18-551, Spring 2004

Group 7, Final Report

Private Eyes:
Biometric Identification
By Iris Recognition

Randy Attai (rra)
Matthew Brockmeyer (mjb2)
Darren Shultz (dshultz)
Frank Tompkins (fbt)

TABLE OF CONTENTS

PROBLEM AND SOLUTION

Security is of utmost importance in today's world. Specifically, we are concerned with the need to quickly and reliably determine whether a given person has the right to access some information or place. The field of biometrics addresses this issue by utilizing various physical or biological characteristics intrinsic to an individual in order to confirm that person's identity. One of the more recent and most accurate developments in this field is iris recognition.

John Daugman presents a compelling case for the use of the iris as a biometric identifier, the main points of which are summarized in the remainder of this section. The iris is suitable for quick identification because it is externally visible. It is also relatively insensitive to angle of illumination and changes in viewing angle because of its flat nature. The requirement that identification be quick rules out more time-consuming methods such as DNA testing and retinal scans. These methods also require greater cooperation from their subjects.

The iris is suitable for reliable identification because of its invariability and uniqueness. For pattern recognition to reliably classify objects, the variability between those objects (inter-class variability) must be greater than the variability between different instances of a particular object (intra-class variability). Face recognition, for example, is highly susceptible to this relation failing. An individual's face can appear quite different when time, lighting, angle, or emotional state change. In addition, all faces possess the same basic set of features, laid out in the same general way. Given this, the faces of related individuals, or even unrelated ones, can appear quite similar even under careful scrutiny. In comparison, the iris is very stable over time. It does not change with age and is well protected from damage and scarring, which can often foil fingerprint identification. Also, the transformations and distortions when imaging – even those caused by pupil dilation – are easily reversible. It has also been found that the intricate patterns irises contain show a great amount of variability among different people. In fact, irises are so unique that the left and right eyes of a single individual and the eyes of identical twins are easily distinguishable.

Given this suitability for use in identification, the next challenge is to capture an image of the iris. The iris can prove to be quite problematic to image. It is small (around 11 mm in diameter) and finely detailed. Detailed imaging requires the subject to be fairly close and, therefore, at least somewhat cooperative. Illumination often can have the undesirable side effect of causing glare in the image by reflecting off the eye's glistening surface. Further complications can arise such as blurring from movement or lack of focus; obstruction from eyelids, eyelashes, or contact lenses; and rotation of the head. In particular, rotation of the iris image has the potential to cause great intra-class variability (i.e. the same iris can appear different in different images).

We have implemented a practical system for the identification of irises based on the Discrete Cosine Transform (DCT) of the matrix representing their luminance. This system was tested on a database of 263 irises with mixed, but generally encouraging, results. The remainder of this paper will introduce you to the work that has been done, provide an overview of our system, and discuss the results we have obtained, along with the shortcomings of our solution, and the improvements over systems devised in previous years of 18-551.

PRIOR WORK

JOHN DAUGMAN, OBE

Virtually all commercially available and publicly operational iris recognition systems in use today are based on the algorithms developed at Cambridge University by John Daugman. He holds U.S. patent number 5,291,560 for "Biometric Personal Identification System Based on Iris Analysis." His research has been licensed to many companies such as IBM, British Telecom, US Sandia Labs, Iridian Technologies, IrisGuard, Inc., Securimetrics, Inc., Panasonic, IrisAccess LG Corp, IrisPass OKI Electric Industries, and EyeTicket Corp. USA. His localization and extraction methods take advantage of the eyes' circular features to detect the desired edges. The image is then mapped onto a dimensionless polar coordinate system, and phase information is extracted from each isolated iris pattern using quadrature 2D Gabor wavelets. Hamming distances between these phase bit streams can then be used to determine similarity between irises.

PREVIOUS 18-551 GROUPS

Two groups in the past years of 551 have attempted iris recognition projects. Both followed methods designed around the work of John Daugman. The first group, in the spring of 1999 (group 3), ran into trouble trying to build their own database using a Logitech QuickCam VC with manual focus, and were never able to get code running on the EVM. The rest of their process was very similar to the second group, which will be described in slightly more detail.

The second group did their iris recognition project in spring of 2003 (group 6). They successfully implemented many of Daugman's published techniques, circular edge detection, Gabor filters for feature extraction and hamming distance for classification. Their database consisted of 48 images, 48 classes with only one image in each. All images had the subject's eyes held wide open, as there were no obstructions from eyelids or eyelashes. They experienced significant trouble with the reflections from the flash inside the pupil.

Using Daugman's circular edge detector algorithm for the iris/sclera boundary, but a fake center and fixed radius for the pupil/iris boundary, Group 6-2003 started off with many assumptions. This was to account for their problems with the flash reflections in the pupil, the center was picked as the iris/sclera center and the radius for every eye was fixed to 50 pixels (not generally true of the eyes in either case, but accurate enough). The group also assumed the eye was close to centered in each image. Group 6-2003 was able to get code working on the EVM, however, their main shortcoming was their database.

Group 6-2003's training set was 15 images, and they tested on 144 images (48 original images * 3: original, blurred, airbrushed noise). Only 45 (3*15) of these should have matched, and 15 of those were exact copies of the image in the training set. This process didn't really make sense as 99 images (69%) were "imposter" images, and all the accepted images were either exact duplicates of the training set, or those images artificially altered by a paint program. This group also noted that a 5-degree rotation in angle of their images was enough to cause failure.

OUR IMPROVEMENTS

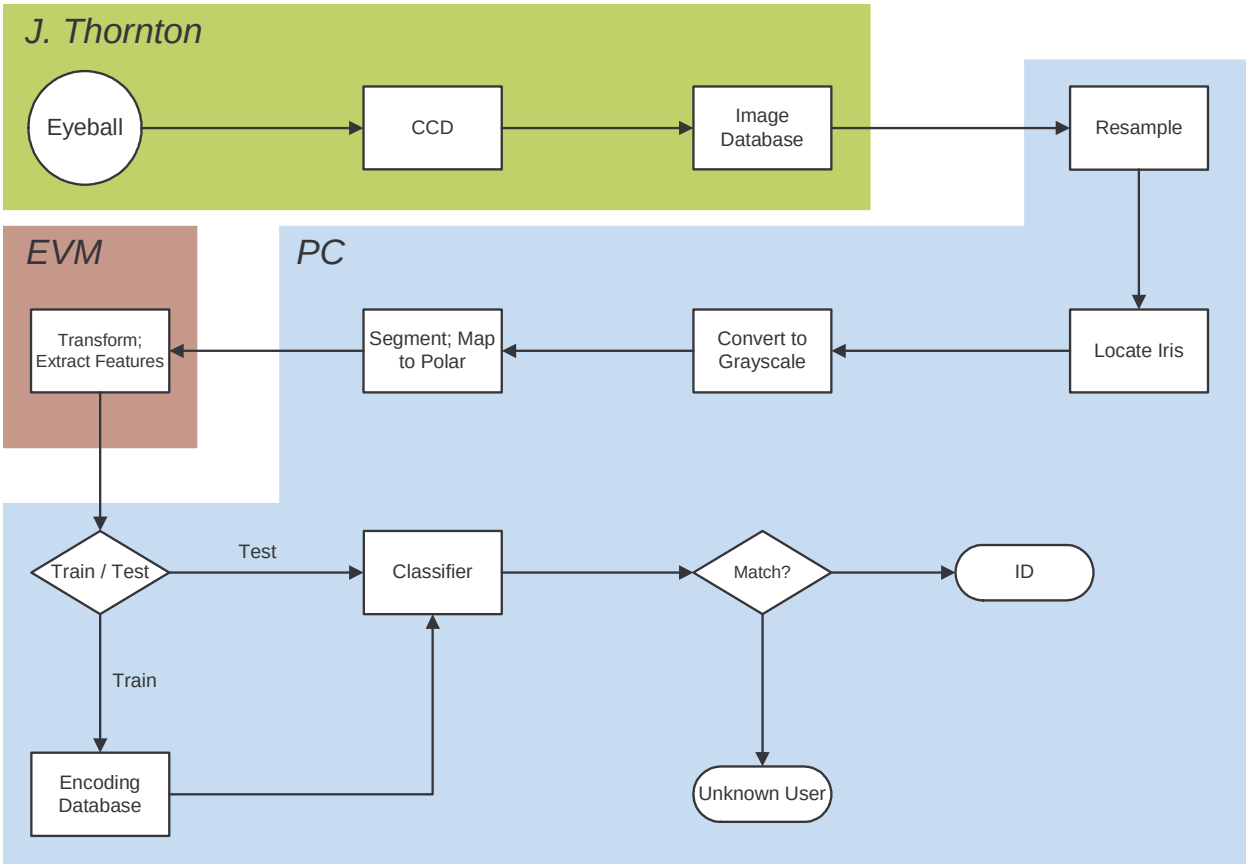
In comparison to the two past groups we have the following improvements:

- Implementation of all equivalent steps of past groups without EVM trouble.
- Our iris location makes no assumptions and has no hacks to guess at edge location.
- Flash reflections are not problematic
- Database is larger and more detailed (higher resolution).
- More advanced classifier
- Logical testing process

OVERVIEW AND DATAFLOW

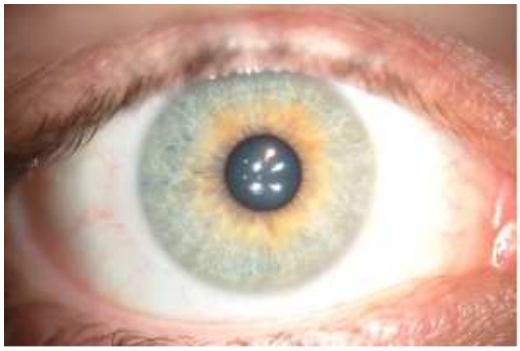
DATAFLOW GRAPH

First, a digital camera takes several pictures of both eyes of many people. These large images are stored in JPEG format in a database (provided by Jason Thornton), and then each image is resized to 300x200 pixels. These smaller images are ready to be processed by our iris recognition program, which is described in the subsequent section. The training set, which consists of six images each from twenty eyes (120 total images), is generated by running the recognition program on each image and storing the output DCT in a database. To test an eye in the test set, that eye's image is sent to the recognition program, and the resulting DCT is compared to the DCTs of every image in the training set. The k-best classifier then decides which individual in the training set owns the test eye or declares the test eye an imposter.

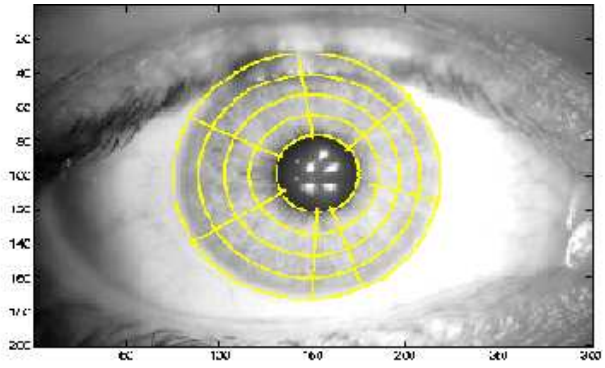


WALKTHROUGH

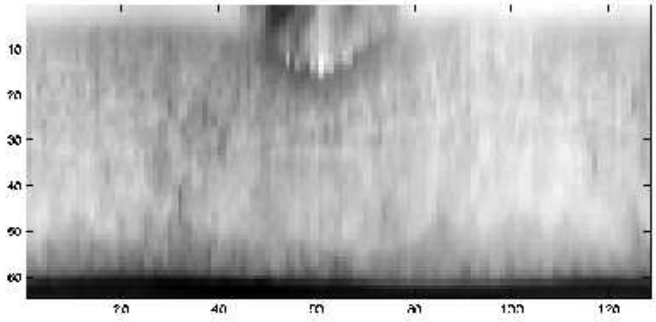
The 300x200 color image of the eye is used to find the circular edges of the iris. The color image is then converted to grayscale according to the NTSC standard and passed to the polar mapping code. This code essentially divides the iris into sections by making radial lines and concentric circles. It then makes a radial cut in the bottom portion of the eye and maps the roughly annular iris to a rectangular region. Next, the unwrapped iris image is sent to the EVM and its discrete cosine transform is calculated. The DCT array is then sent back to the PC for classification.



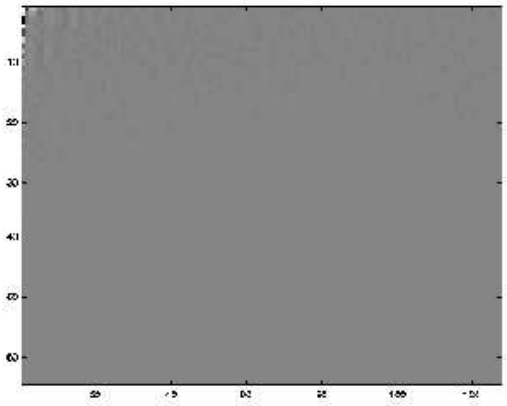
300x200 color image



Segmented iris



128x64 unwrapped iris



DCT of unwrapped iris

EVM IMPLEMENTATION

The primary speed issue we found in implementing the Discrete Cosine Transform on the EVM was that the cosine functions, which we originally had placed in the innermost loop of our program, uses a function call on the EVM, which prevents Code Composer's optimization algorithms from unrolling the loop in which they were present. Additionally, it is extremely wasteful to calculate these values in the innermost loop, as for each pair of values of the outermost loops, every value in the matrix will be multiplied by the same two cosine vectors (one horizontal and one vertical) evaluated at different points. Thus, by precalculating the horizontal and vertical vectors in the outermost two loops respectively, we reduce the number of cosine values that must be calculated by a factor of 128. This additionally permits code composer to unwrap the innermost loop, providing the maximum amount of parallelism in the multiplications and adds contained therein.

Another optimization involved the moving of several divisions that would be applied identically to each pixel in the image outside the two inner loops, applying it instead to the sum calculated from all the pixels in the image multiplied with the appropriate cosine values. This reduced the number of divisions per iteration through the innermost two loops from 8192 to one, providing an obvious and drastic reduction in computation efficiency. Unfortunately, once we got to this point it was impossible for us to further reduce the innermost loop, as we had not discovered any fast DCT algorithms at that point, and had not put much effort into finding them as we felt the average user of our system would not consider a delay of about 4 seconds to be unreasonable until Professor Casasent informed us that this was unacceptable. As this innermost loop consumed the vast majority of computation time, any further optimizations outside of this loop would have had a trivial effect.

Given that the algorithm made extensive use of the values of each pixel in the input image, we chose to cache the entire image in on-chip memory. This was possible due to the fact that 128x64 floating point numbers only took up 32KB of memory, and allowed for a great increase in the speed of our code. Additionally, one line of output was paged at a time, and then transmitted asynchronously by DMA to off-chip memory while the next line of output was being calculated. The two lines of output that were paged took up a total of 1KB of the on-chip memory, while our code took up 2KB. The DCT output took up a total of 32KB of external memory.

Inner Loop Incl. Average	
Before	1362
After	159
This is executed 64*64*128 times!	

Operation	**Total cycles**
Processing function	96,670
Without overhead	94,154
Innermost Loop	63,833

The Database

Jason Thornton, a doctoral student in ECE at CMU provided us with a starting database of 263 iris images. The database consisted of 23 classes, each of which had between 6 and 13 images; 82% of the classes had between 11-13 images, inclusive. A 'class' is defined as one unique eye; a person's left and right eye would be 2 separate classes. The images were acquired using a high-resolution digital camera. The participant would position her eye for the camera by resting her chin on a chin-rest and placing her forehead against a molded plastic bar. All pictures were taken with four flashes and from a distance of about 8-24inches. The pictures for each class were taken over a period of approximately 30 seconds to 3 minutes and participants were told they could shift around a bit and "be natural", it was not required that the eyelids be open as much as possible, and often the camera was re-focused between each picture. Within the 263 images there were several blurry pictures, and a few "blinks". We did not remove these images, but instead kept them for testing.

Test and Training sets

For our purposes, these images were divided into a training set and a test set. The basic heuristic used for separation was to simply place every other image in a different set, with exactly six images going to the training set for each class, however we attempted to keep any cases with exceptional occlusion out of the training set whenever possible. Some classes were reserved as imposters, and hence had all of their images placed in the test set; these consisted of one person's left eye which had no corresponding right eye in the database, and two eyes (one light colored and one dark colored) which had their corresponding "partner" from the same person in the training set. In the final implementation, all unnecessary information was stripped out of the training set (i.e. everything except the DCT coefficients we had calculated for each image).

ALGORITHMS

Preprocessing

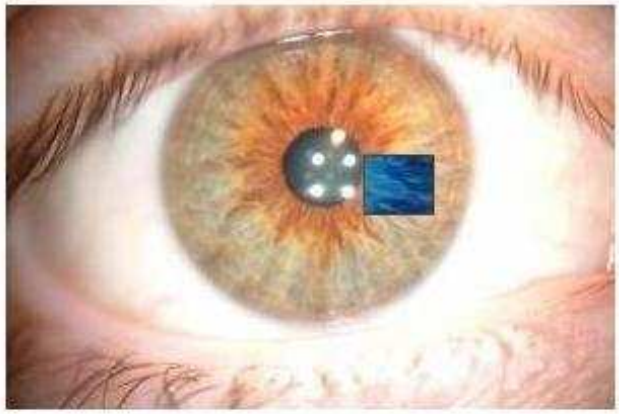
The database images we started with were originally 2848 by 4256 pixel jpeg files. This large size would have made much of the processing prohibitively slow, and the level of detail provided was not required for accurate results, as the patterns of luminance present in a lower resolution image are sufficient for discriminating between 2 different irises. We decided to resize our images to 200 x 300 pixels (or 7.05% their original size), which seemed to provide a good trade off between file size and level of detail. In hindsight, 400x600 images may have provided a slight advantage in discrimination, but the difference was negligible during our early empirical testing in MATLAB. This *preprocessing* was done in a MATLAB ".m" file as a batch process on all 263 images. The bulk of the preprocessing was simply dealing with folders and file names; the actual process for each image could be reduced to the following MATLAB commands:

```
eye = imread(path);        %example path: C:\MATLAB\work\image1.jpeg
eye = imresize(eye,0.0705);
eyeR = eye(:,:,1); eyeG = eye(:,:,2); eyeB = eye(:,:,3);
save(filename, 'eyeR','eyeG','eyeB', '-ascii');
```

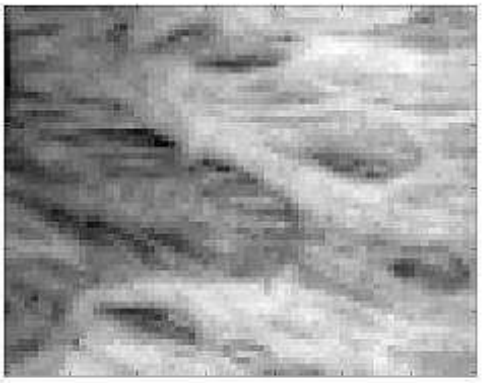
The first line reads in the jpeg image in as a 2848x4256x3 array, the second resizes it to a 200x300x3 array, while the third breaks the 3-D array into 3 separate 2-D arrays and the final line saves the file as a 2.74MB ASCII file (a concatenation of the 3, 2-D arrays). MATLAB used nearest neighbor interpolation to resize the images.

The preprocessing took about 20minutes, with over 95% of the time spent on reading in the files. The ASCII files contained 180,000 floats stored as ASCII, using 16 bytes each, with values between 0-255. This was due to the way MATLAB saves files and a rather large waste of space, the current preprocessed database was now 25% larger than the original jpeg files.

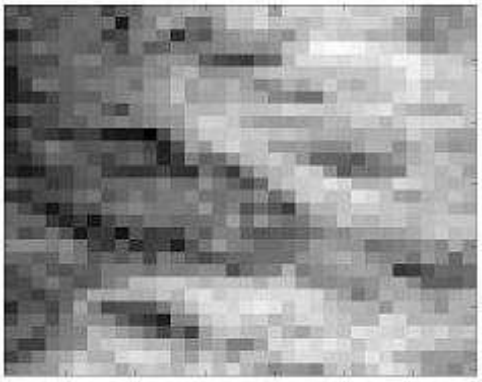
As this text file full of floats was using 16 characters to store each float, thereby creating the aforementioned waste, we then used a specially designed batch file and executable to convert them to a binary representation of a floating point number. This resulted in a 75% decrease in file size, with no loss of information. In hindsight, as the floats happened to be representing integers from 0-255, we could have decreased the file size an additional 75% (total of 93.25% reduction in size) by instead storing them in binary as chars, but the reduction actually obtained was more than sufficient for our needs.



This image was converted to the various sizes, and the shaded section was enlarged to the right to show the differences in detail they provide.



The level of detail provided by a 600x400 image is nearly indistinguishable from the original



The level of detail from a 300x200 image is markedly lower, but still shows the outlines of the major features

IRIS LOCATION (CIRCULAR EDGE LOCATION)

Two boundaries must be located within the image of the eye: (1) the pupil/iris (inner) boundary, and (2) the iris/sclera (outer) boundary. We assume that both boundaries are circular and that the center of the inner is farther than 48 pixels from any edge of the image (keep in mind that the images we are using are 300x200 pixels). The former assumption is intended to simplify the algorithm, while the second was chosen to speed up the algorithm. Although the boundaries are elliptical, we found that in most cases they have very low eccentricity (less than 0.1), so our first assumption is justified. We justified the second one by noting that no image in the database we used was in violation and that if a significant portion of the iris were missing from the image, recognition would be impossible.

Our edge location algorithm is based on the circular edge detector described by Daugman[1]. The technique is based on the following equation:

$$\max_{r, x_0, y_0} \left(G(r) * \partial_r \int_{r, x_0, y_0} \frac{I(x, y)}{2\pi r} \, ds \right)^2$$

The integral is a closed circular path integral along a contour lying in image space; **I(x,y)** represents the image as a two dimensional function in Euclidean space where each point (with integral coordinates) represents a pixel in the image. Roughly, one computes a (discrete) contour integral with radius **r** and center (x_0, y_0) by summing the intensities of each pixel along the integration path. The partial derivative of this value with respect to **r** is taken in order to find the radius at which the change in the integral's value is maximal. This derivative is then "blurred" through convolution by a Gaussian **Γ**. As this is an iterative algorithm with three parameters (**r**, **x0**, and **y0**), it is computationally intensive. Of course, the equation must be suitably discretized for implementation on a computer (the integral becomes a summation, etc.).

The first step in our edge location algorithm is a modification of Daugman's technique in which color information is used both to accentuate the iridian boundaries and to deemphasize the camera glare generally present inside the pupil. The idea to use color information came to us when we realized that the sclera is white and the pupil is almost black (both regions contain roughly equal amounts of RGB), whereas the iris has a greater disparity in the amounts of red, green and blue color present. Thus we chose to create two highly skewed intensity map images, derived by mixing the RGB color planes in certain ways, on which to run the iterative circular edge detector. We used the following scheme to create two grayscale images, one for the inner boundary detector and one for the outer boundary detector:

$$I_{inner}(x, y) = 3R(x, y) - G(x, y) - B(x, y)$$

$$I_{outer}(x, y) = \exp\left(-\frac{1}{R(x, y)\, G(x, y)\, B(x, y)}\right)$$

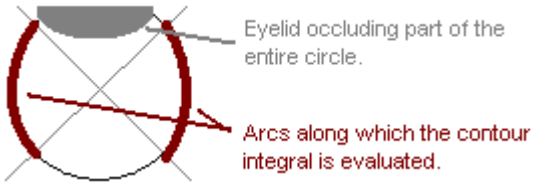
Where I_{inner} is the image used to find the inner edge and I_{outer} is the image used to find the outer edge. **R**, **G**, and **B** are the color planes for the original image.

The expression for I_{inner} was chosen because most eyes contain more green or blue than red and the glare from the camera flash contains essentially no red. The expression for I_{outer} was designed to accentuate regions in which R, G, and B are all high and to deemphasize regions in which at least one of R, G, and B is low. I_{inner} and I_{outer} were both normalized so that the pixel intensities lie between 0 and 1 prior to edge detection, although this is not strictly a necessary step.

To find the inner boundary, we actually forgo the Gaussian smoothing and require the radius of any contour to be at least four pixels (the smallest pupils in the database have radii on the order of 10); this seems to prevent the detector from being distracted by the camera glare. Once the inner edge has been located, a box of width and height eight fifths times the inner circle's radius and centered at the inner circle's center is searched for the center of the outer edge; that is to say, (x_0, y_0) are chosen from within this box to find the outer edge. This assists the algorithm in finding the outer edge and also saves greatly on computation time. We do not sacrifice accuracy this way since the center of the outer edge must lie within the inner circle.

The optimized edge location code (for both edges) runs in about ¼ second on the lab PCs. The primary optimization was using trigonometric lookup tables in the innermost loops. In order to compute the circular contour integral, we pick sixteen rational points along the integration path and use the integral parts to index into the image intensity array (either I_{inner} or I_{outer}).

These points are only selected from the left and right quarters of the contour in order to preclude the possibility of eyelid occlusion from skewing the edge detector results.



Once the boundaries have been located, the information is passed to the polar mapping routine in the form of six quantities: inner radius, outer radius, and the x and y coordinates of the centers of both circular edges.

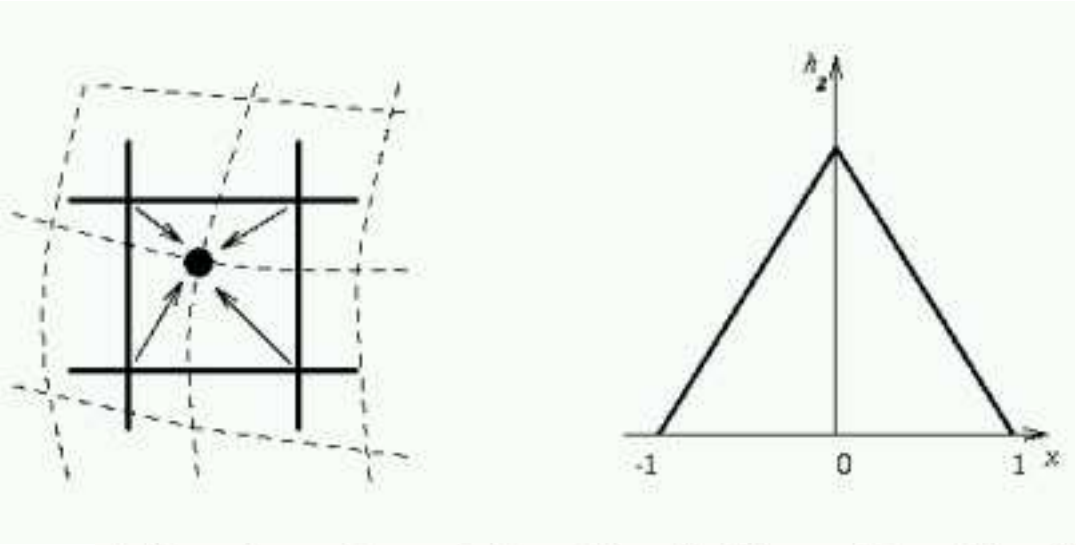
SEGMENTATION AND MAPPING TO DIMENSIONLESS POLAR COORDINATES

Jason Thornton also provided us with code for a polar transform in a MATLAB ".m" file. The polar transform normalizes for radius between the pupil/iris and iris/sclera boundaries. This is used to make the processing invariant to the natural fluctuations in size of the pupil of a living person, as the process uses the same number of radial divisions regardless of the actual dilation of the pupil. We ported this MATLAB code directly to C, as it was highly effective and did not appear to require much alteration. The effect of the code is discussed below.

For each angle from 0 to 2*pi (the number of angles is determined by an angular resolution argument; 128 in our case), we can imagine drawing a line from the center of the iris, radially outward. The space between the inner and outer radii along that line (i.e. the segment of the line which lies within the iris itself) is then divided into several equal segments (the number of segments is determined by a radial resolution argument; 64 in our case). The coordinates of points which are thus determined are the points which will be used to build the unwrapped image of the iris. Note that this process should, in theory, provide roughly the same image, regardless of the dilation of the pupil, as the action of the iris can be modeled as simple elastic deformation.

The vast majority of the coordinates thus calculated will not fall on integral values; rather they will fall "between" actual pixels in the original image. In order to determine the luminance values these "pseudo-pixels" should have in the unwrapped image, it is necessary to apply some sort of interpolation algorithm to the surrounding pixels.

We decided to utilize the same method of interpolation utilized by Thornton's Matlab function, which was bilinear interpolation. Bilinear interpolation uses a weighted average of the four real pixels surrounding the pseudo-pixel to determine the luminance value for the pseudo-pixel (see image below). This results in less loss of data than nearest-neighbor interpolation, at the cost of potentially reduced contrast. We adopted this primarily in keeping with the intent of directly porting Thornton's Matlab, however as will be discussed momentarily it does seem to be the best tool for the job. It may have been better to perform some sort of averaging over the entire region each pseudo-pixel represents in the original image, however the resolution of our polar mapping is sufficiently close to the resolution of the 300x200 image that the improvement this would yield is negligible.



$$f_2(x,y) = (1-a)(1-b)g_s(l,k) + a(1-b)g_s(l+1,k)$$
$$(1-a)bg_s(l,k+1) + abg_s(l+1,k+1)$$

where

$$l = \text{floor}(x), \quad a = x - l$$

$$k = \text{floor}(y), \quad b = y - k$$

[2]

Other interpolation algorithms, which were considered but rejected, include nearest neighbor interpolation – in which the pixel having the minimum Euclidean distance from the pseudo-pixel is used – and bi-cubic interpolation – in which the 16 nearest pixels are used to determine a value for the pseudo pixel. Nearest neighbor interpolation was rejected as it creates a noticeable loss of detail in the image. Bi-cubic interpolation was rejected due to its increased computational complexity and negligible (for our purposes) impact on the resulting image quality relative to bilinear interpolation.

FEATURE EXTRACTION (DISCRETE COSINE TRANSFORM)

We investigated several possible methods for obtaining features, including various wavelet analyses, correlation, FFT, and DCT. We decided against using wavelets because we did not desire to copy verbatim the methods of Daugman and the previous two 18-551 groups, and we were unable to figure out how to garner useful information from other wavelets as most of the literature we found seemed to focus on using wavelets for image compression. Having quickly surmised that correlation was a one-way ticket to nowhere, we were left with FFT and DCT. Although the magnitude of the FFT is translation invariant (and thus would provide full rotation invariance in our case – keep in mind the unwrapping process), we decided to use the DCT instead for several reasons. First and foremost, our early experiments computing frequency domain distances between unwrapped irises in MATLAB indicated empirically that the DCT was superior. Second, the literature seems to indicate that the DCT is well suited for photographic image processing due to its close approximation of the KLT[3]. Finally, we supposed that since DCT is essentially the real part of an FFT, it should not really matter which we chose to use anyway.

Once the 128x64 pixel unwrapped iris image is available, it is sent to the EVM to have its discrete cosine transform computed. The transformed 128x64 pixel image is then interpreted as a description of the frequency content of the unwrapped iris image. One could consider the feature vector (or feature matrix) we use to be the 128 x 64 = 8192 DCT coefficients present in the transformed image. The 2-D DCT equation is given by:

$$S(u,v) = \frac{C(u)\,C(v)}{\sqrt{64 * 128 / 4}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s(x,y) \cos\frac{(2x+1)\pi u}{256} \cos\frac{(2y+1)\pi v}{128}$$

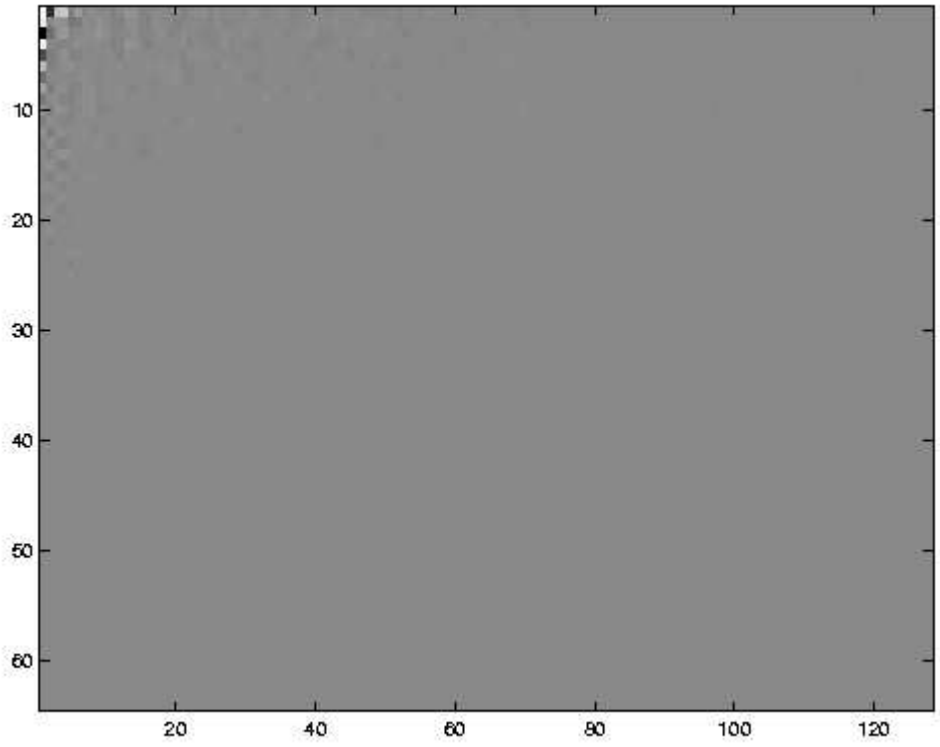
$$\text{Where} \quad C(t) = \begin{cases} \frac{1}{\sqrt{2}} & u = 0 \\ 1 & u > 0 \end{cases}$$

s(x,y) is the unwrapped iris image, and **S(u,v)** is the transformed image. The 8192 numbers in **S** are then the features that we have extracted.

We chose not to use a block DCT because this would remove any rotational tolerance we have as a result of computing the DCT of the entire 128x64 unwrapped image. Since unwrapped images of the same eye should generally differ by horizontal shifts only (ignoring eyelid occlusion) the first column in the DCT array should be roughly identical for all images of the same eye (ignoring small variations in intensity, or, equivalently, assuming normalization). One can see this by examining the DCT equation above and taking **v = 0**. The second column will be slightly different due to mixing with the horizontal frequencies (the rows of the DCT), and the third column will differ to a greater extent, and so on. We found that most of the energy (over 99%) is contained in the first several columns, so these columns dominate in the distance computation. Thus, the fact that the first few columns change relatively little as a result of horizontal shifts in the unwrapped image, our DCT algorithm has some degree of rotational invariance. For the threshold value used in our iris recognition program, we found that irises in the database would be correctly recognized upon rotation by 15 to 55 degrees in either direction (total of 30 to 110 degrees of rotational invariance). Increasing the threshold would of course increase the degree of rotational invariance, but might also increase the number of false

positives. We felt that this was unnecessary given that 30 degrees of rotational freedom should be plenty for any real iris recognition system.

The 128x64 array of DCT coefficients (the feature matrix) output by the EVM has low frequencies in the upper-left corner (small coordinates) and high frequencies in the lower-right corner (large coordinates). Most of the energy is located at low frequencies, as can be seen below in a typical DCT.



A sample DCT.

K-BEST CLASSIFICATION

Originally, we had plans to use a Bayesian classifier for the identification process. The probabilities provided by such a classifier would have been useful in outputting confidence levels for the user. However, our relatively small training set (six images for each class) wouldn't have allowed us to implement a very effective Bayesian classifier. So, the classifier we used basically implements a k-nearest neighbor algorithm with some variation. This allows us to perform relatively accurate classification in spite of our small training set while still taking advantage of its greater-than-one size for each class. A previous 551 group, Group 6 from 2003, could only implement a 1-NN algorithm because of their one-per-class training set.

In our classifier, the distance is computed between the DCT coefficients of the test image and every image in the training set. Distance is computed by summing the squares of the differences of the coefficients in the DCT matrices (i.e. a Euclidean metric is used). The k nearest training images are stored along with their corresponding class information. We empirically determined the best value for k to be three. The list of k nearest training images is then used in a 1-NN algorithm with the addition of a low threshold. The optimal low threshold was determined to be 20. This low threshold (along with the nearest neighbor information) essentially forms "tight" disjoint subspaces, one around each class' training set. If the test image falls within one of these tight subspaces, it was immediately classified into the corresponding class. We included this nearest neighbor search into the k-NN algorithm because we found that very close distances, even to outliers, often indicate a match. This could come, for example, from the diversity of our training set and the resulting comparison of occluded images. An occluded test image found to be very close to a training image usually results only if that training image is similarly occluded and the test and training images are members of the same class.

If no match was found in the 1-NN algorithm, the list is then used in the traditional k-NN algorithm with another, higher threshold. The optimal number for this threshold was determined to be 70. Occurrences of a class within the list are only counted if the associated distance falls below the threshold. The class most represented in the k-NN list is then determined to be the identity of the test image. Ties are awarded to the class that has the single nearest neighbor. Once again, the combination of threshold and nearest neighbor information forms disjoint subspaces, this time with geometrical boundaries determined by k-NN rather than 1-NN rules. A test image found to be outside all training set subspaces has no match and is classified as an unknown user.

SOFTWARE

The only code we used that we did not write ourselves was the MATLAB routine that segmented the iris and mapped it to doubly dimensionless polar coordinates. This code was provided by Jason Thornton. This MATLAB code was utilized for testing purposes only; it is not used in the final iris recognition program. We ported the iris segmentation and polar mapping code from MATLAB to C in order to incorporate it directly into our iris recognition routine. This involved writing C versions of several MATLAB functions, including **repmat** and **interp2** (bilinear interpolation). This porting process was described above in the algorithms section.

We wrote MATLAB preprocessing code to shrink the large images in Jason Thornton's database to a more reasonable size (300x200) and to save the smaller images in binary RGB format for ease of processing. In essence, this code creates a new database of 300x200 binary RGB images, one for each large image in Jason's database.

We coded the circular edge detection from scratch, using Daugman's description as a guide. We augmented his method by utilizing color information present in the eye images and redesigning and simplifying the iris/pupil (inner) boundary calculation in such a way that the bright spots resulting from the camera flash would not hinder the inner edge location process (this was described in the algorithms section above).

The DCT code was also written from scratch. We programmed the EVM to take the DCT using a quadruple for loop that directly computes the standard DCT formula (given in the algorithms section), reusing as many cosine factors as possible. This code takes about four seconds to run, which is far too slow given that significantly faster algorithms exist. Unfortunately, we were unable to find any fast DCT algorithms (that don't require division into 8x8 blocks) prior to the demo, but we did finally come across a paper which describes a fast DCT that works for any signal whose size is a power of two[4]. If we had time to implement it, this would probably make our entire iris recognition routine run in less than a second.

We also wrote a k-best classifier in C. Its operation is described above in the algorithms section.

ANALYSIS OF RESULTS

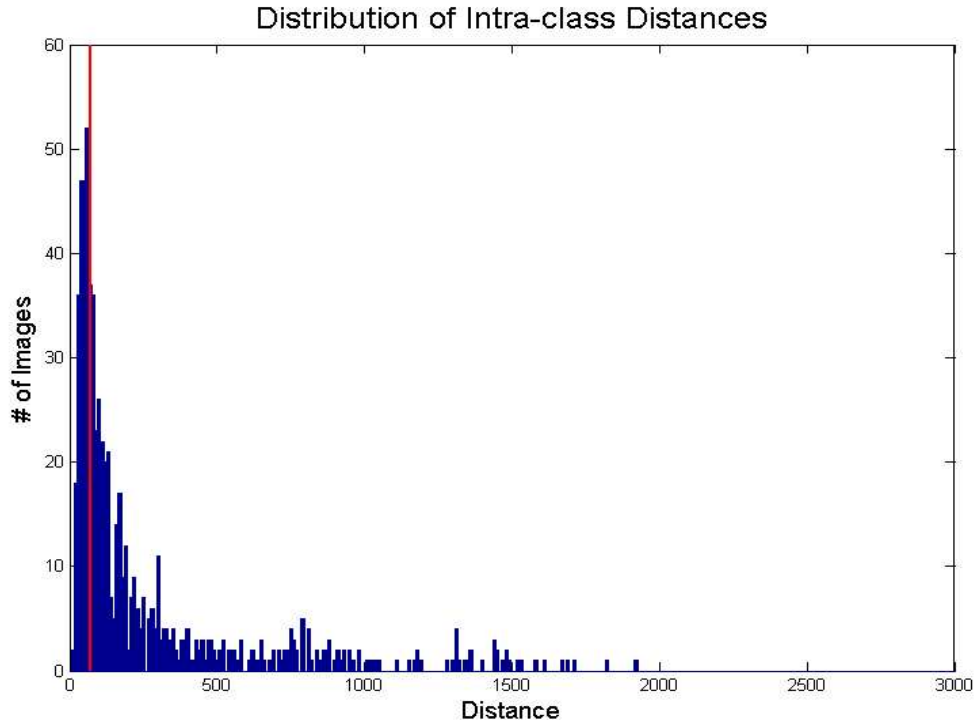
We tested 143 images. 114 of these images were from irises present in the training set (i.e. they should be accepted and identified). 29 of these were from imposters who are not represented in the training set (i.e. they should be rejected as unknown users). Of our 143 test runs: 69.23% gave the correct results, 26.57% were false negatives, and 4.2% were false positives.

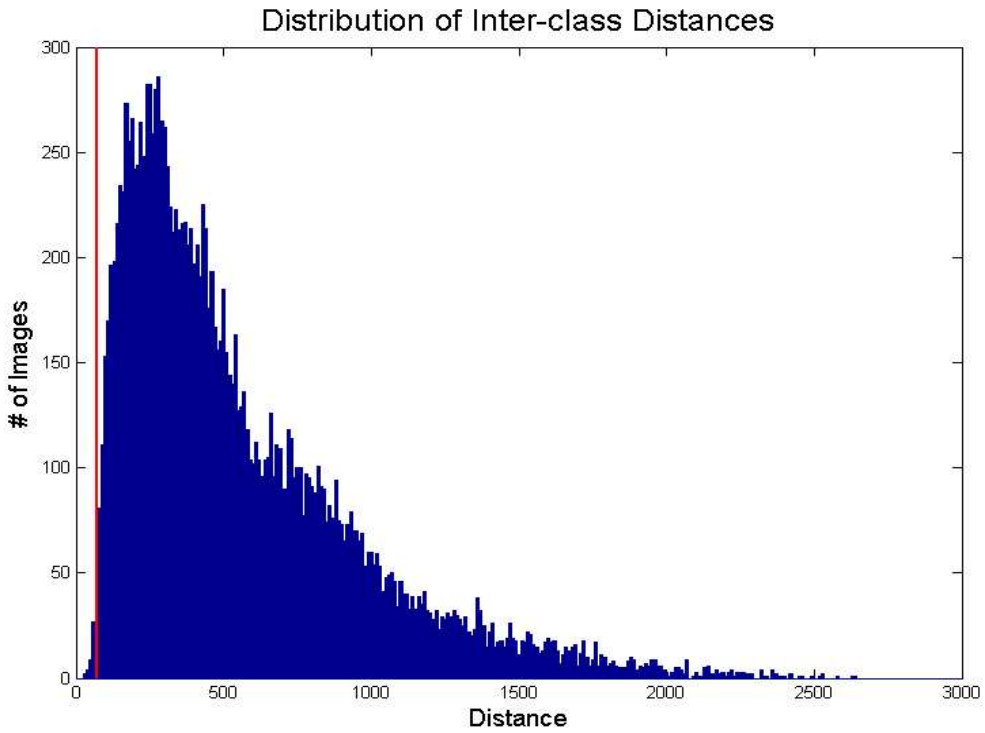
False negatives are usually considered to be acceptable annoyances in the name of security. In particular, a false negative turns out to be only a minor inconvenience when a subsequent re-test provides the desired accurate identification. Under such conditions, our system performs adequately since nobody was found to be completely unidentifiable. That is, for every individual in the test set, they were correctly identified at least once. Furthermore, the false negatives can be largely attributed to too much occlusion of the iris. The average amount of occlusion in the images – as estimated visually – that produced false negatives was 33%. This is well above the average amount of occlusion in the entire test set, which was only 18%. In a security scenario, this combination of identifiability and occlusion would be equivalent to the possibility of someone being rejected as an unknown user by their first photo and then being correctly identified by a subsequent photo in which they opened their eye wider.

From a security standpoint, false positives are, of course, the largest concerns. You wouldn't want to grant access to an unauthorized person. Our post-test analysis shows that all of the false positives can be eliminated by not testing left against right eyes or by a slight adjustment of the threshold used in the classifier. Two-thirds of our false positives were the result of a left eye being identified as that same individual's right eye or vice versa. In the real world, this would not be a concern. No system would treat an individual's left eye as a separate person from their right eye. The rest of the false positives could have been avoided if the threshold used in the classifier had been 67.5 instead of 70. Intuitively, making the threshold stricter to reduce false positives will increase the chance of a false negative. Our results show that changing the threshold from 70 to 67.5 would have resulted in only one additional false negative.

By looking at the distributions of intra-class and inter-class distances (see next page), we can see the appropriateness of our choice of threshold. Intra-class distances were calculated from the 114 test images compared against their 6 corresponding training set images. A histogram of these 684 distances shows a curve that is sharply decaying after a distance of about 70. It also shows us that a threshold of about 200 would significantly increase the classifiers ability to make matches and reduce the number of false negatives, since the area under the curve to the right of the line is essentially the "false negatives area." However, increasing the threshold in this way would greatly increase the number of false positives. We can see this on the distribution of inter-class distances. These distances were calculated from the 143 test images compared against all images in the training set known to be from a different iris. A histogram of these distances shows a curve that is sharply rising after a distance of about 70. The area under this curve to the left of the threshold is essentially the "false positives area." As mentioned earlier, minimizing this area is extremely desirable. The choice of 70 for the threshold (or 67.5 as we determined later) is roughly optimal for minimizing both the false negative and false positive areas. In the k-NN classifier, using a k greater than one also helps to reduce the

effective size of these undesirable areas by reducing the chance that improbable distances will determine the classification. If we had a larger database, we could have used a higher k and/or a lower threshold and potentially had better results.



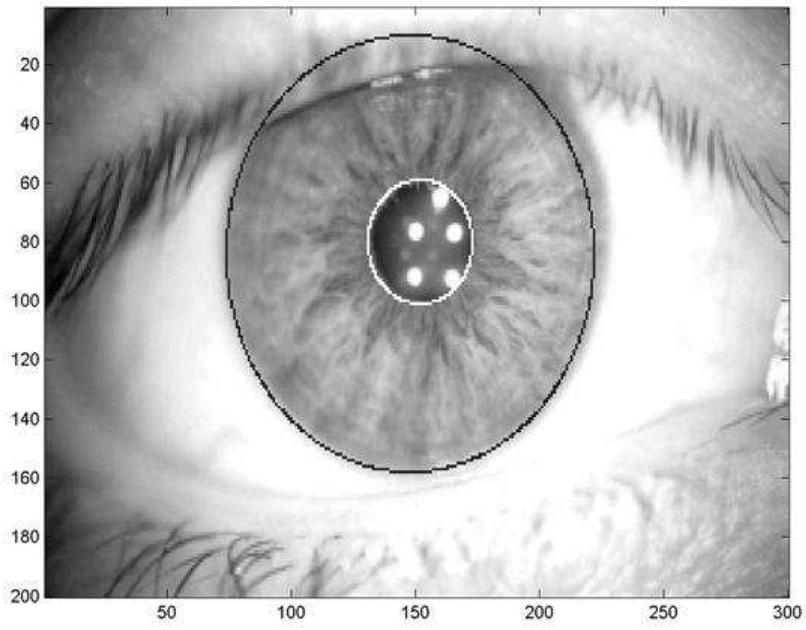


**Red line indicates threshold of 70.*

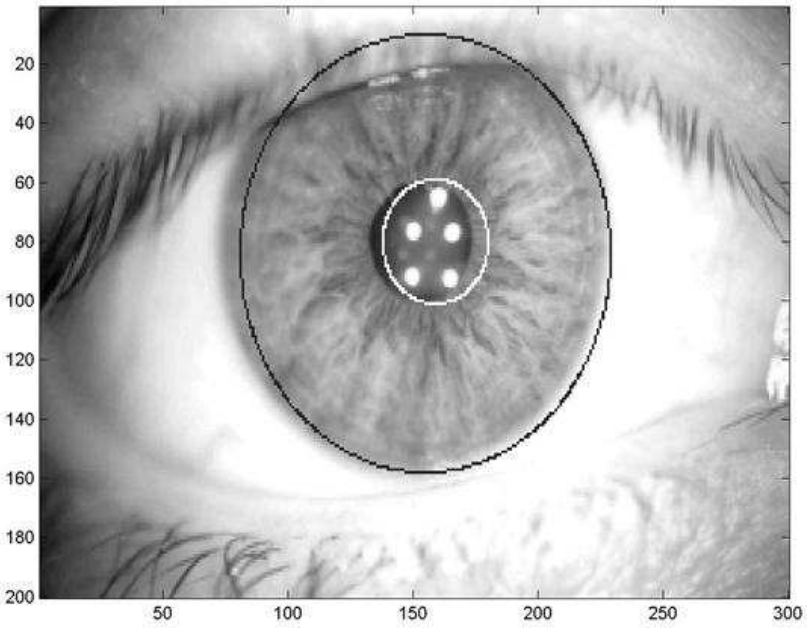
Importance of Proper Edge Detection

As suggested by Professor Casasent, we conducted some tests in order to determine how important the accuracy of our edge detection algorithm was in correctly determining the identity of an individual. In testing the sensitivity of the results to the output of the edge detection algorithm, we purposely altered the values of the center of the inner and outer boundaries as well as the size of the radii. We tested on 2 eyes, changing only one variable in each test to avoid compounded errors. First we moved the true center of both the inner and outer boundaries in the x direction and then the y. Since we are dealing with a circular object, it ended up not mattering terribly much which direction was chosen. In the first image, the program would fail to match after the centers had been "bumped" by 7 pixels (x_center+=7).

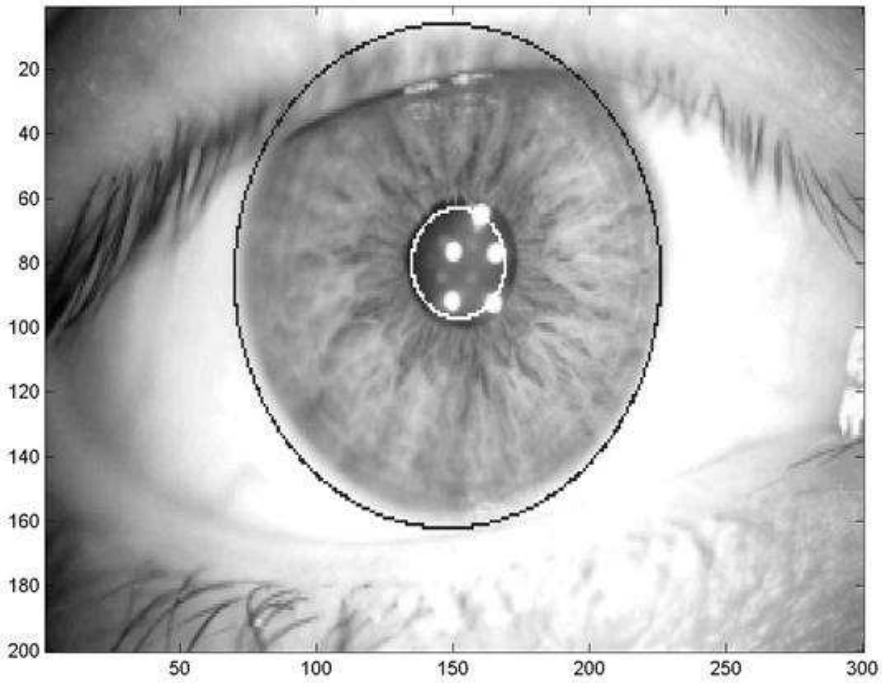
In the 2nd image, the program failed after the center was bumped 6 pixels. As seen in the images below, this can be seen to be a rather small error in edge detection that leads to false negatives. In terms of the size of each boundary (radius), the program would fail after we had moved the inner boundary in by 4 pixels and the outer out by 4 pixels in the first image, and by 5 and 5 in the 2nd. This would add spurious data, leading to greatly different frequency content and thus recognition failure.



An Illustration of The Actual Output of Our Edge Detection Algorithm



An Illustration of The Amount of Translation Needed to Produce a False Negative



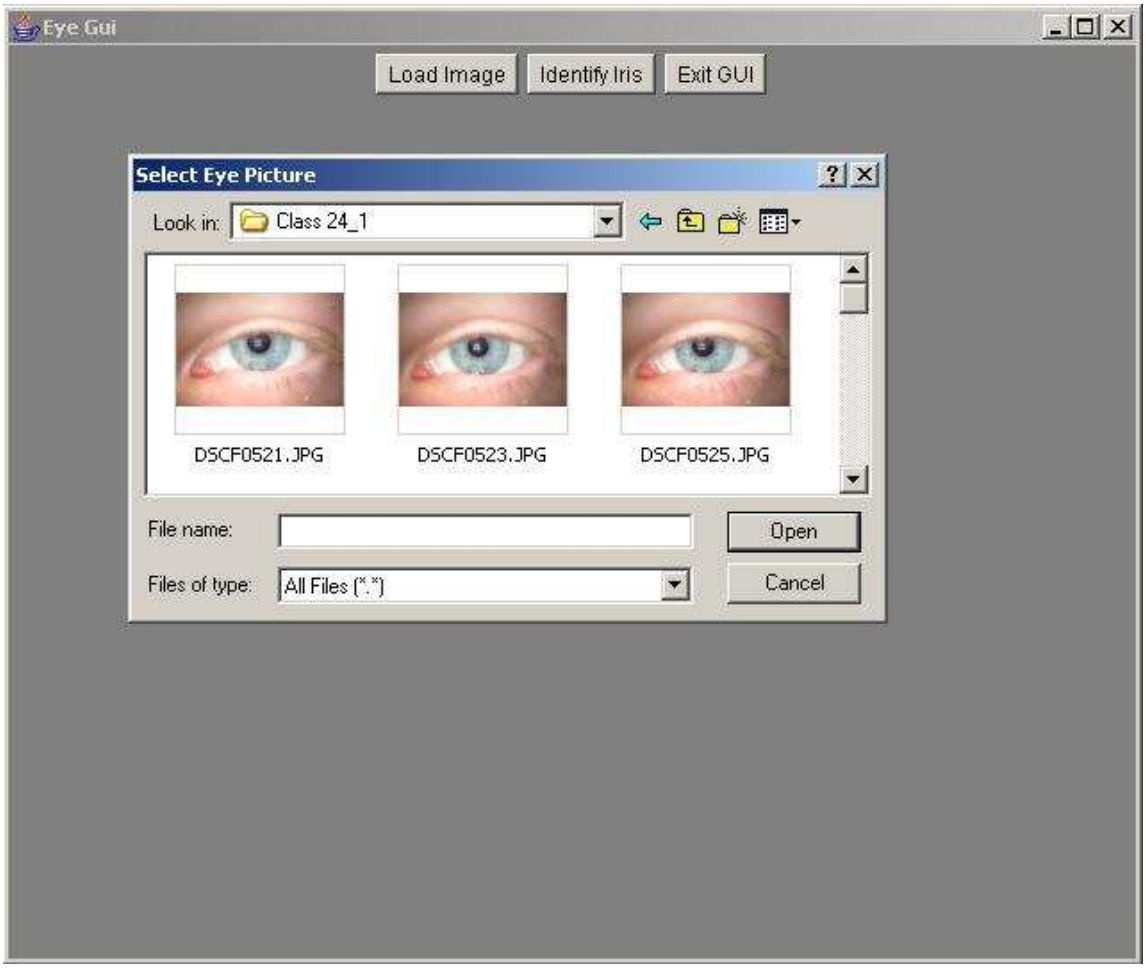
An Illustration of The Amount of Radial Distortion needed to Produce a False Negative

ANALYSIS OF FEATURE SELECTION

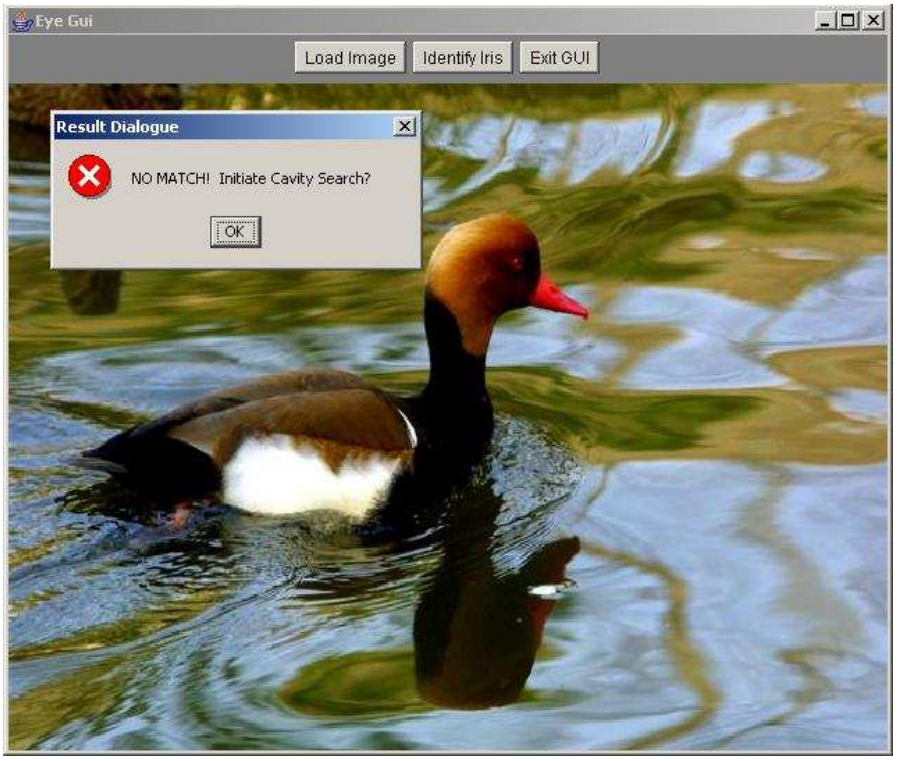
In an effort to determine the effectiveness of the feature set we chose, we tried computing the distance between DCTs while ignoring the DC offsets (the first DCT coefficients). This change caused reductions in in-class distance on the order of 10 and reductions in out-of-class distance on the order of 100. Needless to say, ignoring the DC offset made iris recognition much more difficult for us. Ignoring all frequencies below some threshold also causes severe difficulties. It is possible, indeed probable, that normalizing the remaining frequencies would improve the situation markedly, however, we lack time to experiment with this idea. We also tried computing the distance using only the DC offset, and discovered that iris recognition did not work at all. Therefore, we conclude that our iris recognition routine utilizes both low and high frequency information. This is significant, as it shows that the useful portion of our feature set consists of more than just the average intensity of each image.

APPENDIX: THE GUI – A TALE OF WOE

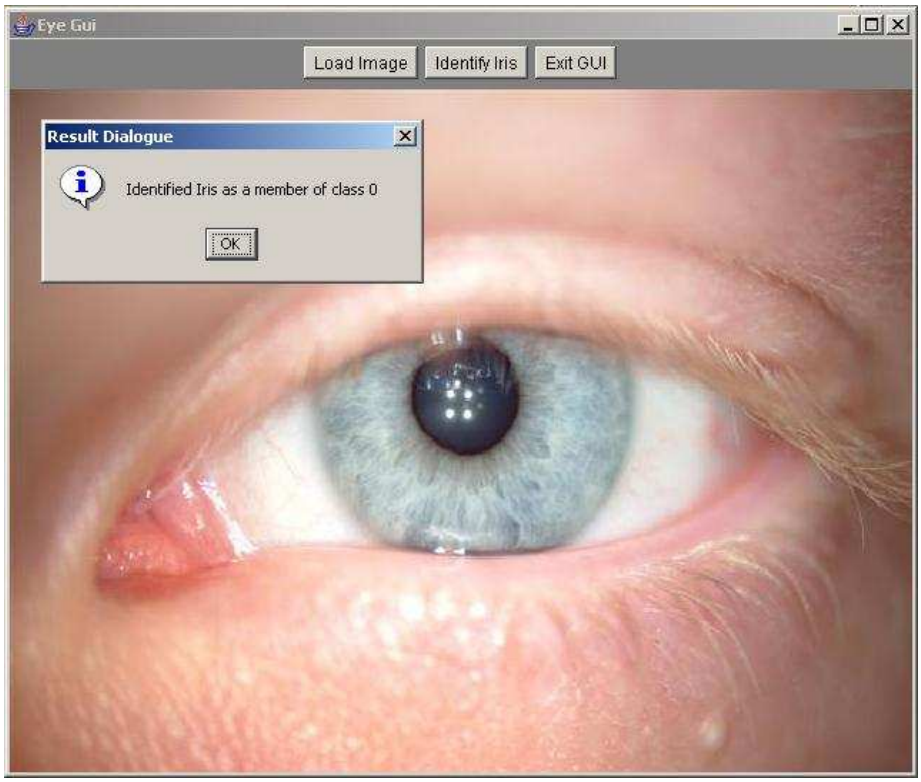
We implemented in Java a completely functional Graphical User Interface that permits the user to open an image, preview it, and then run the identification software. The actual functionality provided, to be more specific, was that it would determine the binary file associated with the JPEG selected in our database, and pass that as a command line parameter to our executable. It then waited for the completion of execution, and got the return value of the executable. While the GUI worked flawlessly at the passing of parameters and the acquisition of return values for nearly every executable we tried it with, for some reason our identification algorithm was not one of these. We have determined that all of our code prior to the return statement at the very end completed correctly, so the reason why this should have occurred is a mystery that we have been unable to solve. Included below are some images of the GUI running, but using an executable other than our identification software to provide mock results.



Opening a File



Reaction to a return value indicating an imposter



The Reaction to a return value indicating an identified eye

REFERENCES

J. Daugman. "How Iris Recognition Works." IEEE Trans. CSVT 14(1), 2004

In this paper, Daugman describes a simple method for iris recognition using Gabor wavelets to extract iridian features.

Image and Equations from http://ct.radiology.uiowa.edu/~jiangm/courses/dip/html/node67.html

Describes bilinear interpolation in a mathematically precise way. We used this information to port MATLAB's **interp2** function to C for the polar mapping code.

M. Kuhn. "Information theory and coding – Image, video and audio compression." University of Cambridge. http://www.cl.cam.ac.uk/Teaching/2003/InfoTheory/mgk/

This series of lecture slides describes many basic features of human aural and optical perception and the algorithms used in signal processing to exploit those features. We found its description of the DCT and KLT very useful in determining how to extract features.

B. Sherlock and D. Monro. "Algorithm 749: Fast Discrete Cosine Transform." ACM Transactions on Mathematical Software, Vol. 21, No. 4, December 1995

This paper describes a fast way to calculate the DCT of a signal with a length that is a power of two. The method used is somewhat similar to the FFT.

---

[1] J. Daugman. "How Iris Recognition Works." IEEE Trans. CSVT 14(1), 2004

[2] Image and Equations from http://ct.radiology.uiowa.edu/~jiangm/courses/dip/html/node67.html

[3] M. Kuhn. "Information theory and coding – Image, video and audio compression." University of Cambridge. http://www.cl.cam.ac.uk/Teaching/2003/InfoTheory/mgk/

[4] B. Sherlock and D. Monro. "Algorithm 749: Fast Discrete Cosine Transform." ACM Transactions on Mathematical Software, Vol. 21, No. 4, December 1995