a life-long dream of working for Bill Gates.

7.2 David Carr

Figure 7.2: David working on HAM radio

The youngest group member at sophomore status is David Carr. Normally a hardware man, David worked on coding up the Gabor filter as well as the actual iris code generation–both daunting tasks in which he succeeded. In his spare time, David enjoys skateboarding with fellow group member, Paul. In addition, he has particular interest in HAM radios. David can be found most of the time in the lab or in his room either soldering or programing on his beloved Unix machine.

7.3 Dmitry Khabashesku

Figure 7.3: Dmitry working hard

Mr. Khabashesku is a junior electrical engineering student at Rice University. His project responsibilities consisted of finding the pupil and unwrapping the iris into cartesian coordinates. A difficult task he completed quite well. Mr. Khabashesku's first love is electronic music. In his spare time he tries to complete his project of building an analog synthesizer. A transfer from UT, he has enjoyed the smaller more personal campus of Rice. Here he has found many people like himself in the dense gaming community around school.

7.4 Paul Robichaux

Figure 7.4: Paul Robichaux confused

Paul Robichaux is a 5th year electrical engineering senior at Rice University. Most of his efforts were focused on managing the team and heading up the poster presentation for the project as well as the connexions write-up. Paul hails from Lake Charles, Louisiana but was a transfer from Huntsville, Alabama. He enjoys reading various science journals in his spare time in addition to skateboarding at the few skate parks offered by Houston. Paul hopes to graduate with a BS in electical engineering in the fall of 2005.

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Module: "Iris Recognition: The I Team"
By: Paul Robichaux
URL: http://cnx.org/content/m12496/1.2/
Pages: 30-33

Iris Recognition
A project by Rice University students done in Fall 2004 for ELEC 301. An investigation of signal processing techniques applied to iris recognition.

About Connexions
Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.