Fingerprint Recognition Using Image Processing

Morgan Leake and Rochelle Mattern

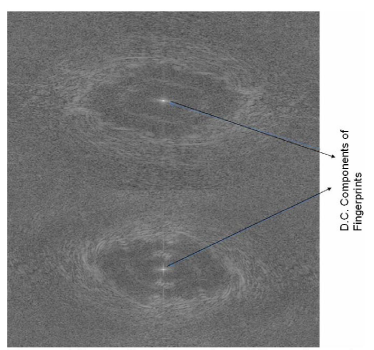
March 24, 2013

1. **Abstract**

**Image processing and correlation have helped fingerprints become a reliable method of identification. Fingerprints possess many qualities which makes them a unique identification form. These qualities include universality, the fact that each human possesses a fingerprint; uniqueness, the fact that no two humans possess the exact same fingerprint; and consistency, the fact that it does not change with age. Fingerprint patterns can be grouped into different categories such as loops, arches, and whorls. Each unique fingerprint is comprised of different arrangements of the patterns. Two methods of fingerprint comparison are commonly used: correlation and minutiae. Both methods require prior image processing in order to assure that the images being compared are not negatively affected by a difference in image quality. Such processing can include noise filtering, binarization and thinning. The method used in this investigation will encompass the use of image filtering, cleaning, and correlation in order to create an algorithm to compare fingerprints.**

1. Background

Fingerprint recognition first became a dependable form of human identification in the late 1960s due to increasing developments in computer technology. Advancements have led to image capturing and processing automation which has made this form of identification more desirable due to increasing ease. The three major reasons that identification through fingerprinting is extremely reliable include fingerprints are universal to all humans, they are unique to each person, and they remain consistent with aging. Many of these qualities do not hold true for other forms of identification, which is what makes this method so widely used. Fingerprints as a form of identification have become one of the most well-known biometrics [2].

Fingerprints can be divided into different groups, which are based upon the type of pattern. An English scientist, Sir Francis Galton, was the first to suggest a system of classification for fingerprints in the late nineteenth century. His classification system was based on grouping the patterns into arches, loops, and whorls. The points used for identification were generally referred to as “Galton points.” Galton’s identification system was based on a three-way process. The process begins by looking at “the shapes and contours of individual patterns” as observed from either an ink-based print or a digitally acquired print with special attention paid to the size of the fingerprint. One method used to quantify the size is to count the individual ridges found in the loops [3]. The Federal Bureau of Investigation (FBI) of the United States has expanded Galton’s three patterns into eight different distinguishable patterns: the radial loop, ulnar loop, double loop, central pocket loop, plain arch, tented arch, plain whorl, and accidental. The loop can be described as a “concentric hairpin or stale-shaped ridges” [3]. The terms radial and ulnar help to describe the slope of the lines on the fingerprint. The term radial describes loops that slope toward the thumb, while the term ulnar describes a slope trending toward the pinky. As a generalization, tented arches make up 5 percent of the total fingerprint patterns, whorls create approximately 30 percent, and loops about 65 percent, where the most commonly observed pattern is the ulnar loop [3].

For fingerprint recognition, engineers and mathematicians have developed different algorithms over the years to compare fingerprints to one another. One of the algorithms used is an algorithm called Spaced Frequency Transformation Algorithm. This algorithm uses the 2D Fast Fourier Transform to compare the fingerprints. The Fast Fourier Transform is used to take the image in the spatial domain and convert it to the frequency domain to allow the computer to

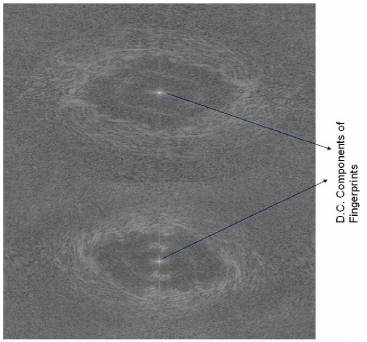
compare the images. The Fast Fourier Transform of a 2D image characterizes the image in low and high frequency components as shown in Figure 1. Each pixel in the output represents a 

Figure 1 Fourier Transforms of two different fingerprints

different frequency present in the original image. The different “shapes, types of swirls, and the position of symmetrical points” each person has in his or her fingerprints causes the Fast Fourier Transformation of each fingerprint to be unique and identifiable [3]. The uniqueness allows analysts and engineers to use the Fast Fourier Transformation to distinguish one person’s fingerprints from another’s. The computer will compare each pixel of the two images and determine the number of similarities between both images. If the count is above a certain number, the fingerprint is taken as a match to the fingerprint it was compared against; if not, the fingerprints do not match one another. Even though this is a good method to use, it is not completely accurate. When tested on a database of 750 fingerprints, it has an accuracy rate of 99% when comparing full fingerprint images. When a partial fingerprint is compared to a full fingerprint on a database, the accuracy of the algorithm drops to 85%. The 15% rejection rate for the partial compared to a full fingerprint breaks down into a 13% false rejection rate and a 2% false recognition rate. If a partial fingerprint, however, is compared to another partial print, the rate of accuracy drops even lower to 77%. For this case, the 23% failure rate is split into a 15% false rejection rate and an 8% false recognition rate [3].

Another algorithm used for fingerprint recognition is the Line Scan Algorithm. This algorithm was originally developed to decrease the time needed to process the fingerprints using the Spaced Frequency Transformation Algorithm. This algorithm takes the image and initially crops out any information in the picture that is unwanted leaving only the image of the fingerprint itself. The program calculates boundaries of all the cropped images of the fingerprints

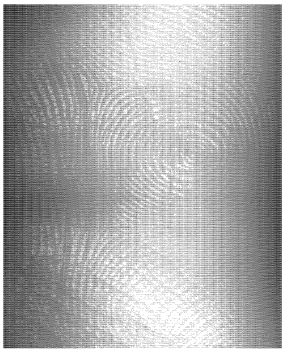


Figure 2 Cropped image containing only region of interest

in order to resize the images so they are all of uniform dimensions. Once the images are of uniform size, correlation curves are found for the images compared that relate to the row intensities of these images. The correlated images will produce symmetric patterns that can be used to determine whether or not a fingerprint is a match. The correlated images will be compared to the autocorrelation of the fingerprints in the database. Fingerprints that are a match will have curves that are similar to one another, while fingerprints that are different will have curves that are noticeably dissimilar from one another. The Line Scan Algorithm can be accomplished at a higher speed than the previous algorithm giving this algorithm an advantage. This algorithm also has a higher rate of accuracy when comparing partial fingerprints than the previous algorithm. A test of 150 fingerprints yield a 95% success rate for partial fingerprints and a 99% accuracy rate for the case of full fingerprints [3].

Another method implemented to compare fingerprints is referred to as the minutiae algorithm. On a typical fingerprint, there are about one hundred minutiae points, which are “local ridge characteristics that appear as either a ridge ending or a ridge bifurcation” [3]. When using minutiae points to compare fingerprints, the points are represented using a coordinate system. Methods using these points have included the measurement of the distances between points, which is not the most reliable since finger size may vary with age; spaced frequency transformation, which follows the frequency of ridge patterns; and line scanning, which is based on correlation.

1. Procedure

The following section has been broken into tasks according to the proposed schedule located in Appendix A and includes a list of equipment and supplies.

Equipment and Supplies

* VersaFine Stamp Ink Pad (Onyx Black)
* 8 ½ x 11 in. White Copy Paper cut into 2 x 2 in. squares
* Scanner
* Laptop PC
* MATLAB R2012a
* MATLAB R2012a Image Toolbox

**Tasks**

1. Import fingerprints and create database and traceability document.
2. Create an Excel sheet with a list of classmates’ names and numeric file name in order to maintain tractability without linking the results to names.
3. Take right thumb fingerprint of each classmate by lightly inking the fingerprint and then pressing the thumb onto the square of paper. (Be sure the fingerprint is pressed onto the paper and not rolled.)
4. Using the scanner, scan each fingerprint into the computer and crop the image using the Snipping Tool.
5. Save each image in jpeg format, and name each file with the identification number described in the Excel database.
6. Using the ‘imread’ function, import each of the images into MATLAB.
7. Preprocessing
8. Use the ‘rgb2gray’ function to convert the three-dimensional matrix of the image to a two-dimensional matrix. This function also converts the image to grayscale, which is necessary for later preprocessing.
9. Make all images the same size matrix using ‘imresize’. This makes the matrix from each fingerprint have a width of 210. It is necessary to make each matrix the same width in order to correlate the fingerprints with the unknown later in the process.
10. Adjust the image in order to increase the contrast of the grayscale image by using the ‘imadjust’ function, which enhances the features within the fingerprint.
11. Thin out the lines of the fingerprint using the ‘clean’ setting of the function ‘bwmorph’. This is necessary because when using ink and manually taking fingerprints, often times the fingerprint is over inked and needs to be thinned for better processing.
12. Detect the edges of the image using the ‘prewitt’ option of the ‘edge’ function. This makes the edges appear white, while everything else becomes black.
13. Correlate input unknown fingerprint with each fingerprint in database.
14. Autocorrelate the unknown fingerprint with itself. The autocorrelation can be done using the ‘xcorr2’ function.
15. Using the ‘xcorr2’ function again, cross-correlate the unknown fingerprint with each fingerprint in the database.
16. Plot 3D graphs of the autocorrelation and each cross-correlation using the ‘mesh’ function on one figure for easier comparison.
17. Finding a match
18. Find the maximum ‘z’ value on the mesh graph and its ‘x’ and ‘y’ coordinates using the data cursor for the autocorrelation plot and each of the cross-correlation plots.
19. Take a slice of the mesh plot at the x-coordinate found in the previous step and plot the 2D plot of the matrix at this point. This allows easier analysis of an individual row within the total matrix. To do this use the ‘plot’ function and the defining the ‘z’ value for which row of the matrix to plot.
20. Using the plots created in the previous step, subtract the autocorrelation plot from each cross-correlation plot, and plot the difference graphs. With the difference graphs, an ideal match will give you a plot with a straight line at zero.
21. To compare the difference graphs, square each element in the difference matrix in order to make all values positive. This eliminates positive and negative values from cancelling one another. This can also be done by taking the absolute value of each element in the matrix
22. Using the ‘sum’ function, add up all the elements in each matrix. Ideally the match will be equal to zero; however, since the images being correlated are not the exact same images, the match will be equal to the lowest sum.
23. Automating the match
24. The find the lowest sum that will result in the match, make a matrix containing each sum. Then use the ‘min’ function to find the minimum sum in the matrix. Using ‘if’ statements with the position of the sum in the matrix, determine which sum value is the lowest and its position in the matrix.
25. When the ‘if’ statement is true, have MATLAB display the identification number corresponding to the fingerprint match from the database.
26. Results and Conclusions

Originally, each classmate’s fingerprint was taken in order to create a database to be able to compare unknown fingerprints to. The fingerprint identification numbers were traced through an Excel spreadsheet that linked the identification numbers to a name. The database of images is shown in Figure 3.



Figure 3 Database of fingerprint images

In the final result, only three fingerprints were selected to use for a database to compare the unknown fingerprint to when testing the code in order to make the plots larger and easier to analyze. (NOTE: Even though only three fingerprints were chosen to demonstrate the results, the method is still valid for all ten fingerprints.) Through research in fingerprint matching, it was discovered that the most commonly used preprocessing includes thinning, contrast enhancement, and edge detection. These processes are what the preprocessing applied is based upon. The preprocessing included in the final result was the result of a trial and error process used to create an algorithm that concluded with the best results. Through this trial and error process, thinning was included and removed, and the order of thinning and edge detection was switched. It was found that the order needed to be to convert the image to grayscale, resize the matrix, adjust the contrast, thin the lines in the image, and then edge detection. Additionally, it was discovered that the results were much better with the thinning included in the preprocessing since the fingerprint was manually taken causing most fingerprints to have excess ink. Furthermore, several different settings of the ‘bwmorph’ function for thinning and the ‘edge’ function for edge detection were

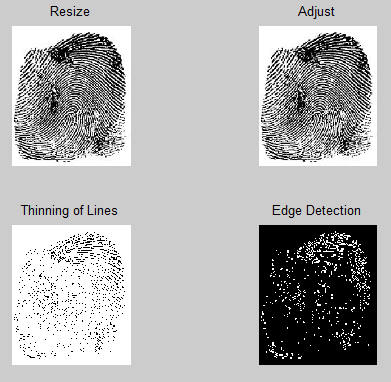


Figure 4 Preprocessing of images

tried. Again, this was a trial and error process to see which setting produced optimal results. The final code utilized the ‘clean’ setting of the ‘bwmorph’ function and the ‘prewitt’ setting of the ‘edge’ function. It was discovered that when using edge detection each image must be grayscale; however, the images had already been converted to grayscale for both correlation reasons and other preprocessing reasons. The fingerprints after each process are shown in Figure 4.

Following preprocessing, a correlation section of code was created in order to take the autocorrelation of the unknown fingerprint and the cross-correlation of the unknown with each of the three fingerprints from the database. This was done using the 2D cross-correlation, which is the ‘xcorr2’ function in MATLAB. When using the ‘xcorr2’ function, it was important that the image was a 2D matrix with double precision. This was accomplished through changing the image to grayscale and defining the precision through the ‘double’ function. After correlating the fingerprints, a ‘mesh’ subplot was created for the autocorrelation and each of the cross-correlations, which can be seen in Appendix B. This was done to see the matrix plotted versus the intensity of the pixels in the fingerprint. A slice was then taken of each mesh graph to examine one row of pixels from the image of each fingerprint as seen in Appendix B. To determine which row to take the slice at, the coordinates of the highest intensity on each mesh graph was taken. The x-coordinates from each peak were then used to examine that row of the mesh graph as a 2D plot using the ‘plot’ function. After the graphs of the slice were taken, the difference between each cross-correlation slice and the autocorrelation slice was taken. The autocorrelation slice graph was subtracted from each cross-correlation slice graph to create a difference graph between the unknown fingerprint and each fingerprint from the database, which can been seen in Appendix B. The purpose of taking the difference of the graphs is to determine the best match for the unknown fingerprint to a fingerprint in the database. If one of the fingerprints from the database was the exact same image as the unknown fingerprint inputted, the difference graph would display a straight line located at zero. Since the unknown fingerprint is not the same image as any of the fingerprints in the database and is another image of the same fingerprint, the difference graph will not be completely zero. Since the graph will not be completely zero, the graph closest to zero will be the match. To compare the difference graphs and determine which graph was the closest to zero, each element in each matrix was squared. The square of each element was taken to make all the elements in the matrix positive, so positive and negative values did not cancel each other creating a false match. Once the square of each element in each matrix was taken, all the elements in each matrix were summed. The fingerprint from the database that matches the unknown fingerprint will have the lowest sum out of all the sums taken. To compare the sums, an array was created with all the values of the sums from each difference graph created. The minimum of the array and its corresponding position was found by using the ‘min’ function. After the minimum was found, ‘if’ statements were used to output a message to inform the user which fingerprint from the database was a match to the unknown fingerprint inputted. Some problems were initially run into when trying to get MATLAB to display which fingerprint from the database was the best match. Comparing the maximum of the autocorrelation mesh graph to the maximum of each cross-correlation mesh graph was initially attempted. Problems were encountered with this method though because it only compared the maximum peak and did not take into account local maximums that may be present and need to be taken into consideration. Finding local maximum points was researched, but with the time constraint of the project, it appeared too difficult to implement. A sample of the output can be seen in Figure 5. Once MATLAB outputs which fingerprint from the database the unknown fingerprint matches, the user can then refer to the Excel spreadsheet to determine a name that the unknown fingerprint belongs to. The database can be referenced in Appendix C.

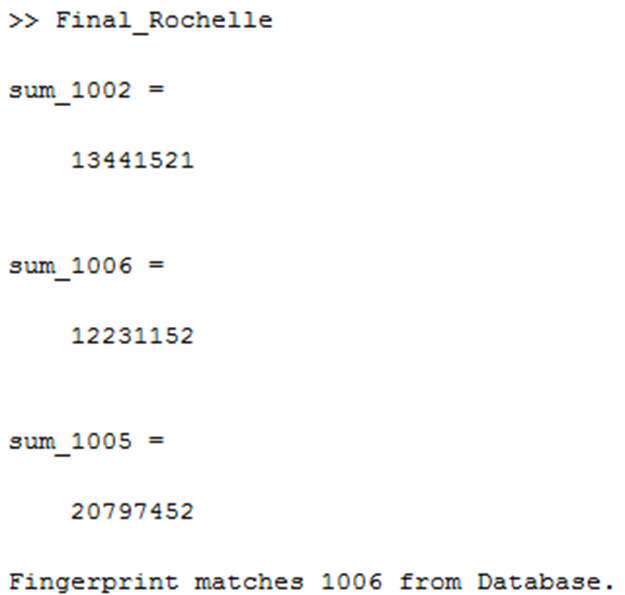


Figure 5 Sample output from MATLAB code

References

[1] Zhang Jinhai. (2011, July 26-30). Fingerprint identification system based on DSP. *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2011*. [Online]. *2*. pp.1071,1074. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6037143&tag=1

[2] NSTC Subcommittee on Biometrics. (2006, August 7). Fingerprint Recognition. [Online] pp. 1-4. Available: http://www.biometrics.gov/documents/fingerprintrec.pdf

[3] Mil''shtein, S.; Pillai, A.; Shendye, A.; Liessner, C.; Baier, M. (2008, May 12-13). Fingerprint Recognition Algorithms for Partial and Full Fingerprints. *Technologies for Homeland Security, 2008 IEEE Conference on*. [Online] pp.449, 452. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4534494

**Appendix A: MATLAB code for Final Results**

% Import each image and convert to 2D image

I\_1002 = imread('1002.jpg');

info1002 = imfinfo('1002.jpg');

img2d\_1002 = rgb2gray(I\_1002);

I\_1006 = imread('1006.jpg');

info1006 = imfinfo('1006.jpg');

img2d\_1006 = rgb2gray(I\_1006);

I\_1010 = imread('1005.jpg');

info1010 = imfinfo('1005.jpg');

img2d\_1010 = rgb2gray(I\_1010);

input = imread('Rochelle.jpg');

info\_input = imfinfo('Rochelle.jpg');

img2d\_input = rgb2gray(input);

%Resize and adjust image

width = 210;

images = {img2d\_1002,img2d\_1006,img2d\_1010,img2d\_input};

for k = 1:4

dim = size(images{k});

images{k} = imresize(images{k},[width\*dim(1)/dim(2) width],'bicubic');

end

resize\_1002 = images{1};

resize\_1006 = images{2};

resize\_1010 = images{3};

resize\_input = images{4};

adjust\_1002 = imadjust(resize\_1002);

adjust\_1006 = imadjust(resize\_1006);

adjust\_1010 = imadjust(resize\_1010);

adjust\_input = imadjust(resize\_input);

%Thin the lines in the fingerprint

thin = {adjust\_1002,adjust\_1006,adjust\_1010,adjust\_input};

for i = 1:4

thin{i} = bwmorph(thin{i}, 'clean');

end

thin\_1002 = thin{1};

thin\_1006 = thin{2};

thin\_1010 = thin{3};

thin\_input = thin{4};

%Edge Detection

detect = {thin\_1002,thin\_1006,thin\_1010,thin\_input};

for j = 1:4

detect{j} = edge(detect{j},'prewitt');

end

edge\_1002 = detect{1};

edge\_1006 = detect{2};

edge\_1010 = detect{3};

edge\_input = detect{4};

figure(1)

subplot(221)

%autocorrelation of input(unknown fingerprint)

autocorr = xcorr2(double(edge\_input));

mesh(autocorr);

title('Autocorrelation of Unknown Fingerprint');

xlabel('Pixels');

ylabel('Pixels');

zlabel('Pixel Intensity');

grid on

subplot(222)

%cross-correlation of input with image 1002

input\_img1002 = xcorr2(double(edge\_input),(double(edge\_1002)));

mesh(input\_img1002);

title('Cross-correlation of Unknown with Fingerprint 1002');

xlabel('Pixels');

ylabel('Pixels');

zlabel('Pixel Intensity');

grid on

subplot(223)

%cross-correlation of input with image 1006

input\_img1006 = xcorr2(double(edge\_input),(double(edge\_1006)));

mesh(input\_img1006);

title('Cross-correlation of Unknown with Fingerprint 1006');

xlabel('Pixels');

ylabel('Pixels');

zlabel('Pixel Intensity');

grid on

subplot(224)

%cross-correlation of input with image 1005

input\_img1005 = xcorr2(double(edge\_input),(double(edge\_1010)));

mesh(input\_img1005);

title('Cross-correlation of Unknown with Fingerprint 1005');

xlabel('Pixels');

ylabel('Pixels');

zlabel('Pixel Intensity');

grid on

%Slice taken of Mesh graph

figure(2)

subplot(221)

plot(autocorr(243,:));

title('Slice Autocorrelation of Unknown Fingerprint');

xlabel('Pixels');

ylabel('Pixel Intensity');

grid on

subplot(222)

plot(input\_img1002(233,:));

title('Slice Cross-correlation of Unknown with Fingerprint 1002');

xlabel('Pixels');

ylabel('Pixel Intensity');

grid on

subplot(223)

plot(input\_img1006(251,:));

title('Slice Cross-correlation of Unknown with Fingerprint 1006');

xlabel('Pixels');

ylabel('Pixel Intensity');

grid on

subplot(224)

plot(input\_img1005(273,:));

title('Slice Cross-correlation of Unknown with Fingerprint 1005');

xlabel('Pixels');

ylabel('Pixel Intensity');

grid on

%Difference taken between Autocorrelation and each Cross-correlation graph

figure(3)

diff\_1 = input\_img1002(233,:) - autocorr(243,:);

subplot(221)

plot(diff\_1002);

title({'Difference Between Cross-correlation of','Unknown with Fingerprint 1002 and Autocorrelation'});

xlabel('Pixels');

ylabel('Pixel Intensity');

grid on

diff\_2 = input\_img1006(251,:) - autocorr(243,:);

subplot(222)

plot(diff\_1006);

title({'Difference Between Cross-correlation of','Unknown with Fingerprint 1006 and Autocorrelation'});

xlabel('Pixels');

ylabel('Pixel Intensity');

grid on

diff\_3 = input\_img1005(273,:) - autocorr(243,:);

subplot(223)

plot(diff\_1005);

title({'Difference Between Cross-correlation of','Unknown with Fingerprint 1005 and Autocorrelation'});

xlabel('Pixels');

ylabel('Pixel Intensity');

grid on

%Square each element in each matrix

squared\_1002 = diff\_1002.^2;

squared\_1006 = diff\_1006.^2;

squared\_1005 = diff\_1005.^2;

%Sum all the elements in each matrix

sum\_1002 = sum(squared\_1002);

sum\_1006 = sum(squared\_1006);

sum\_1005 = sum(squared\_1005);

%Create matrix of sums and find the minimum sum

compare = [sum\_1002,sum\_1006,sum\_1005];

[a,b] = min(compare);

match = b;

%Determine which sum is a match and output which fingerprint from database

%the input matches

if match == 1

disp('Fingerprint matches 1002 from Database.')

end

if match == 2

disp('Fingerprint matches 1006 from Database.')

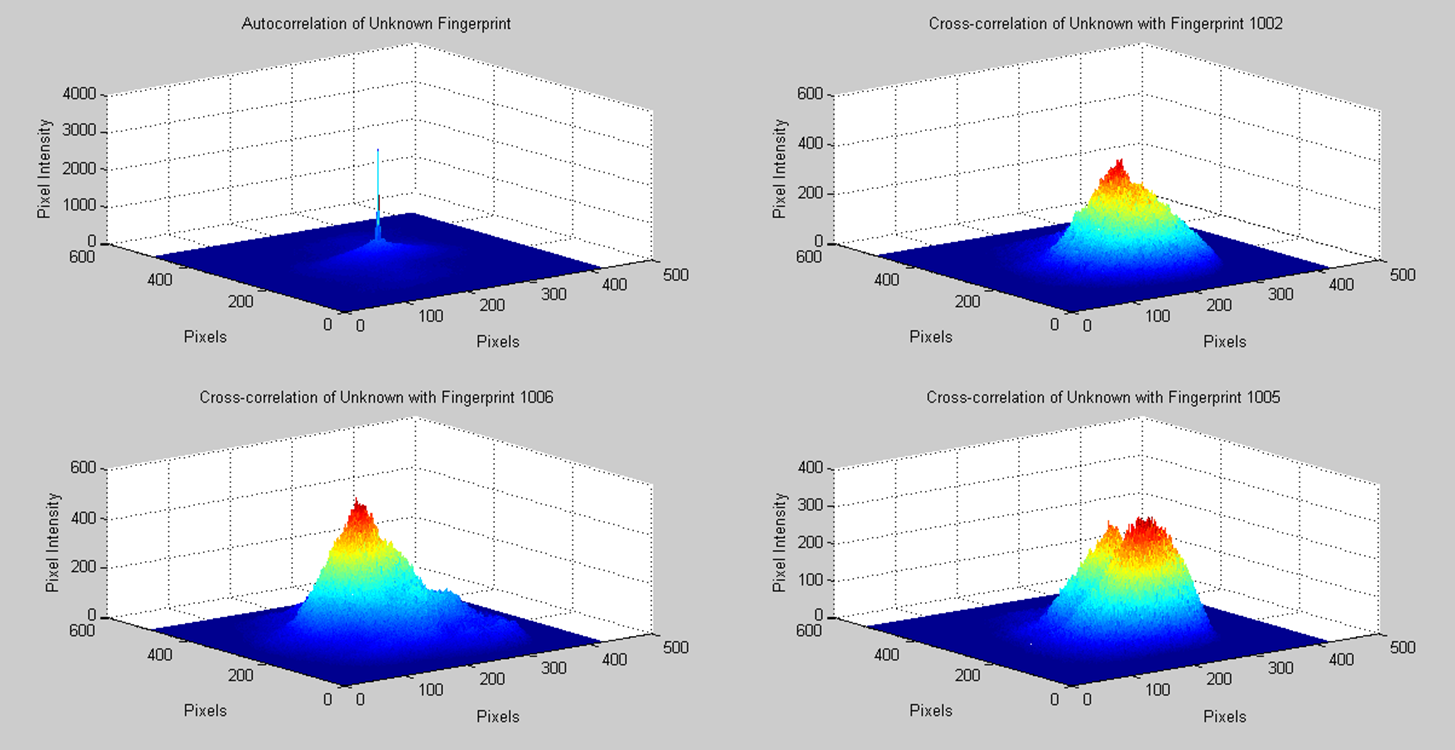
end

if match == 3

disp('Fingerprint matches 1005 from Database.')

end

**Appendix B – Sample Graphs**

****

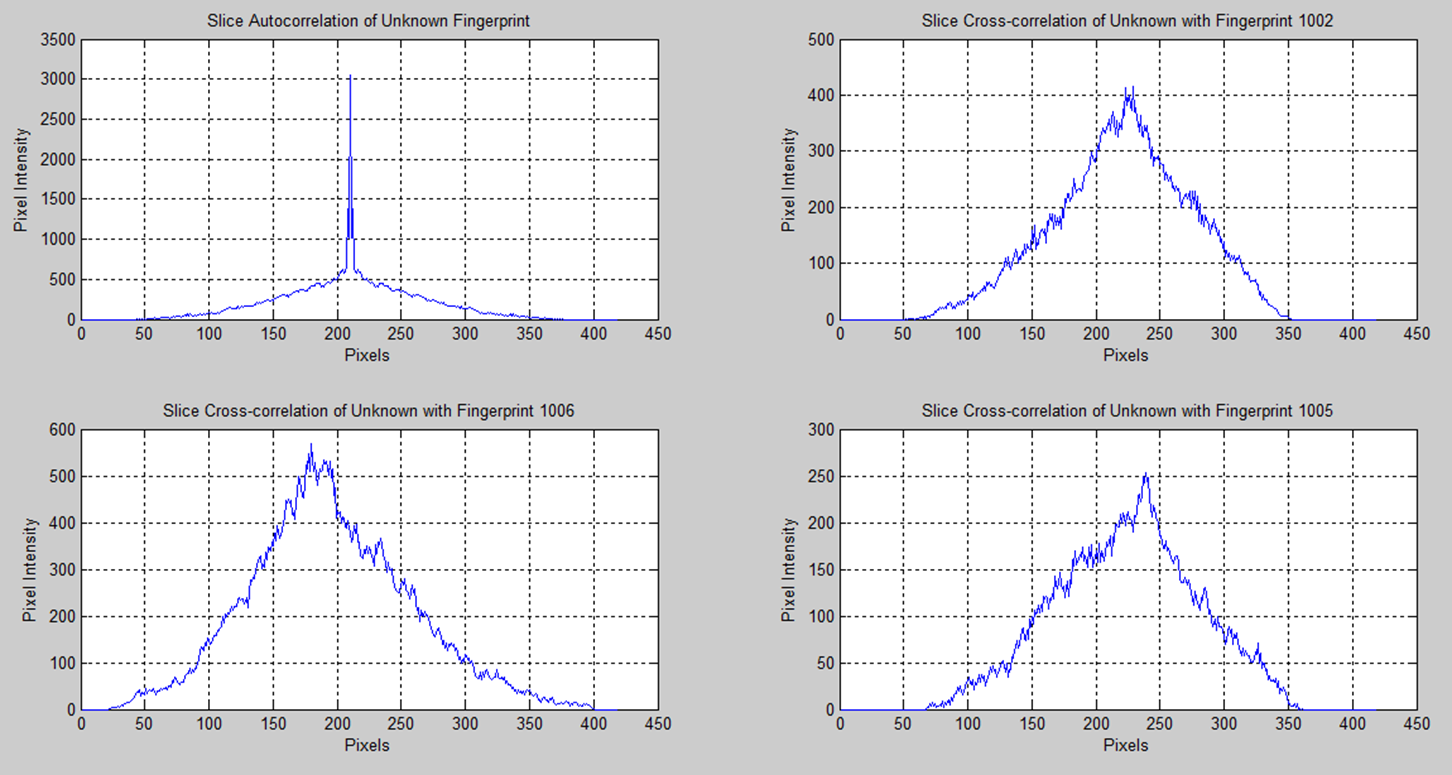
****

Figure 2 Slice of mesh graphs

Figure 1 Mesh graphs

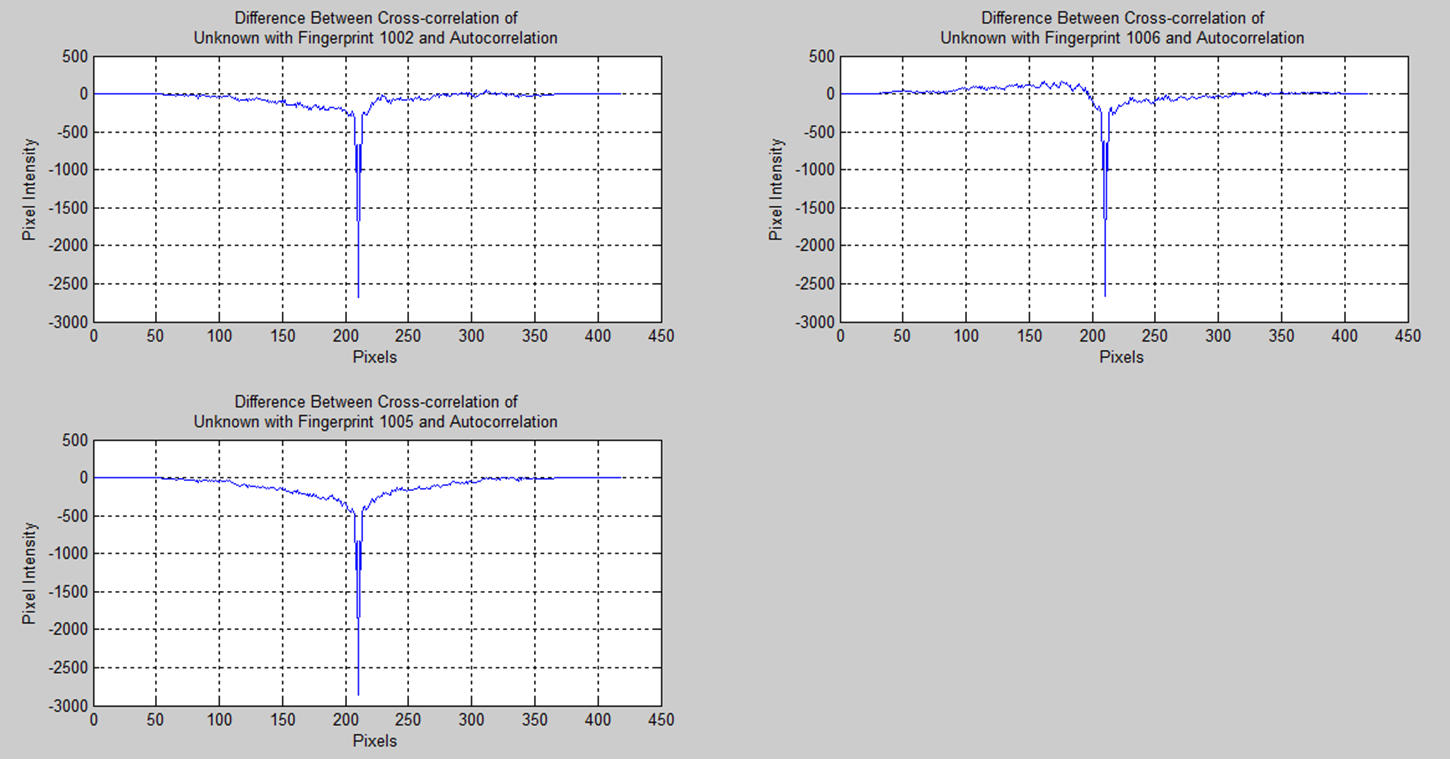
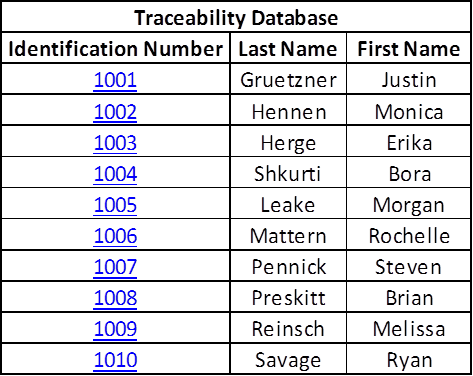
****

Figure 3 Difference Graphs

**Appendix C – Excel Database**

****