18-551
Spring 2001
Group #5 Final Report

Fingerprint Recognition

Tamer El-Calamawy
Eric Huang

TABLE OF CONTENTS

1.0 INTRODUCTION

As we enter the new millennium, identification is used for almost everything nowadays. With technology on a rapid pace of improvement, the need for accurate identification is very necessary, especially with the increased use of automated transactions.

Humans have certain unique properties to them. The study of these properties is called biometrics. Biometrics is the ability to automatically recognize a person using distinguishing traits such as fingerprints, face, retina or iris from the eye, voice, and hand geometry. Each of these methods of recognition has advantages and disadvantages. Attributes such as cost, size, reliability, operating environment, speed and accuracy help determine the suitability for different applications. Without examining each of these biometric recognition methods in detail, a case can be made that fingerprint recognition has the broadest applicability for most systems and is the best place to begin a search for an appropriate biometric.

With the online shopping getting more and more popular, transactions on the internet has brought many security problems to surface as people often give out their personal information and credit card number for each of their purchases. Many times, a person's credit card number can be stolen and misused. If a biometric recognition system is applied, it can help reducing the risk of shopping online to an extent.

The use of fingerprint recognition has existed as a means of identification for many years. Not only fingerprints are more accessible, but also fingerprint recognition systems generally have lower costs, faster speed, and more reliability compare to other biometric recognition methods.

Each person has a different pattern of fingerprint, and these patterns are made of ridges, which make loops and whirls that are unique to each person.

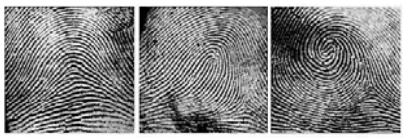


Figure 1 Different types of prints: arch, loop, whirl

Fingerprints are commonly classified as 5 different types: whorl, left loop, right loop, arch, and tented arch. For most recognition systems, difficulties arise in distinguishing between fingerprints of the same type. Many recognition systems used neural networks to find minutiaes, the ridge ends or splits, in fingerprints to match a fingerprint. Since the number of minutiaes varies among fingerprint, many people have tried to exploit this fact to enhance fingerprint matching. However, the image quality is very critical to minutiaes detection as we will discuss later. The features extracted can also vary in length depends on the type. This has resulted in some difficulties in devising a classification system.

Our system is implemented so that it yields a fixed length of features, and many classification schemes can be applied. It's more immune to noises because of the characteristic of Gabor filters that the features only preserve directions and frequency at a location .

Compare our system and the minutiae-based systems, the center point of the fingerprint is important to find for reference in obtaining features. The orientation also plays an important role for its accuracy. The algorithm for center point determination should be consistent and a rotation algorithm should be applied for better results.

Many of the problems for fingerprint recognition systems are related to obtaining the fingerprint images. Plastic distortions, scanning artifacts, scanning resolution, and uneven pressure are examples of such issues. A system has to address these problems in order to maintain reasonable and consistent results.

Fingerprint recognition is a type of image processing that requires memory and computation. In order to have a reasonable performance in speed, a recognition system has to take into consideration of the hardware constraints as well.

2.0 METHOD

2.1 PRE-PROCESSING

A scanner was not accessible, but we obtained a fingerprint database consisting of 5 different people with 60 images for each. The fingerprint images were 256x256 pixels in grayscale.

2.1.1 CENTER POINT DETERMINATION

Initially, a reference point, or center point, must be determined. The following steps in the process will all be referenced around this point. The center point is found at the point of maximum curvature of ridge lines.

i) apply Wiener filter to the image. This filter performs 2-D adaptive noise-removal. It uses a pixel-wise adaptive Wiener method based on statistics estimated from a local neighborhood of each pixel. The size of these neighborhoods are 5x5 pixels.
ii) determine the numerical gradient of the image in the x and y directions.
iii) apply the Wiener filter on the x and y gradients to enhance them further.
iv) divide input fingerprint image into blocks of 10x10 pixels.
v) compute the slope perpendicular to the local orientation of each block using the following formula:

$$\Theta = \frac{1}{2}\tan^{-1}\left(\frac{\sum_{i=1}^{10}\sum_{j=1}^{10} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{10}\sum_{j=1}^{10}\left(G_x^2(i,j)-G_y^2(i,j)\right)}\right)+\frac{\pi}{2}$$

vi) for blocks with slopes ranging from 0 to π/2, trace a path down until slope that is not ranging from 0 to π/2 is found and mark it.
vii) compute the slope in the negative y direction and output an x and y position which will be the center point of the fingerprint for block that has the highest number of marks.

Figure 2 Center point of fingerprint

2.1.2 MINUTIAE DETECTION

A minutiae is a pattern of the flow of the ridges or valleys. A fingerprint can have number of minutiaes. The minutiaes can be distinguished and classified into many different types. See the diagram below:

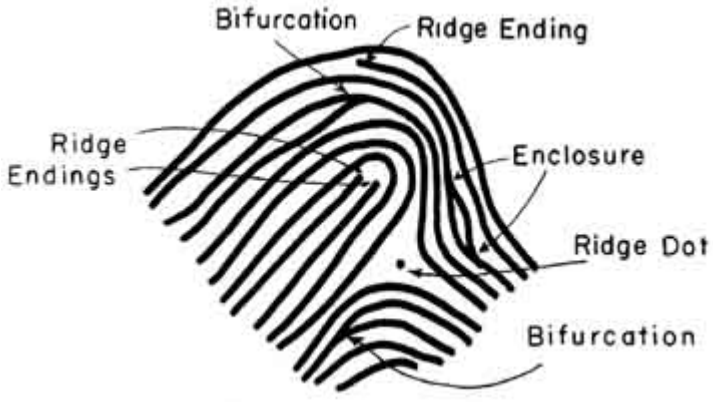


Figure 3 Minutiae Types

Since each fingerprint has its unique sets of minutiaes, many automatic fingerprint matching systems are based on minutiae matching. The traditional way is to use neural networks, which often requires lots of computing power. The quality of images also becomes very critical to minutiae detection because most programs are written to trace down the ridge lines. For example, a short noise line along a ridge can likely be mistaken for a bifurcation minutiae. Therefore, we took noises into consideration, and we tried to have more flexibility on the image quality by just searching one minutiae point near the center point of the fingerprint[1].

We first create a binary image from the scanned image. The threshold is set so that the noise on the image is not as apparent. Since we only need to find one minutiae point within a small area near center, we have used a different approach in our minutiae detection. One difference is that we only look for 'bifurcations' and 'dots'. Before we start the minutiae search, those two patterns are predefined. During the search process, we use the predefined patterns to scan through the image from 4 different directions. Once we find a minutiae, the process is stopped and the location of the minutiae is recorded along with the type. (bifurcation or dot)

2.1.3 IMAGE ORIENTATION

A major obstacle in fingerprint recognition is that the images obtained are not usually perfectly aligned. Usually rotation and displacement of some sort is evident. Attempting to compare two fingerprints with different orientations will affect the result significantly. To adjust for this, rotation of the input image is performed.

To rotate an image both the angle of rotation and the origin point are needed. To find the angle of rotation, 4 points are needed: center point of

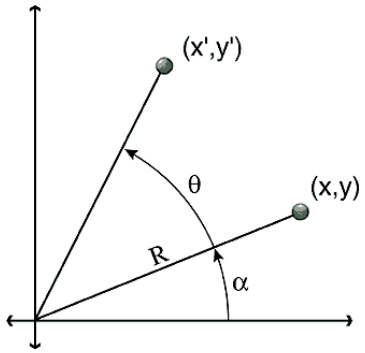
[1] The reason will be discussed later in the rotation section.

image in database and input image, and location of the same minutiae point for both images. Then the angle is calculated using the following equation:

Angle of rotation = $\tan^{-1}$(y2-center_y2 / x2-center_x2)
– $\tan^{-1}$(y1-center_y1 / x1-center_x1);

The origin point to rotate around will be the center point of the input image. The initial technique was to go through each pixel of the input image, and use the rotation equations below to find the corresponding rotated position (x', y'). This technique has problems though, when the corresponding coordinates are found they are not usually integer values. This implies that the resulting rotated image will only have pixels at integer values, and that certain output pixels will not get mapped by any input pixels. The resulting output would be completely inaccurate.

x = R * Cos(a)
y = R * Sin(a)
x' = R * Cos(a + q)
y' = R * Sin(a + q)



The method used to solve this problem is to do the reverse. For every pixel in the rotated image, calculate which pixel in the input maps to it. Doing this is simple because the negative of the angle of rotation can be used. When mapping back though the coordinates will also probably not be integers, though the closest neighboring pixel can be used. This implies that some output pixels will map to the same input, but this is acceptable as it is very difficult to observe and is significantly better than unmapped pixels.

2.2 MAIN PROCESS

2.2.1 SECTORIZATION AND NORMALIZATION

The fingerprint image needs to be normalized in order to that no differences in brightness or contrast exist throughout the image. These differences occur when acquiring the image from a scanner or other digitizing device as well as the pressure applied by the person.

Furthermore, sectorization is used in the next step in the process –feature extraction. The 6 Gabor Filters used are lined up with 12 sectors in each band.

The image is divided into 5 concentric bands, each with a radius of 20 pixels, laid over the center point. The center band has a radius of 12 pixels. The total diameter is (20*5+12)*2-1 = 223 pixels. The five bands each contain 12 wedges. Therefore, there are 60 sectors for the image. The center band is omitted, as its area is small relative to the other bands. The radius values are specifically chosen so as to avoid the effects of circular convolution.

$$N_i(x,y) = \begin{cases} M_0 + \sqrt{\dfrac{V_0 \times (I(x,y) - M_i)^2}{V_i}}, if\ I(x,y) > M_i \\ M_0 - \sqrt{\dfrac{V_0 \times (I(x,y) - M_i^2)}{V_i}}, otherwise \end{cases}$$

2.2.2 GABOR FILTERING

After we finish normalization, we will then pass the normalized image through a bank of 6 different angles of Gabor filters. Each filter is created

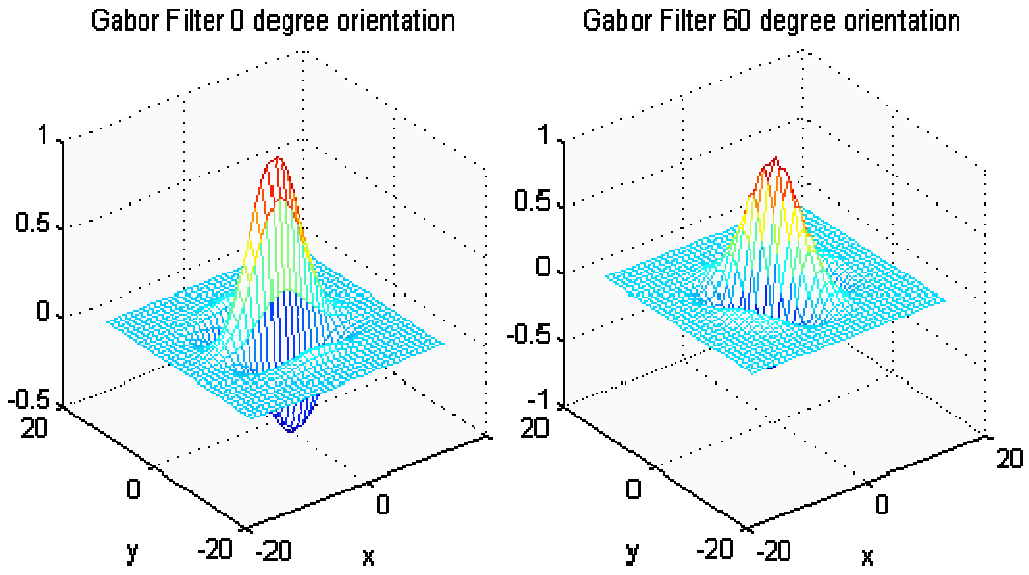
by producing a 33x33 filter image with angles 0, π/6, π/3, π/2, 2π/3 and 5π/6. By applying Gabor filters, we are able to remove noises, preserve ridge and furrow structures, and provide information contained in certain direction of the image.

$$G(x, y, f, \theta) = \exp\left\{\frac{-1}{2}\left[\frac{x'^2}{\delta_x^2} + \frac{y'^2}{\delta_y^2}\right]\right\}\cos(2\pi f x'),$$

$$x = x\sin\theta + y\cos\theta,$$

$$y = x\cos\theta - y\sin\theta$$

The Gabor filter has a height and width(33*33) to maintain its peak center point. The parameters, δ_x and δ_y, the space constants of the Gaussian envelope along x and y axis, were set to be 4.0. The frequency is determined by the inverse of the average inter ridge distance. Too large a value of f will create spurious ridges, and close ridges will merge into one if the value of f is too small. We found that the average distance to be about 10 pixels.



We have considered applying FFT to the normalized image and the filter, so that we can speed up the filtering by using only multiplication. However, applying FFT takes extra steps, and it also requires more memory space for coefficient storage. As a result, we simply have the convolution done in space. The sectorization is then used to detect the presence of ridges in the direction of the corresponding Gabor filter.

2.2. 3 FEATURE VECTOR - VARIANCE CALCULATION

After obtaining the 6 Gabor filtered images, the variance of pixel values in each sector is calculated. This variance gives an idea of the concentration of ridges in each direction in that sector. Higher variance values imply that the ridges in the sector were in a more similar direction to that angle of the Gabor filter, while lower variance values imply that the ridges were in other directions and were thus filtered out. A variance value was obtained for each sector and for 6 of the filtered images. Therefore, a total of 60 variance values are obtained and they represent the feature vector of the fingerprint. The following equation is used to calculate the variance for each sector:

$$V_{i\theta} = \sqrt{\frac{1}{K_i}\sum (F_{i\theta}(x,y) - P_{i\theta})^2}$$

$F_{i\theta}$ - pixel values in ith sector for Gabor filter with angle θ

$P_{i\theta}$ - mean of pixel values

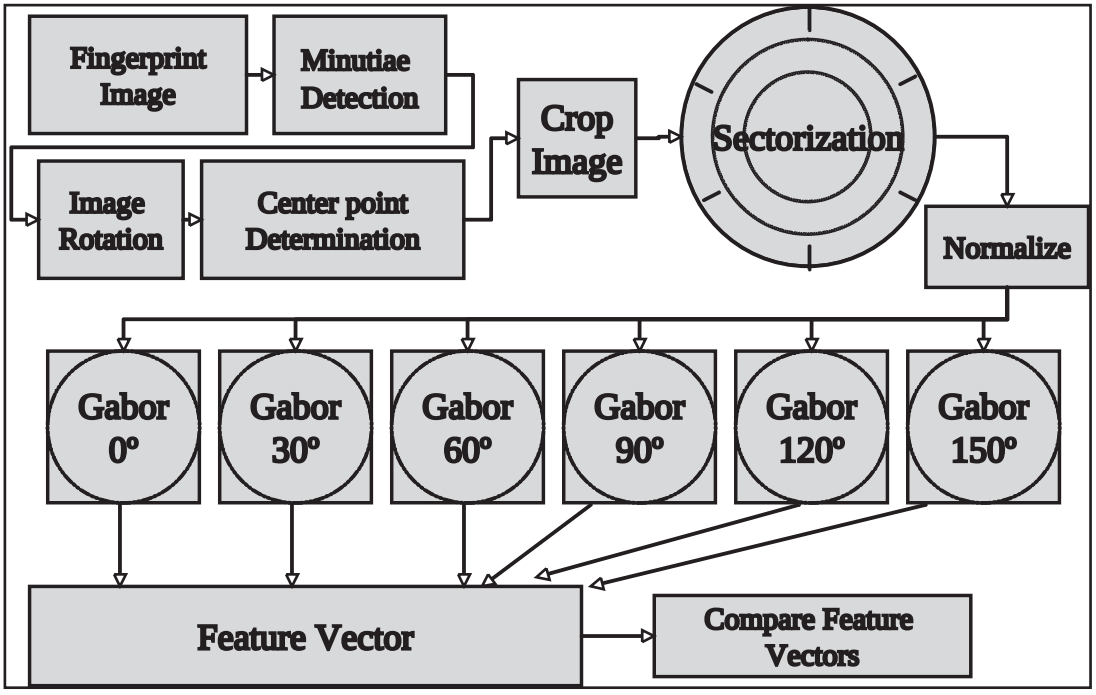
K_i – number of pixels in ith sector

2.3 POST-PROCESSING

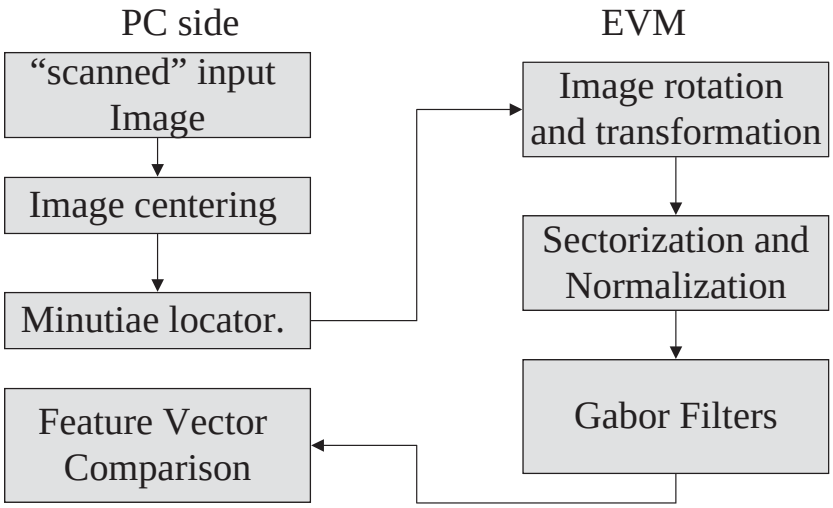
Mean Nearest Neighbor

We choose mean nearest neighbor as our classifier. One feature vector from each individual person is calculated from the training set and then saved in our database for future matching. The feature vectors in the database are gathered from better quality images. For a given fingerprint, we find that the Euclidean distance between the target feature vector and feature vector of each person in our database. The person that yields the lowest distance is chosen and matched. As we have looked into many classification schemes, the mean nearest neighbor requires the least calculations to yield a result. It also consumes the least storage space since it takes only one feature vector from each person. Although this classifier does not take into account the variations of fingerprints of the same person, it is sufficient enough for our project.

2.4 ALGORITHM OVERVIEW



2.5 SIGNAL FLOW



RESULTS & CONCLUSION

Overall, the center point determination algorithm did not give precise results. It was applied to the following 3 fingerprint images. The corresponding center points are shown for each image respectively. It is evident that the algorithm is not very accurate. Noise and low image resolution are important factors that these images possess. Better quality images may result in improved center points. However, the algorithm used is not accurate in general.

Figure 4 Results of center point algorithm

After performing a rotation on the image on the left, the resulting image has a better orientation. However, since the input is 256x256 the quality of the rotated image is not as good as the input. Higher resolutions would make the resulting image much smoother. Due to the mapping of the pixels of such a low resolution image, the result is not the same quality as the input. Nevertheless, the resulting image is still acceptable and can be processed for fingerprint matching. It would have been much better if higher quality images were available.

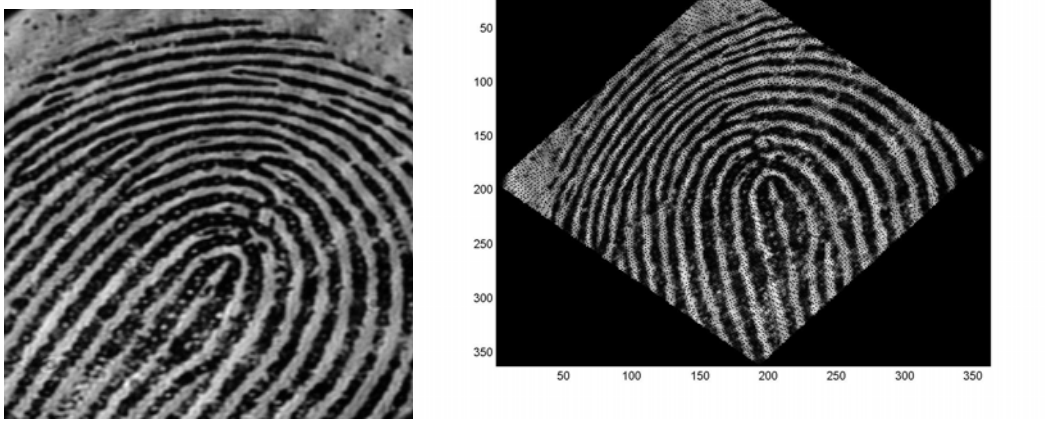


Figure 5 Fingerprint before and after rotation

As for the minutiae detection algorithm, it did not function very effectively. The main reason for this was the presence of noise in the fingerprint images. The process takes in a threshold value and binarizes the image. Therefore, noise values are attributed to a 0 or 1 and affect the results greatly. Minutiae detection is not a simple task and requires extremely high quality images. If restrictions can be placed on the quality of the input images then minutiae detection could work. The

Figure 6 Results of minutiae detector

lighter stroked circle on the fingerprint image indicates a true minutiae point, while the thicker stroked circle indicates a result of the minutiae detector. This result is incorrect due to the presence of noise in the image.

After performing, sectorization and normalization, the brightness and contrast throughout the images became constant. There were no variations apparent. The results of the 6 different Gabor filters are shown in the figure below. By applying these filters, noise was reduced, and the ridges in a similar direction to each filter were preserved. A previous group performed Gabor filtering by using the FFT and then the IFFT to obtain the result. By performing the filtering in the time domain we were able to reduce the time and memory requirements significantly.

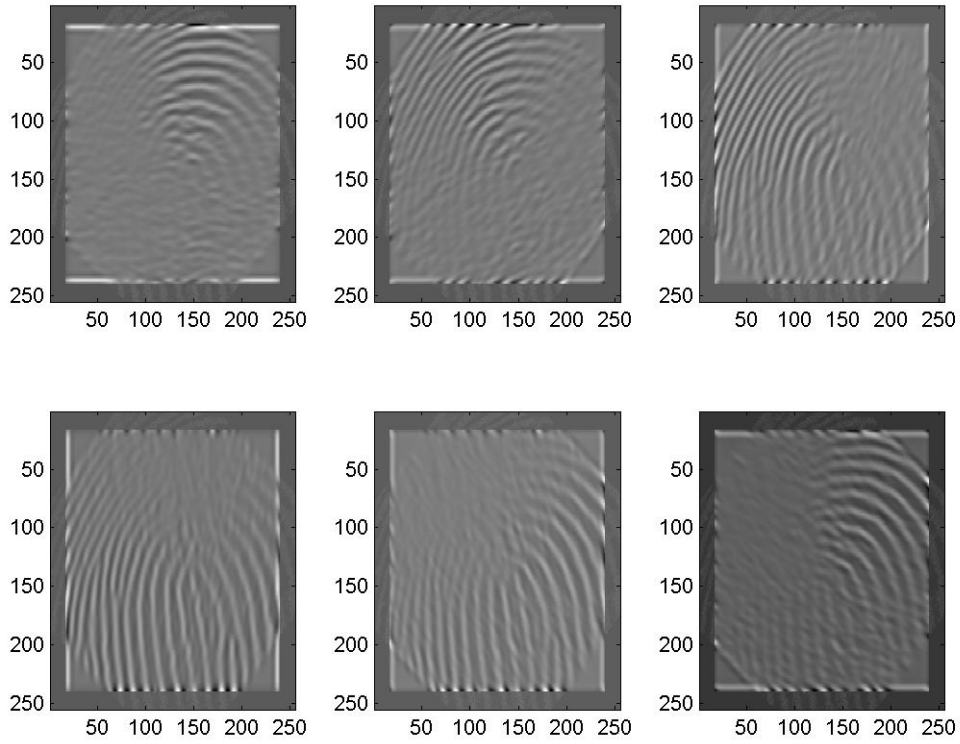


Figure 7 Results of 6 Gabor Filters

Since we had a database of 5 individuals and 60 fingerprints each, we partitioned these prints into a training set and a testing set. The 5 individuals all had the same type of fingerprint – loop. This made it more challenging to do verification among the prints. The training set consists of 15 prints that make up the database. They are also used to determine the threshold for rejection. A few of the remaining prints, many of which were of bad quality, were used as the testing set.

In order to determine a threshold value for our verification process, it was necessary to compare a fingerprint from one class to one in each of the five classes. By looking at the Euclidean distance values, it should be obvious to determine which class the fingerprint belongs. Then after repeating the process a few more times, a threshold can be estimated.

Many obstacles make fingerprint verification difficult such as incorrect orientation, displacement, missing sections, and poor quality. Orientation and displacement issues arise as differences each time a fingerprint is digitized from the person. Perfect alignment while scanning is not possible each time. When verifying a fingerprint, it is rejected if its quality is poor causing the classifier to give a Euclidean distance that is much greater than threshold.

Here is a sample of the results from the classifier when verifying a fingerprint from Class 1:

Class	Euclidean Distance
1	52195
2	95817
3	125299
4	86804
5	83989

As you can see the minimum distance among the results is for Class 1. This implies that the fingerprint most closely matched Class 1, which is correct. A thorough testing of the classifier was not performed due to limitations in time. However, from the results obtained the verification system seemed to perform extremely well considering the fingerprints were all the same type.

This research project permitted us to learn topics discussed in class as well as new topics. These include Gabor Filters, interpolation methods (rotation), and classification techniques. For the future, an improved center point determination algorithm that is more accurate should be considered. The project turned out to be an interesting experience in which team work and time management were important.

MEMORY PAGING & PROFILE RESULTS

The following table shows the results of profiling the code on the EVM:

Code	**No of cycles**
Rotation	191887842
Sectorization/Gabor Filtering/Feature extraction	1.022×10^{10}

Rotation – Internal Memory:

N/A

Rotation – External Memory:

Variable	**Size**
Frame	256x256x4
Result	362x362x4
Points	8
Total	786328 bytes

Gabor – Internal Memory:

62x3x4 = 744 bytes

Gabor – External Memory:

Variable	**Size**
Frame	255x255x4
Result	255x255x4
Gabor	33x33x4
Total	524,556 bytes

Due to the time limitation, not much optimization of the EVM code was performed. Simple optimizations such as using a common index for an array were done, but not DMA transfers. Further optimization of both the rotation and the Sectorization/Gabor Filtering code are definitely possible.

REFERENCES

1. Turn, turn, turn. http://msdn.microsoft.com/library/periodic/period99/turn.htm

2. Fingerprint Recognition Through Circular Sampling.
http://www.cis.rit.edu/~dxc0331/web_thesis/thesis.html

3. Previous 551 project. Spring 1999 Group 19.
http://www.ece.cmu.edu/~ee551/Old_projects/projects/s99_19/finalreport.html

4. Anil K. Jain, S. Prabhakar, Lin Hong, "Fingercode: A Filterbank for Fingerprint Representation and Matching," MSU CPS Technical Report Library.
http://www.cse.msu.edu/cgi-user/web/tech/document?NUM=98-36

5. Fingerprint Verification & Recognition.
http://www.cs.earlham.edu/~odo/biometrics/