ENCM 509 Lab 4-5

February 22, 2019

Group Members: Andrew Schneider, Patrick Settle

Instructor: Dr. Svetlana Yanushkevich

Introduction

This laboratory has two major parts. Part I is to investigate various processing and pre-processing techniques for our collected fingerprints. This processing and feature extraction involves steps like image filtering, segmentation, orientation estimation, finding singularity points, ridge orientation, and ridge enhancement using Gabor filters. The minutiae used in this lab are ridge ends and bifurcations, and this information is stored as a feature vector that indicates the position and orientation of each feature.

The processed data samples are used in Part II, which investigates matching algorithms for scoring a comparison between feature vectors representing a probed image and an image in a database. One of the two approaches is based on the Gabor filtered features, and the other is based on minutiae-skeleton matching after aligning the images. In either case, the matching score is based on the euclidean distance between the probed and database templates.The matching scores will be used as a basis for a simple identification system.

Exercise:

Part I Experiment - Investigating Pre-Processing Techniques

In this part of the lab, we'll investigate various preprocessing techniques on a high-quality fingerprint and a poor-quality fingerprint. The good high-quality fingerprint is Patrick's 27th signature, and our "poor-quality" fingerprint is Patrick's 22nd signature, since it seems to have a bizarre range of contrast and some key features of the print are not made out very well.

2.1.3 - 2.1.5 - Pre-processing, enhancement, de-noising

Lab4Fingerprint1.m was modified to show the effect of histeq(), adapthisteq(), imadjust(), wiener2(), medfilt2() on minutiae extraction.

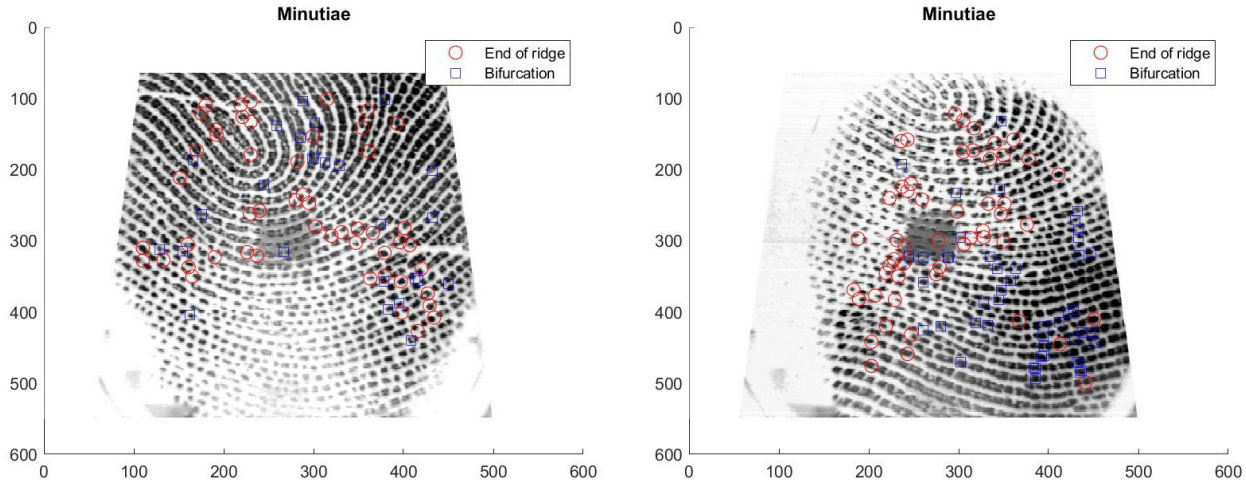

Figure 1. Minutiae Extraction on good (left) and poor (right) samples - No Pre-Processing

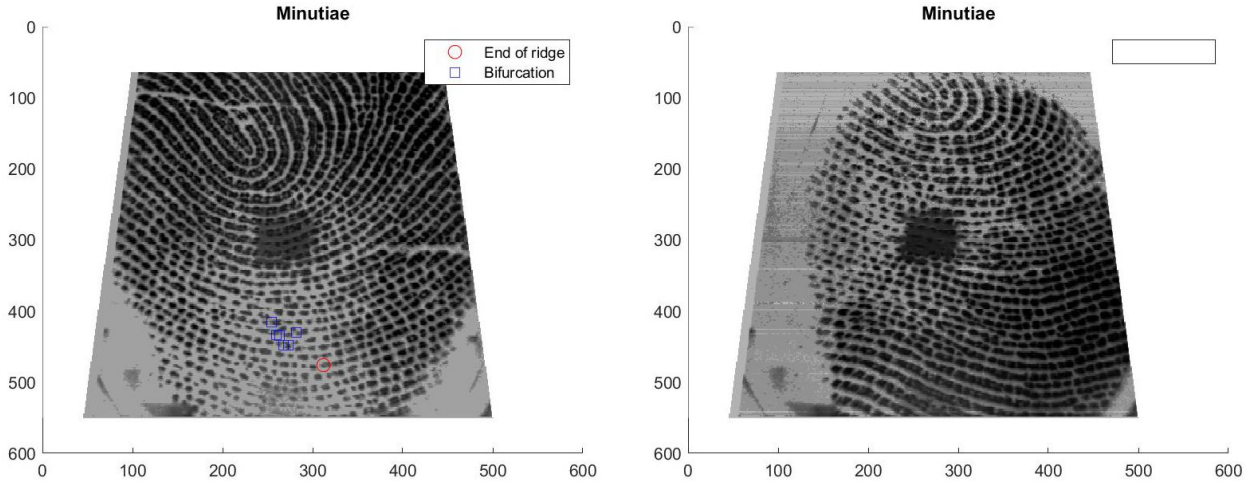

Figure 2. Minutiae Extraction on good (left) and poor (right) samples - Pre-Processed using histeq()

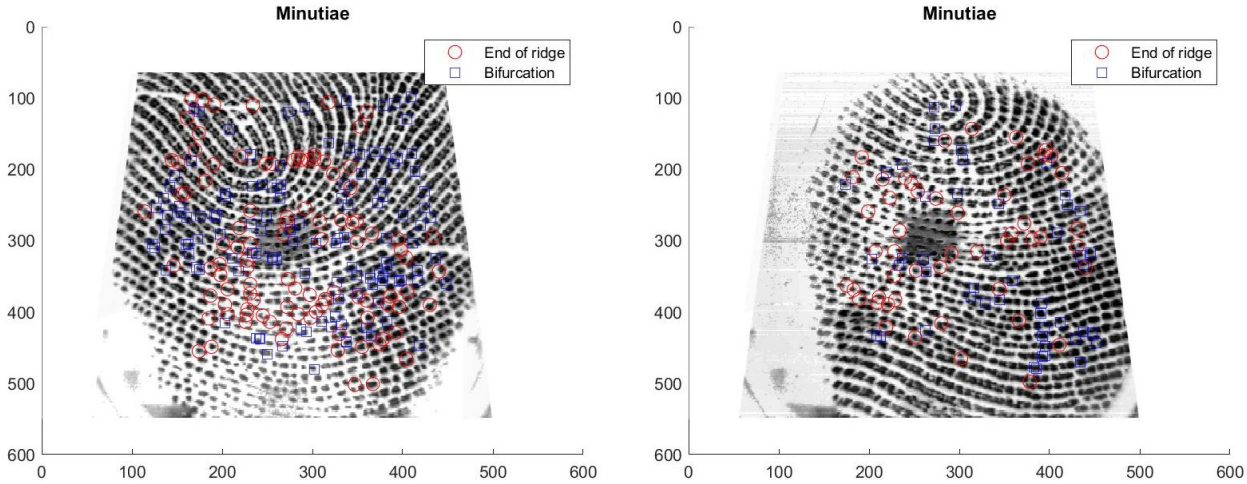

Figure 3. Minutiae Extraction on good (left) and poor (right) samples - Pre-Processed using adapthisteq()

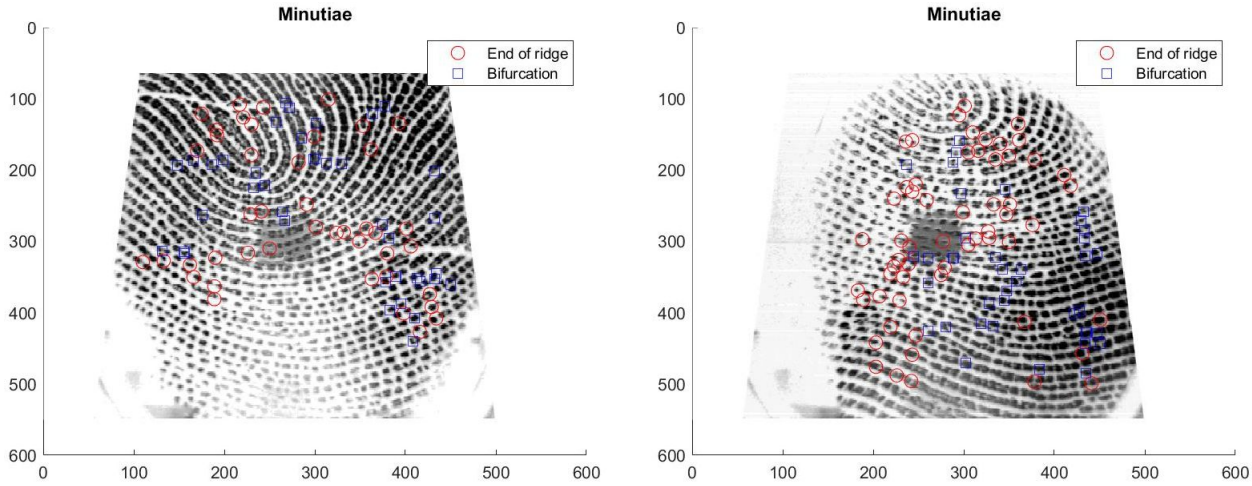

Figure 4. Minutiae Extraction on good (left) and poor (right) samples - Pre-Processed using imadjust()

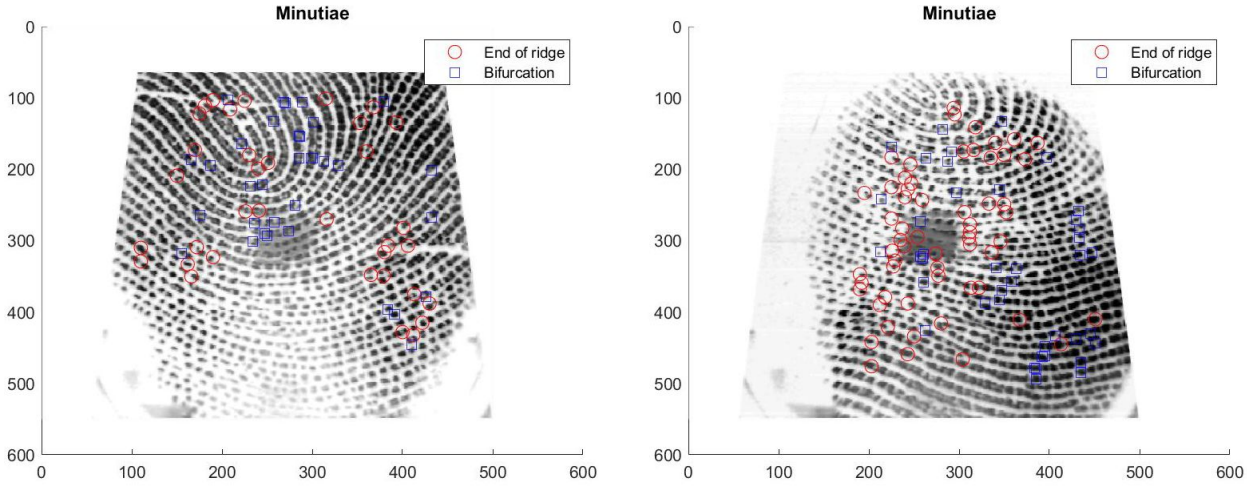

Figure 5. Minutiae Extraction on good (left) and poor (right) samples - Pre-Processed using wiener2()

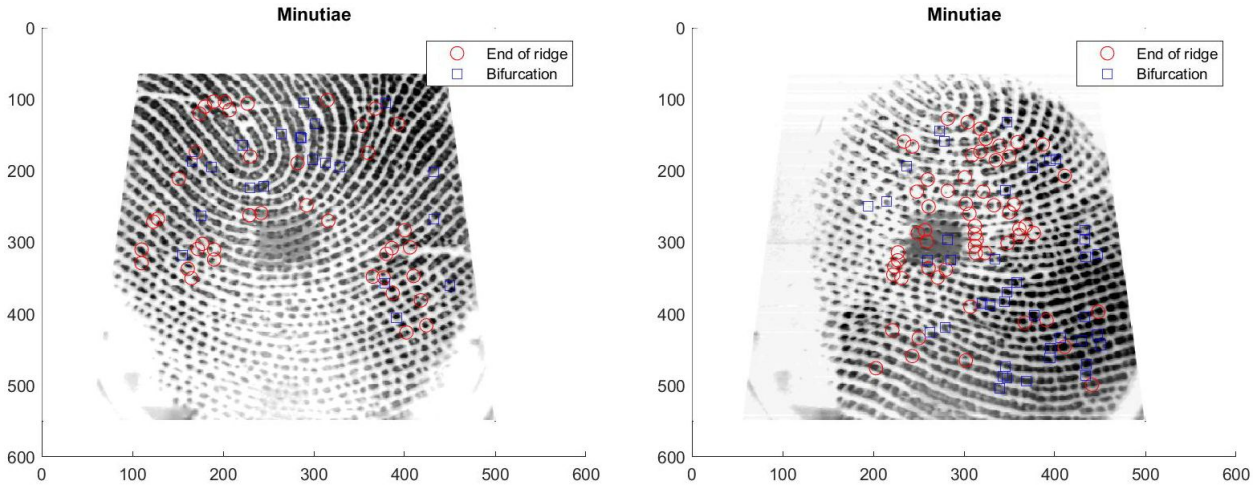

Figure 6. Minutiae Extraction on good (left) and poor (right) samples - Pre-Processed using medfilt2()

Using adaptive histogram equalization resulted in the best-looking image and good minutiae extraction. However, normal histogram equalization negatively impacted the contrast in the images and caused minutiae extraction to be very poor. The rest of the filters appeared to perform about the same.

2.1.6 - Segmentation

If the parameter [16 16] is increased the border of the segmented area becomes smoother and appears to be made of more discrete parts. If the parameter [16 16] is lowered the border of the segmented area becomes more discrete and appears to be made of a series of large squares stitched together.

If gradDev is increased the frequency and severity of variations in the border of the segmented area become more dramatic and pronounced; the border contains more deep valleys and high mountains. If gradDev is decreased the frequency and severity of variations decreases; the borders are made of longer straight segments, and variations from straight are more subtle and less pronounced.

These changes is can be seen in the figures 7 through 11 below.

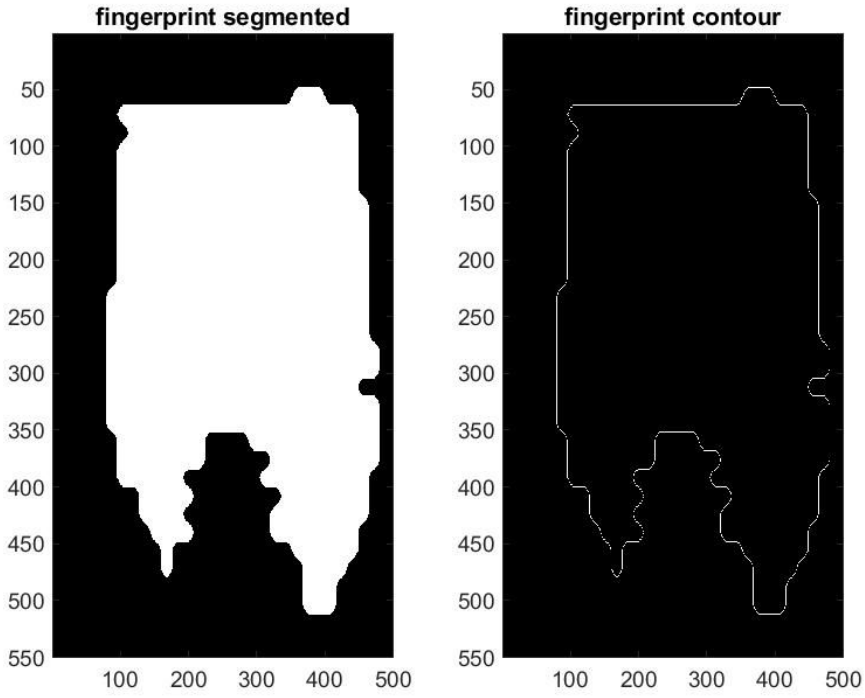

Figure 7. Segmentation on good sample with default settings.

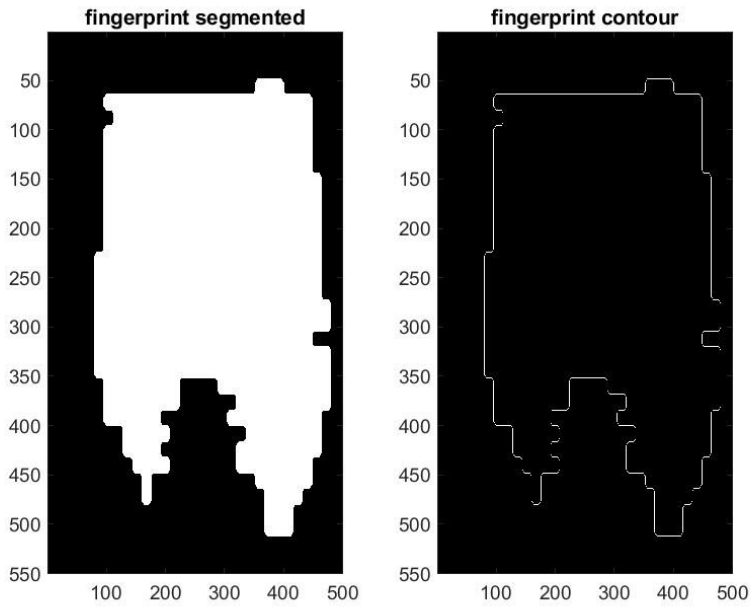

Figure 8. Segmentation on good sample with medfilt2 8, gradDev 30

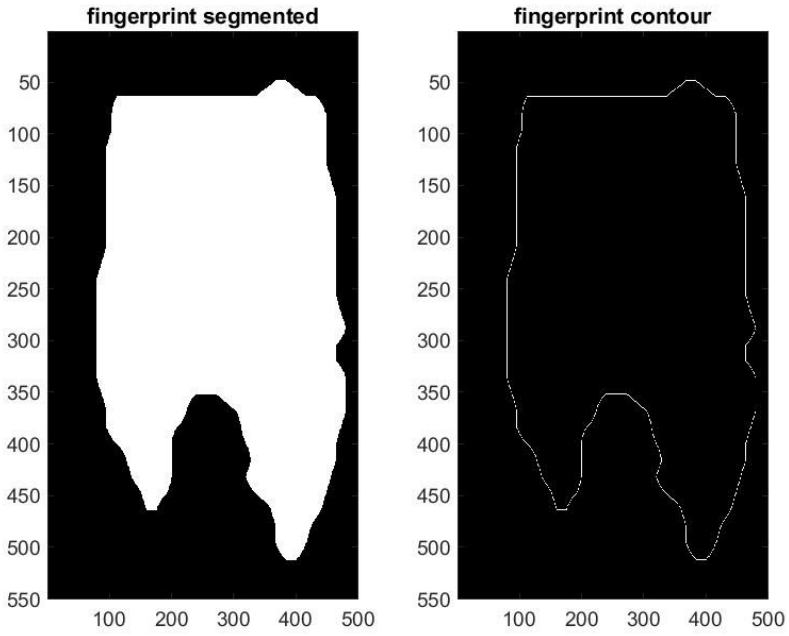

Figure 9. Segmentation on good sample with medfilt2 32, gradDev 30

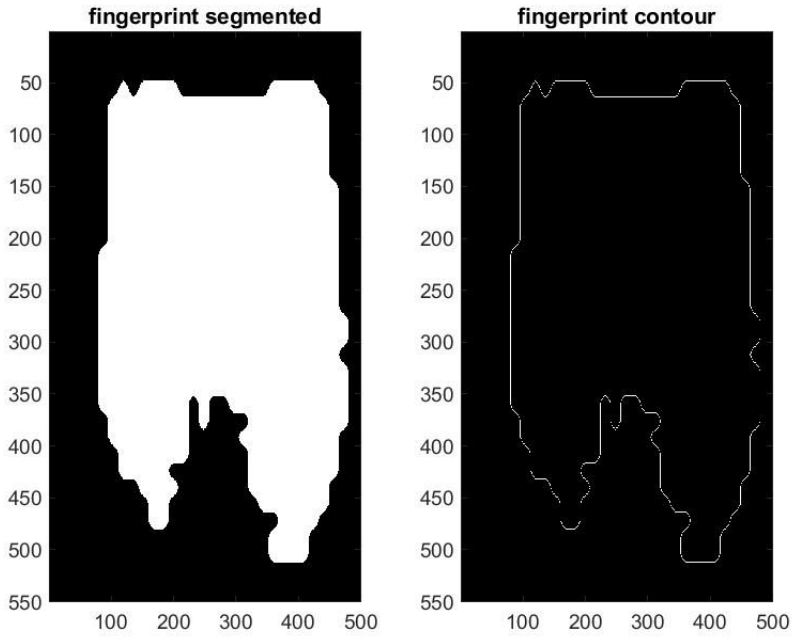

Figure 10. Segmentation on good sample with medfilt2 16, gradDev 25

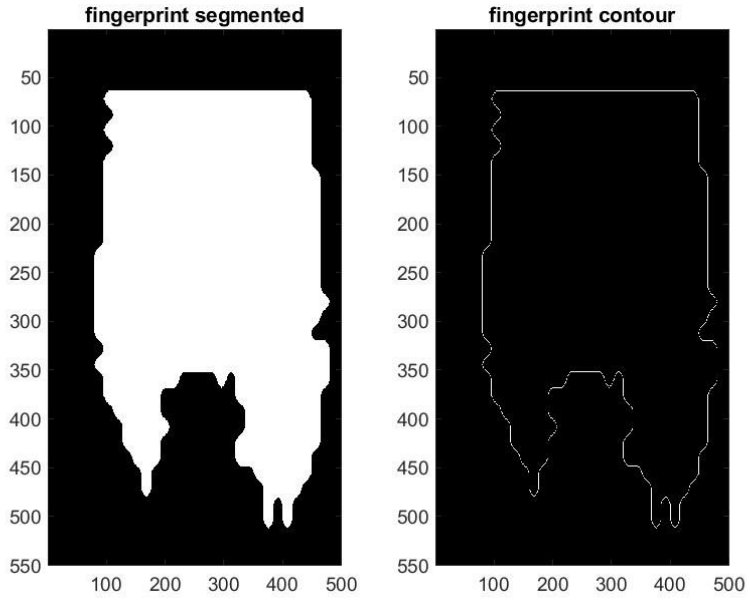

Figure 11. Segmentation on good sample with medfilt2 16, gradDev 35

2.1.7 - Orientation and singularity points

If blkSize is increased the orientation field becomes smoother and more continuous, and fewer singularity points are found. If blkSize is decreased the orientation field becomes scattered and erratic, and more singularity points are found.

If blkSize is unchanged for the good and poor fingerprints, there are 4 and 9 singularity points found, respectively.

If blkSize is increased for the good and poor fingerprints, there are 15 and 24 singularity points found, respectively.

If blkSize is increased for the good and poor fingerprints, there are 1 and 0 singularity points found, respectively.

These effects can be seen in figures 12 through 17 below.

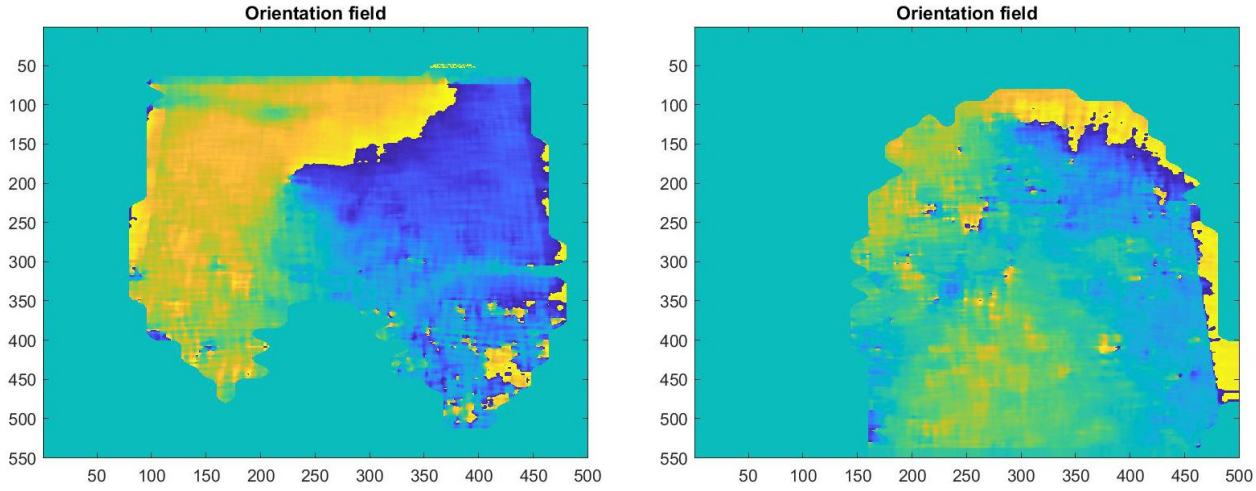

Figure 12. Orientation analysis on good sample (left) and poor sample (right) with blkSize of 10

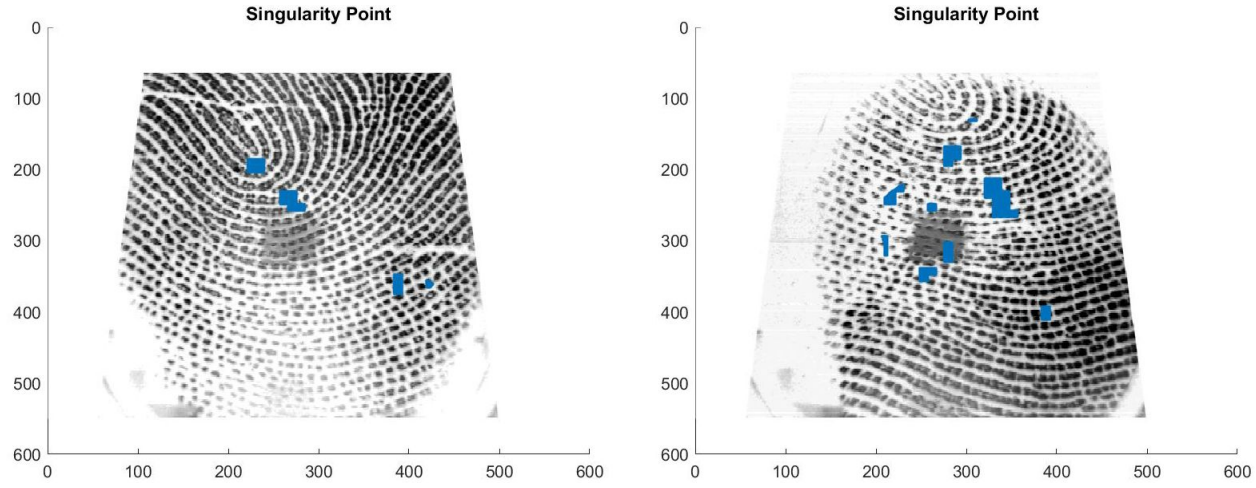

Figure 13. Singularity point analysis on good sample (left) and poor sample (right) with blkSize of 10

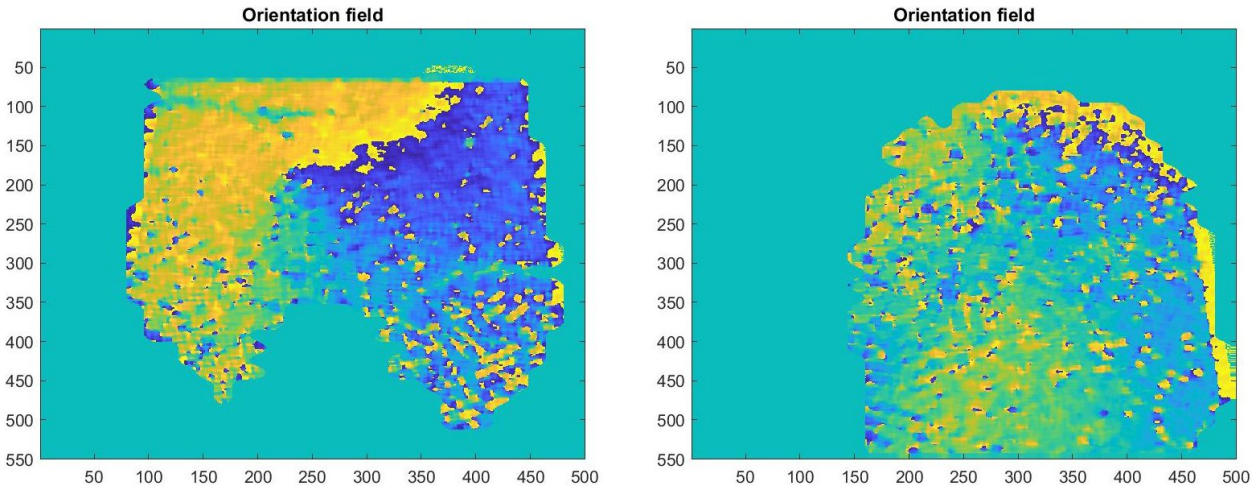

Figure 14. Orientation analysis on good sample (left) and poor sample (right) with blkSize of 5

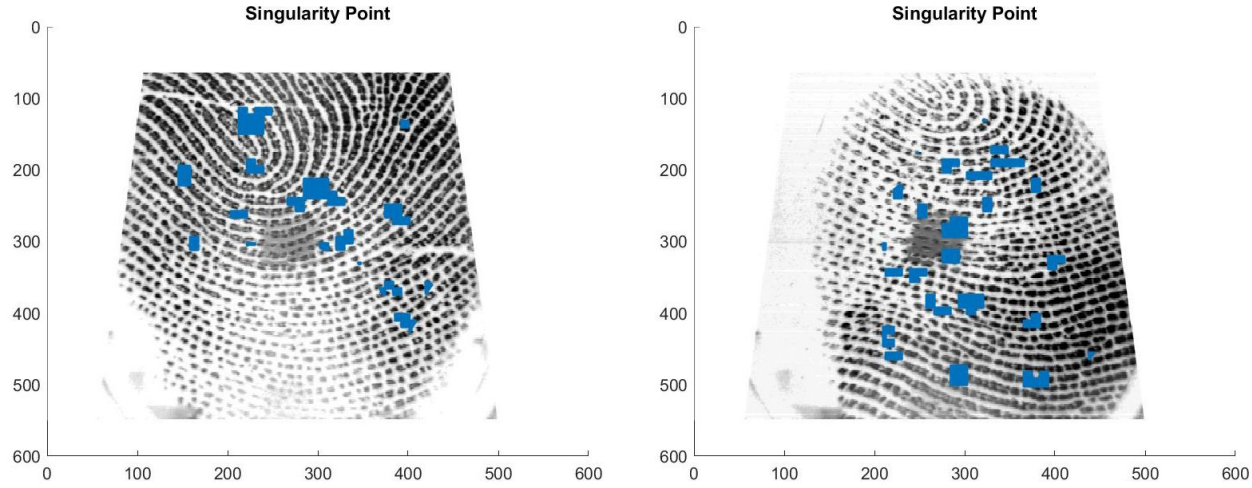

Figure 15. Singularity point analysis on good sample (left) and poor sample (right) with blkSize of 5

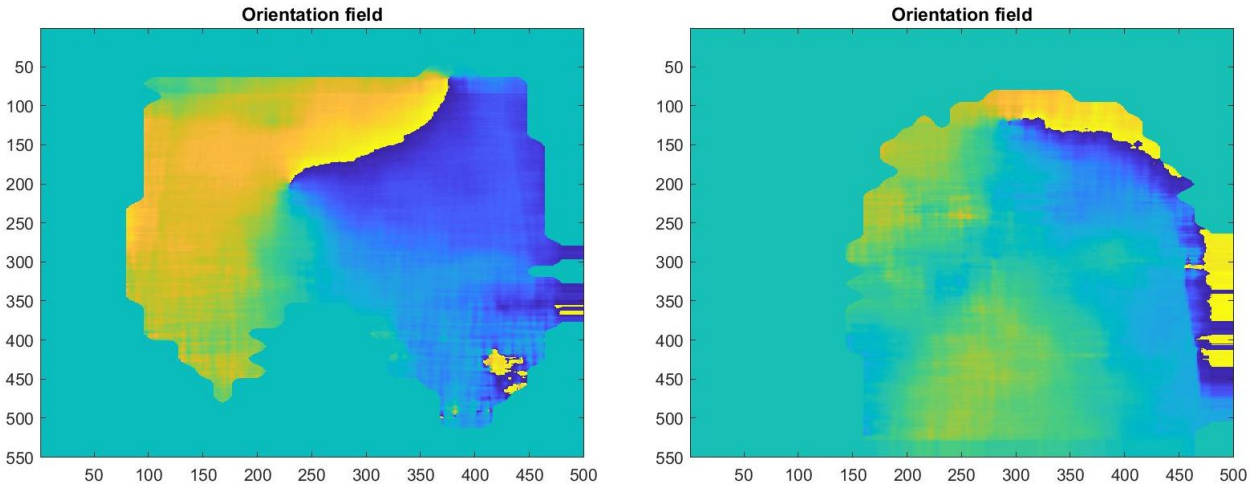

Figure 16. Orientation analysis on good sample (left) and poor sample (right) with blkSize of 20

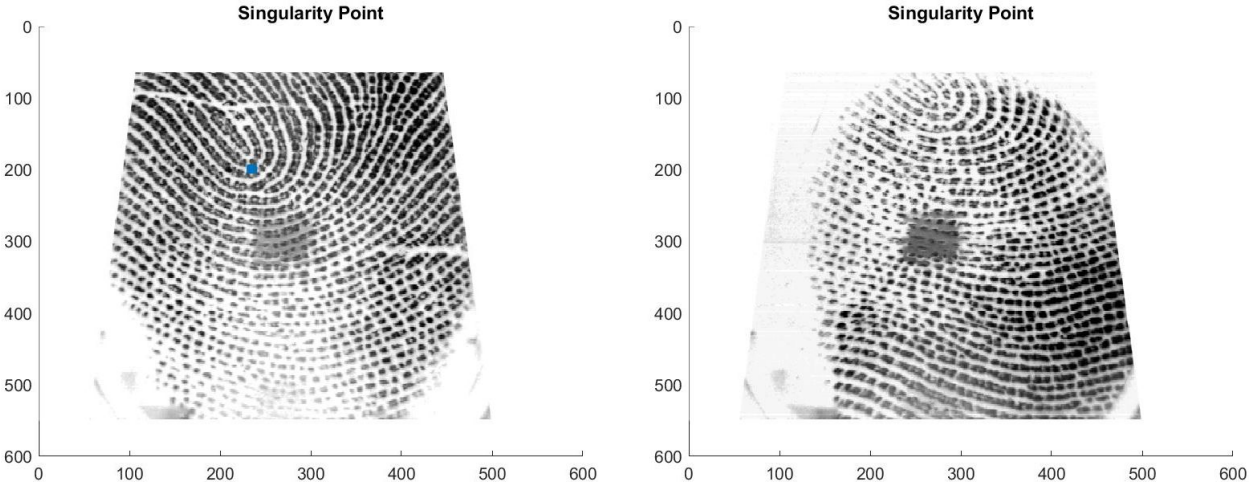

Figure 17. Singularity point analysis on good sample (left) and poor sample (right) with blkSize of 20

2.1.8 - Frequency of ridges

In the call to computelocalfrequency.m, decreasing the ‘border’ parameter does affect the results of minutia finding, table 1 shows how the number of minutiae found changes as a function of border. The source images can be found in figures 18 through 23 below.

Table 1. Effect of border Parameter on Minutia Finding

Value of border	Minutia Count in good image		Minutia Count in poor image	
	Ridge End	Bifurcation	Ridge End	Bifurcation
30	52	25	52	46
15	42	35	62	42
5	38	42	48	43

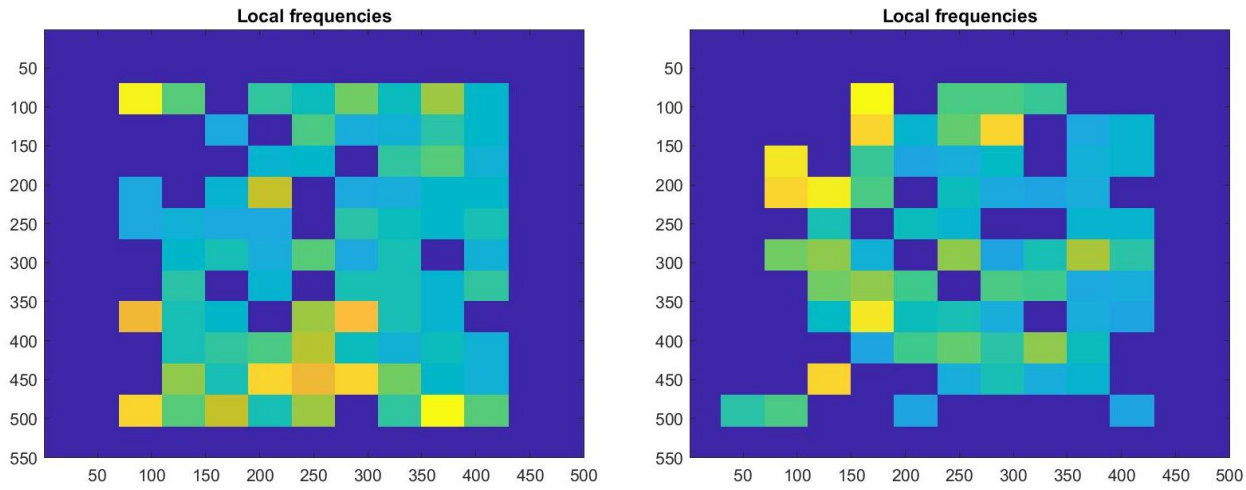

Figure 18. Local frequency analysis on good sample (left) and poor sample (right) with border of 30

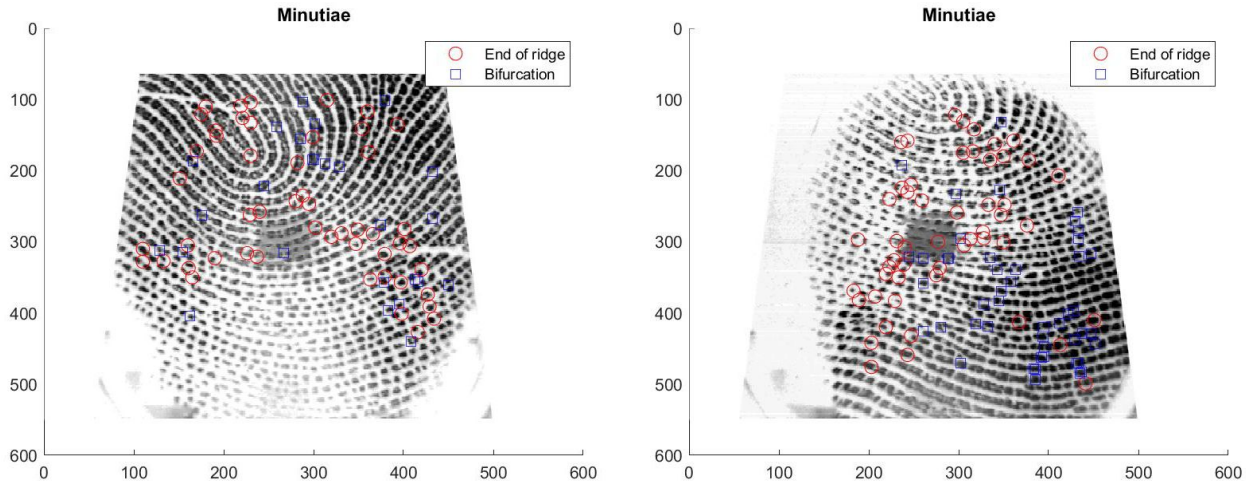

Figure 19. Minutiae analysis on good sample (left) and poor sample (right) with border of 30

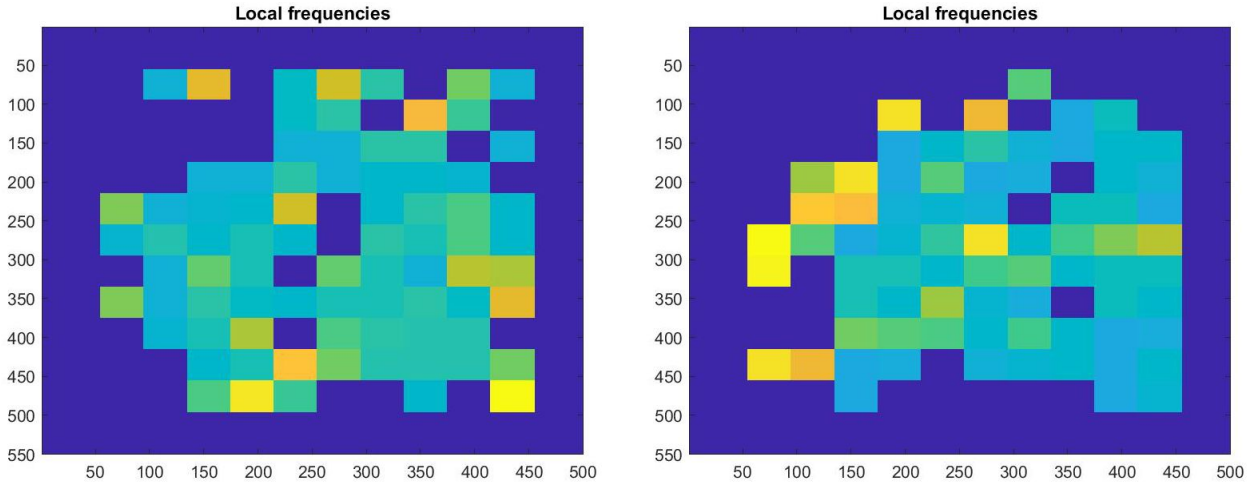

Figure 20. Local frequency analysis on good sample (left) and poor sample (right) with border of 15

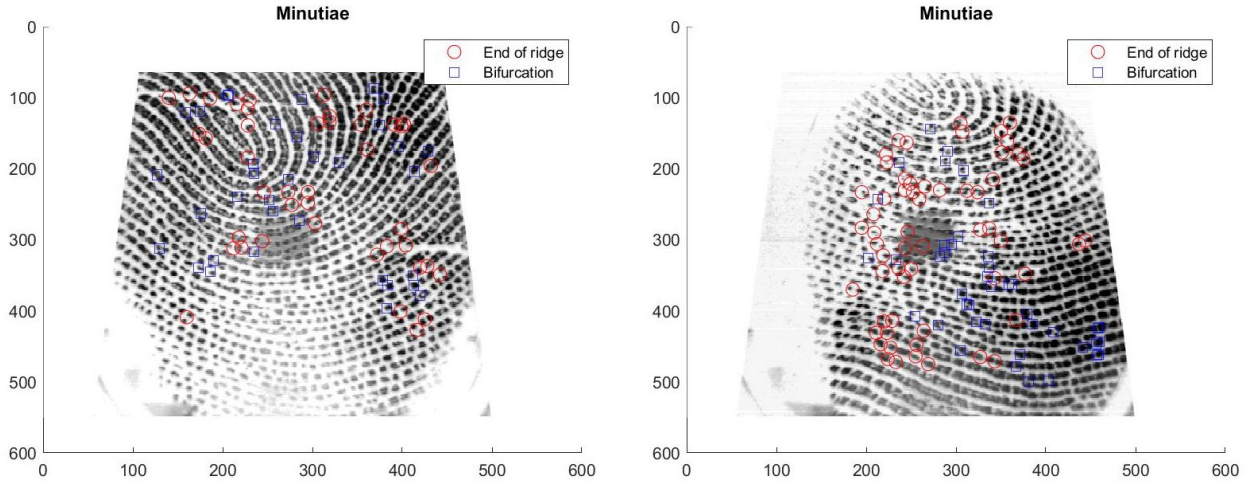

Figure 21. Minutiae analysis on good sample (left) and poor sample (right) with border of 15

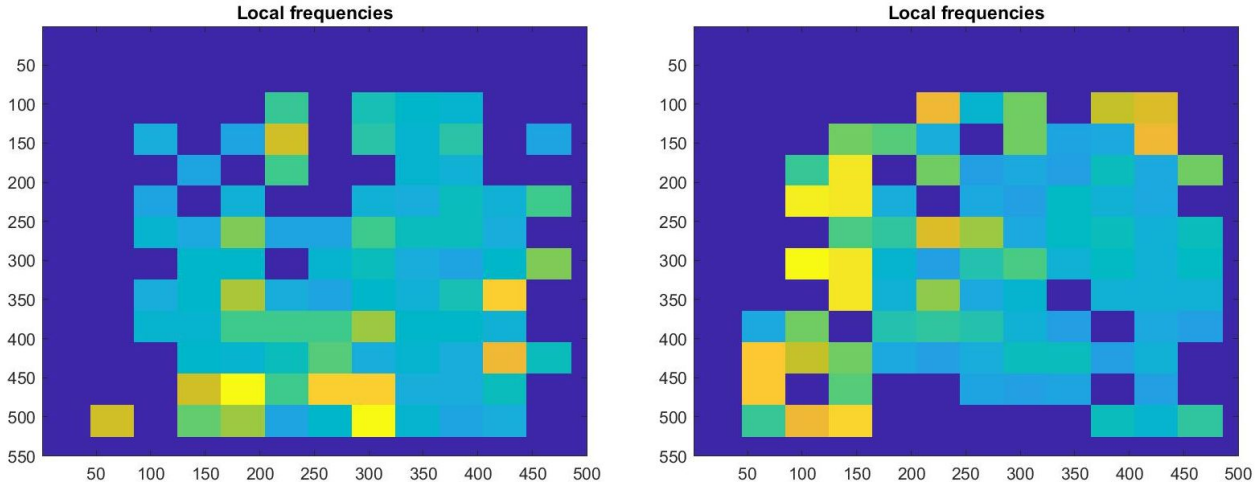

Figure 22. Local frequency analysis on good sample (left) and poor sample (right) with border of 5

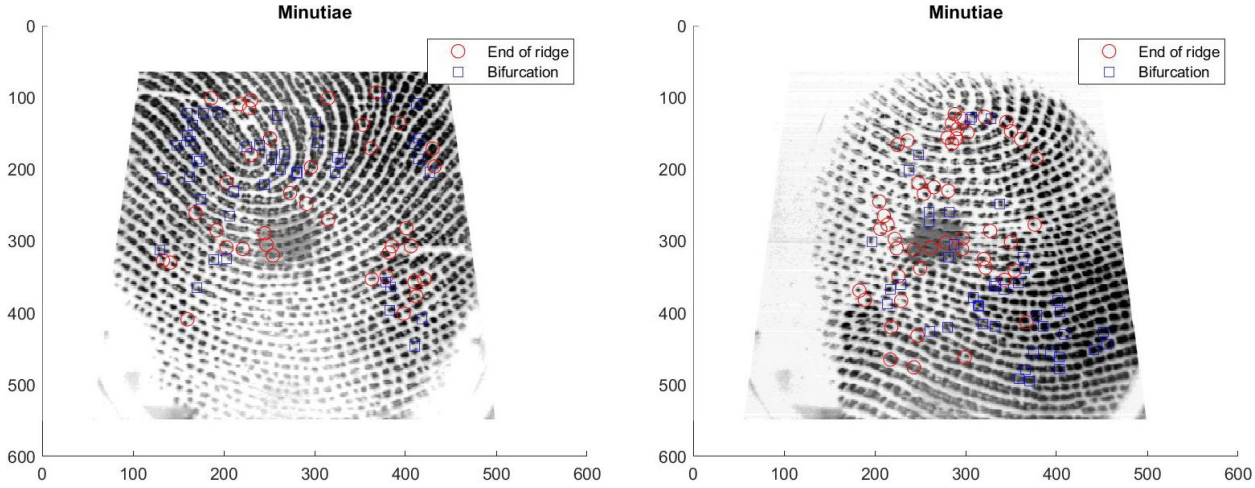

Figure 23. Minutiae analysis on good sample (left) and poor sample (right) with border of 5

2.1.9 Ridge enhancement and skeleton building

Cleanskeleton.m is used to detect holes in the skeleton and clean them up. Visually, there were absolutely no differences in either the good or poor image caused by reducing the blkSize (by increasing the range of i). However, we can observe the effect of not reducing the blkSize enough by setting i to only have a range up to 1. An example of the effect is shown in the figure below:

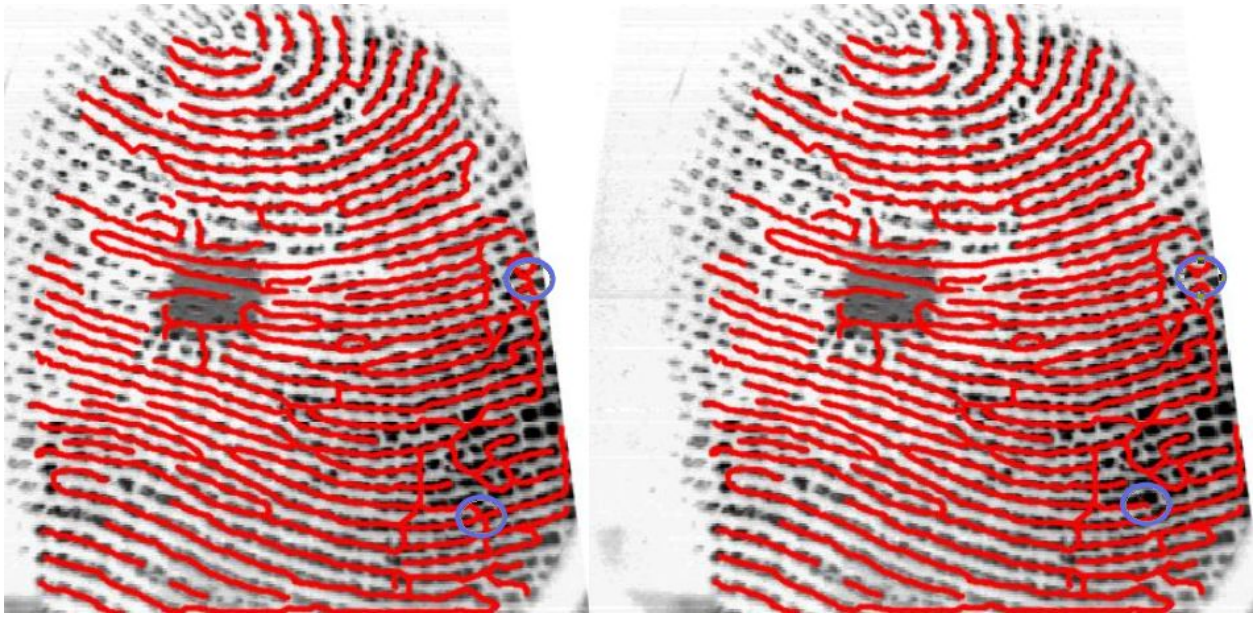

Figure 24. Effect of large blkSize (left) and regular blkSize (right) on skeleton cleanup

When the blkSize isn't small enough, small holes in the skeleton aren't cleaned up, as shown in the purple circled areas in the above figure. However, using a range of i up to 5 is already sufficient to do most of the cleanup, and increasing it further has no visible effect on our samples.

Results of Gabor Filtering

“Results of Gabor filtering on your fingerprints using the adjusted parameters, with illustrations”

Using just the good fingerprint image, the Gabor filter was applied and then adjusted by changing the the following parameters:

- Xsize - Refers to the size of the filter kernel in the x axis, the width of the Gabor filter changes proportionally to Xsize and the number of vertical blocks in the filtered and blocked image changes inversely proportionally to Xsize. This can be seen in figures 26 and 27.
- Ysize - Refers to the size of the filter kernel in the x axis, the height of the Gabor filter changes proportionally to Ysize and the number of horizontal blocks in the filtered and blocked image changes inversely proportionally to Xsize. This can be seen in figures 28 and 29.
- Dx - Refers to the standard deviation of the gaussian envelope in the x axis, the weight of the filter towards the centre of along the y axis is proportional to Dx. In general, this makes the filter more sensitive to longer stretches of a repeated frequency. This can be seen in figures 30 and 31.
- Dy - Refers to the standard deviation of the gaussian envelope in the y axis, the weight of the filter towards the centre of along the x axis is proportional to Dy. In general, this makes the filter more sensitive to wides stretches of a repeated frequency. This can be seen in figures 32 and 33.
- F - Refers to the frequency being looked for in the image (or how close together the little fingerprint ridges are!), the spacing between the high and low bands of the filter is inversely proportional to F. In general the higher F is, the higher frequencies the filter will be suited to detecting. This can be seen in figures 34 and 35.
- A - Refers to the angle at which we will orient our filter, the filter is sensitive to bands of frequency that are parallel to A, assuming 0 rad is due right and angles run counter-clockwise. This can be seen in figures 36 and 37.

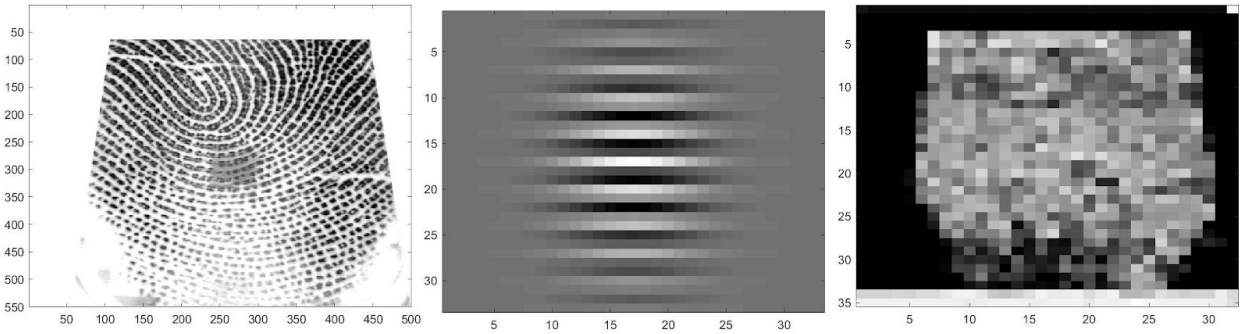

Figure 25. Default Gabor Parameters (Xsize 32, Ysize 32, Dx 8, Dy 4, F 0.3, A 0.0)

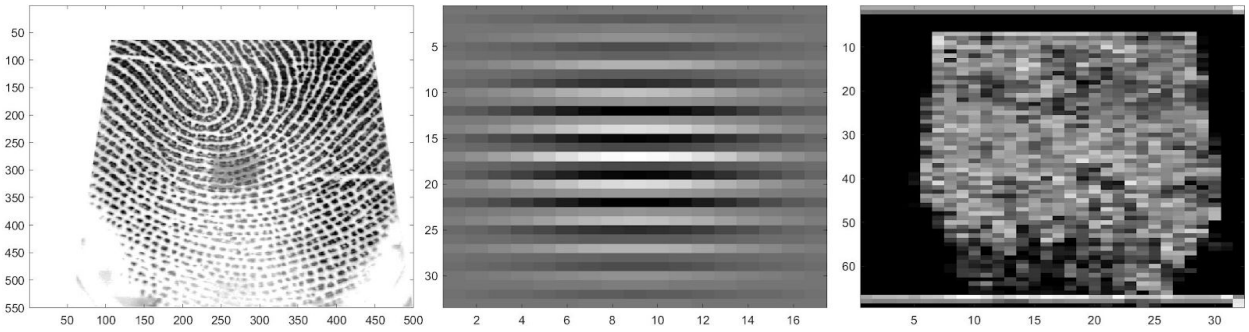

Figure 26. Decreased Xsize Gabor (Xsize 16)

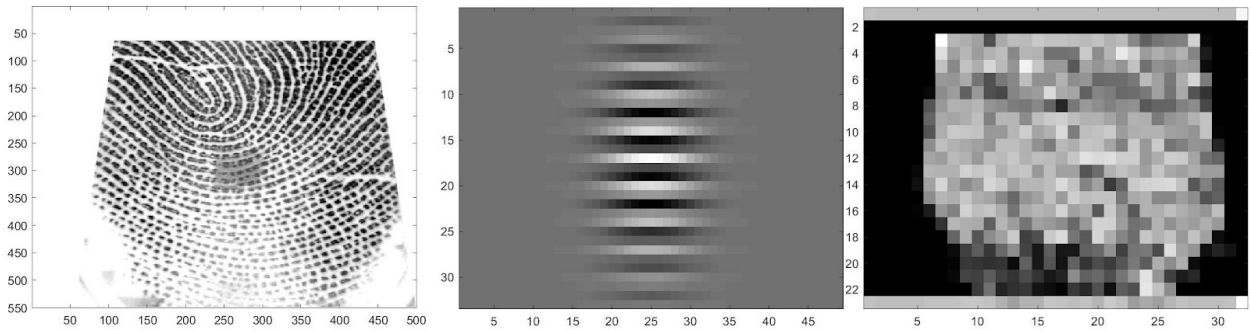

Figure 27. Increased Xsize Gabor (Xsize 48)

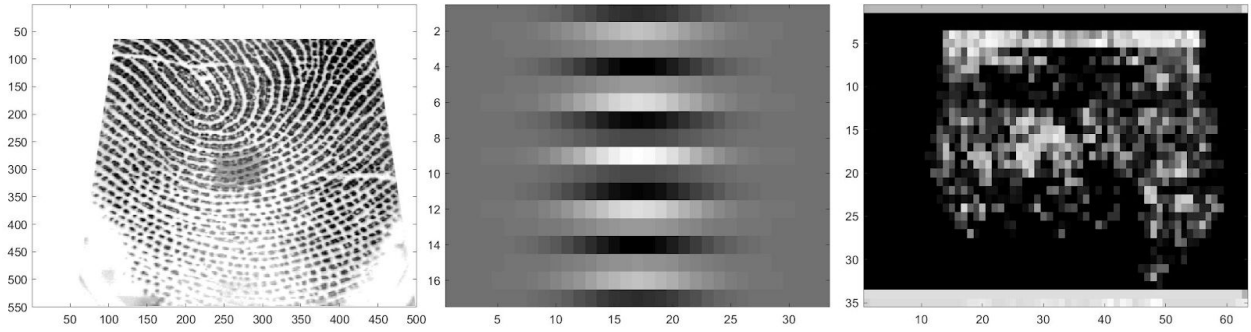

Figure 28. Decreased Ysize Gabor (Ysize 16)

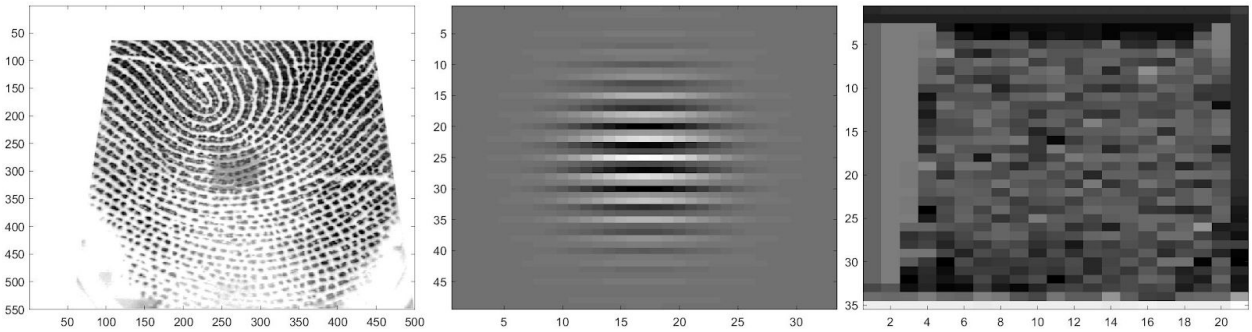

Figure 29. Increased Ysize Gabor (Ysize 48)

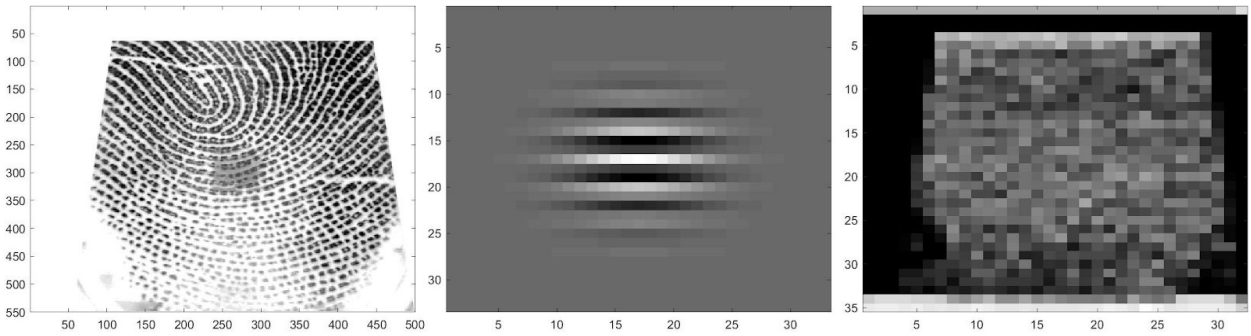

Figure 30. Decreased Dx Gabor (Dx 4)

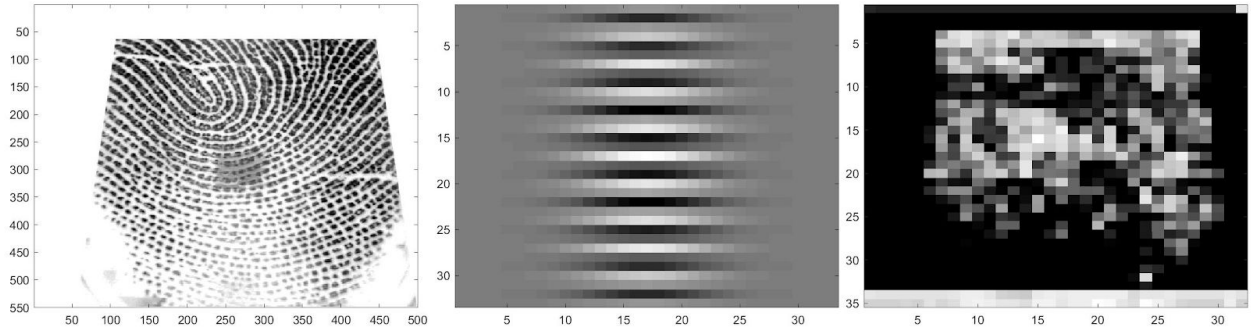

Figure 31. Increased Dx Gabor (Dx 16)

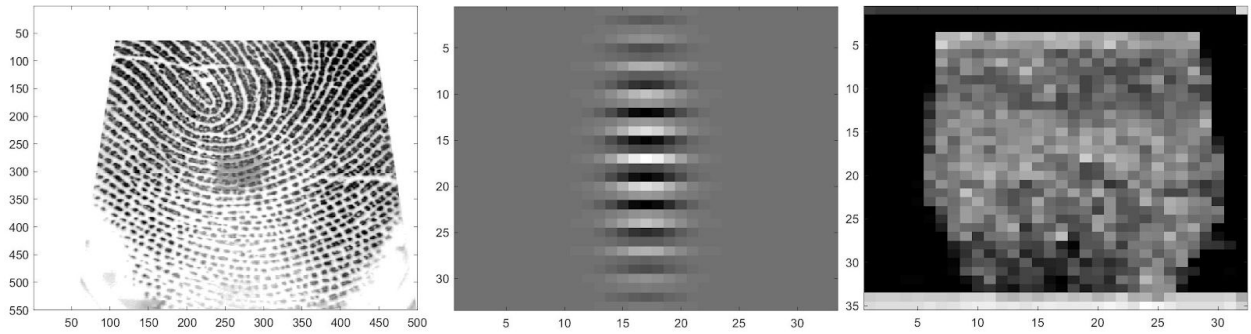

Figure 32. Decreased Dy Gabor (Dy 2)

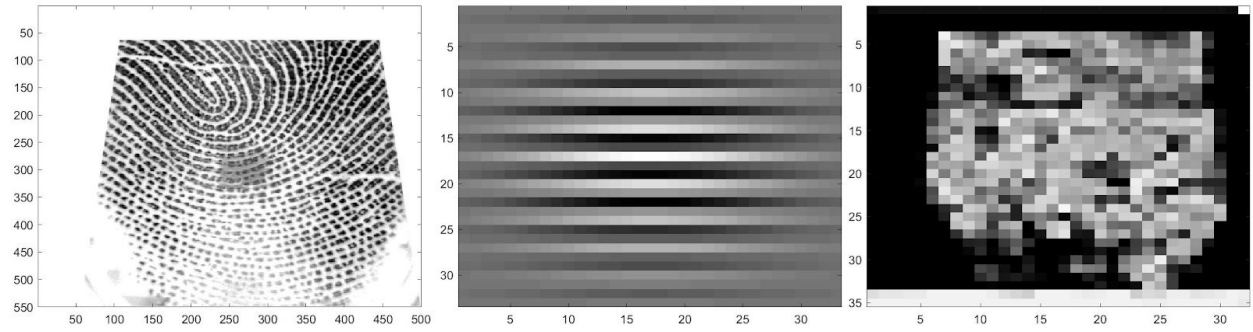

Figure 33. Increased Dy Gabor (Dy 8)

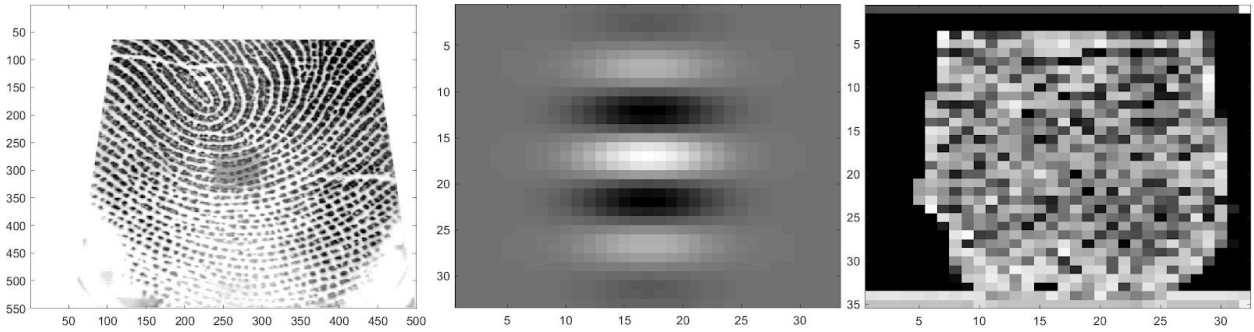

Figure 34. Decreased F Gabor (F 0.1)

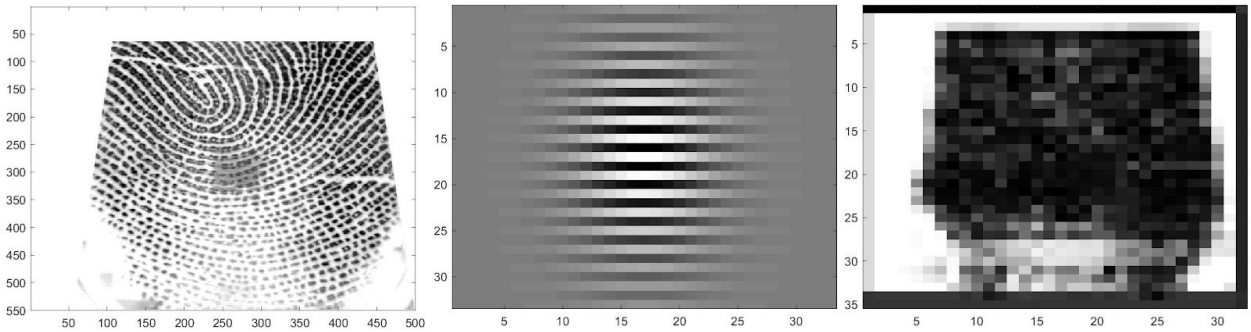

Figure 35. Increased F Gabor (F 0.5)

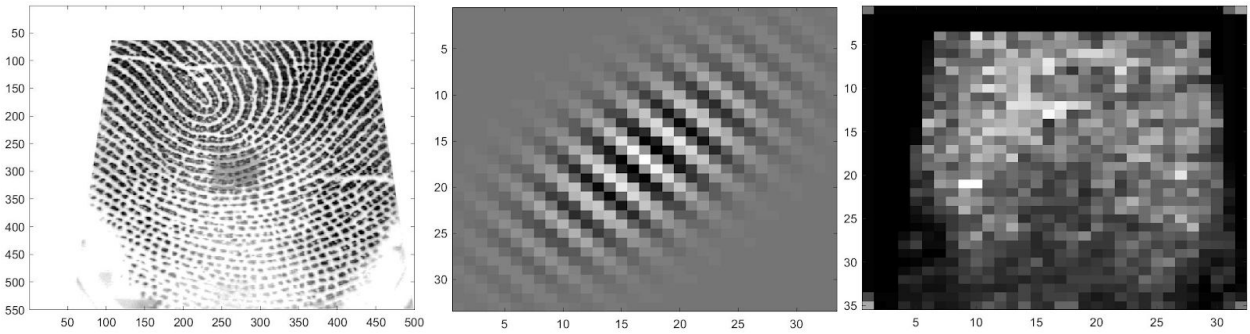

Figure 36. Decreased A Gabor (A -π/4)

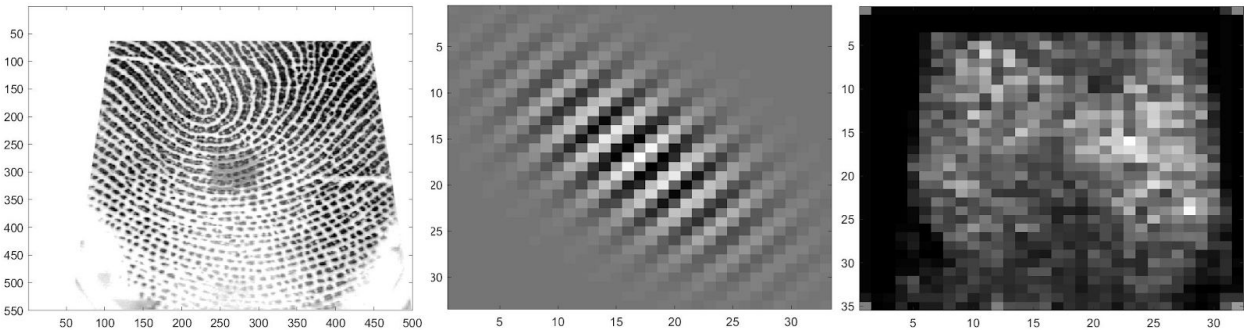

Figure 37. Increased A Gabor (A π/4)

Matching Exercise 1 - Results and Analysis

For each comparison of the probed fingerprint to 9 database templates of the same individual, the scores for the matching method based on Gabor filtering, matches, and mismatches are in the table below. For matching, a threshold of 11.0 was chosen. Note that since the Gabor filtering score works on minimum score, a score must be *below* the threshold to be a match.

Table 2. Matching of Probed Finger to Same Individual - Gabor Filtering

Template No.	**Score**	**Match/Mismatch (*below* threshold of 11.0)**
1	9.079800312	Match (True Positive)
2	9.94250595	Match (True Positive)
3	10.93256954	Match (True Positive)
4	11.2125749	Mismatch (False Negative)
5	11.2125749	Mismatch (False Negative)
6	9.078475919	Match (True Positive)
7	9.874918284	Match (True Positive)
8	8.944531827	Match (True Positive)
9	9.413613561	Match (True Positive)

$$Sensitivity = \frac{TP}{TP + FN}$$
$$Sensitivity = \frac{7}{7+2}$$
$$Sensitivity = 77.8\ \%$$

As a result of this threshold choice, there are 7 true positives and 2 false negatives, for a sensitivity of 77.8%.

Following the same exercise but instead using minutiae-based matching, the results are summarized in the table below.:

Table 3. Matching of Probed Finger to Same Individual - Minutiae-Based Matching

Template No.	Score	Match/Mismatch (above threshold of 0.55)
1	0.6078431373	Match (True Positive)
2	0.5982905983	Match (True Positive)
3	0.5483870968	Mismatch (False Negative)
4	0.5811965812	Match (True Positive)
5	0.5811965812	Match (True Positive)
6	0.2549019608	Mismatch (False Negative)
7	0.59375	Match (True Positive)
8	0.7413793103	Match (True Positive)
9	0.6542056075	Match (True Positive)

$$Sensitivity = \frac{TP}{TP + FN}$$
$$Sensitivity = \frac{7}{7 + 2}$$
$$Sensitivity = 77.8\ \%$$

As a result of this threshold choice, there are 7 true positives and 2 false negatives, for a sensitivity of 77.8%.

The choice of the matching threshold affects the number of matches and mismatches by determining how good of a score is required to be called a “match”. In this exercise the thresholds were chosen somewhat arbitrarily and the impact of the choice is illustrated in the following two figures. For either type of score, scores below the threshold (red line) are regarded as mismatches (false negatives) and scores above the threshold are regarded as matches (true positives).
Clearly, increasing the threshold will increase the number of mismatches, and vice versa. However theoretically, having a higher threshold should make it more difficult for imposters to be matched as genuine.
Note that a good starting value for a threshold is the EER, or the point where FAR = FRR. However, since this exercise doesn’t include any opportunity for false acceptance, we’re unable to choose that point.

Figure 38. Effect of Gabor Score threshold on the number of matches or mismatches

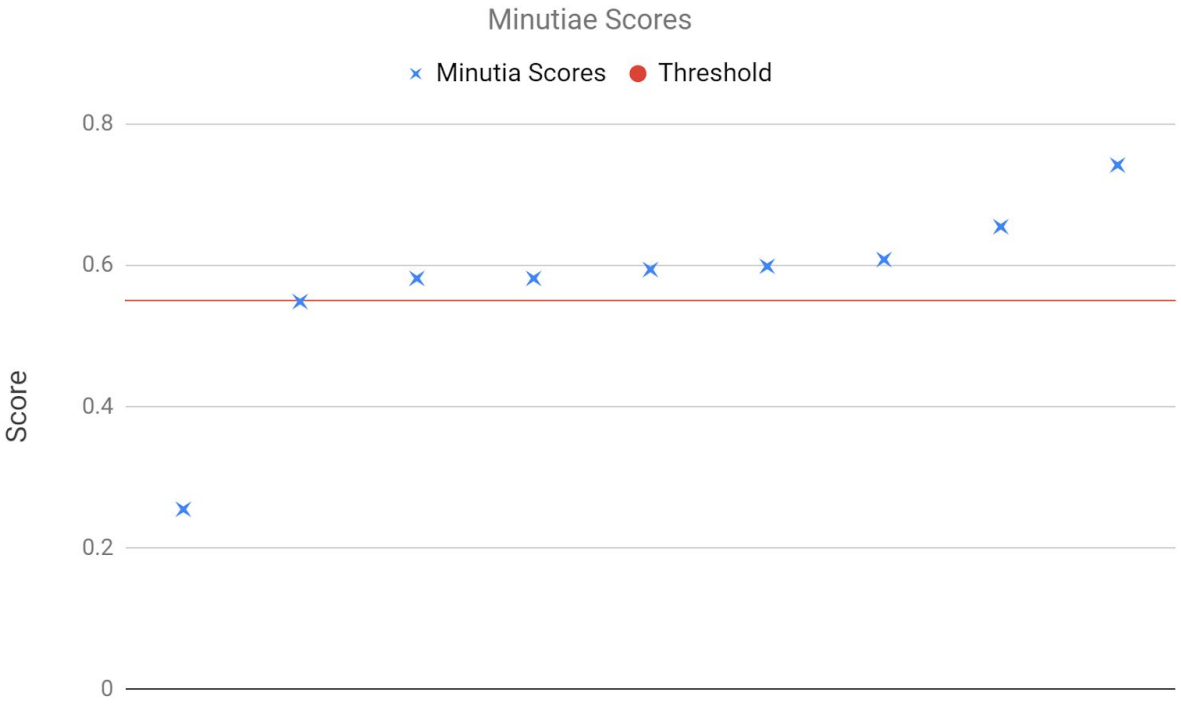

Figure 39. Effect of Minutia Score threshold on the number of matches or mismatches

Matching Exercise 2 - Procedure, Results, and Analysis

Using the minutia-based approach, we'll take one image of Andy's thumb as the probed fingerprint, and compare it against 1 image of the same thumb, and 10 images of Patrick's thumb. The matching scores for each comparison are ordered and shown in the table below, and the top ranked score for the minutia-based approach is 0.495726.

Table 4. Matching of Probed Finger to Templates - Minutia-Based Approach

Template Owner-Number	**Rank**	**Score**	**Match/Mismatch - Using Score of Rank 1 as Threshold (0.495726)**
Andy-1	1	0.495726	Match - True Positive
Patrick-8	2	0.145985	Mismatch - True Negative
Patrick-7	3	0.135593	Mismatch - True Negative
Patrick-4	4	0.130841	Mismatch - True Negative
Patrick-6	5	0.125	Mismatch - True Negative
Patrick-3	6	0.117647	Mismatch - True Negative
Patrick-5	7	0.110092	Mismatch - True Negative
Patrick-1	8	0.095238	Mismatch - True Negative
Patrick-2	9	0.095238	Mismatch - True Negative
Patrick-10	10	0.067797	Mismatch - True Negative
Patrick-9	11	0.045977	Mismatch - True Negative

The probed fingerprint only matches with the same finger, so there is a true match. We can therefore use the matching score from the first rank as our threshold

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
$$Accuracy = \frac{1 + 10}{1 + 10 + 0 + 0}$$
$$Accuracy = 100\ \%$$

Using this threshold, the matching of the probe to different fingers is summarized in the table above. There is 1 true positive and 10 true negatives, which is a good outcome with 100% accuracy.

Because we had a true match (the probed fingerprint matched only matched against the same figure, and was the highest score), we could use the score from that comparison as our threshold. As a result, we only had true positives and true negatives. The threshold was chosen using a ranking-based approach, but since there is such a division between the scores for the matches and mismatches, it might make sense in a real application to lower the threshold slightly. That way, there is a somewhat lower false rejection rate for the genuine user. That is, given this sample of data, the threshold does not need to be as high as it is, and it might cause false rejections.

Matching Exercise 3 - Procedure, Results, and Analysis

In this exercise, one of Andy's fingerprints is used as the probe, and the database consists of 9 of Andy's same finger, and 10 of Patrick's finger. Using the Minutia-based approach, the probe is scored against each of the 19 templates in the database.
Each of the scores are sorted and summarized in the table below. Using the the ranking approach, we'll choose the score for rank 8 as the threshold to maximize the number of true positives and minimize false positives and false negatives.

Table 5. Matching of Probed Finger to Templates - Minutia-Score Ranking Approach

Template Owner-Number	**Rank**	**Score**	**Match/Mismatch - Using Score of Rank 8 as Threshold (0.451613)**
Andy-8	1	0.603448	Match - True Positive
Andy-9	2	0.542056	Match - True Positive
Andy-1	3	0.529412	Match - True Positive
Andy-2	4	0.495726	Match - True Positive
Andy-7	5	0.46875	Match - True Positive
Andy-4	6	0.461538	Match - True Positive
Andy-5	7	0.461538	Match - True Positive
Andy-3	8	0.451613	Match - True Positive
Patrick-8	9	0.145985	Mismatch - True Negative
Patrick-7	10	0.135593	Mismatch - True Negative
Patrick-4	11	0.130841	Mismatch - True Negative
Patrick-6	12	0.125	Mismatch - True Negative
Patrick-3	13	0.117647	Mismatch - True Negative
Patrick-5	14	0.110092	Mismatch - True Negative
Patrick-1	15	0.095238	Mismatch - True Negative
Patrick-2	16	0.095238	Mismatch - True Negative

Andy-6	17	0.078431	Mismatch - False Negative
Patrick-10	18	0.067797	Mismatch - True Negative
Patrick-9	19	0.045977	Mismatch - True Negative

Using this approach, there are 8 true positives, 10 true negatives, and only 1 false negative. The false negative came from a very poor score caused by poor alignment of the probe and the template. Otherwise, there's a good separation between the minutia scores for the prints of the same individual and the other individual. We can calculate the accuracy and sensitivity as well:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
$$Accuracy = \frac{8 + 10}{8 + 10 + 0 + 1}$$
$$Accuracy = 94.7\ \%$$

$$Sensitivity = \frac{TP}{TP + FN}$$
$$Sensitivity = \frac{8}{8 + 1}$$
$$Sensitivity = 88.9\ \%$$

Given that the score for the false negative is so low, the threshold should not be lowered to accommodate it, since that would allow many templates from the other individual to be falsely accepted.

Conclusion

In this lab we examined fingerprint identification and verification techniques: minutiae extraction and Gabor filter based comparison.

In the first part we analyzed the feature extraction techniques used for both minutiae extraction and Gabor filter based feature extraction. This mainly consisted of modifying configuration values within the pipeline and analyzing the results on both intermediates and the final feature extraction results.

In the second part we used both minutiae and Gabor filter comparison techniques to examine the accuracy and sensitivity of identification and verification systems for fingerprints.

Appendices

The code used for this lab is attached in Lab45code folder, and some snippets are included in this appendix.

The code used for this lab is attached in the folder Lab45code. Lab4Fingerprint1.m and Lab4Fingerprint2.m were adjusted slightly for the pre-processing experiments in Part I, and a version of Lab4Fingerprint1.m called Lab4Fingerprint1Auto.m was developed to help speed up computation. A file called GaborShow.m was added for computing and rendering a gabor filter with varied parameters. Laboratory45.m includes a code snippet for running Lab4Fingerprint1Auto.m, and the code for performing matching exercises 1-3.

Calling Lab4FingerprintAuto.m

```
Lab4Fingerprint1Auto(PatrickThumbs{22}, "img/22_Default/skeleton_1");
```

Calling GaborShow.m

```
GaborShow(PatrickThumbs{27}, 32, 32, 8, 4, 0, 0.5);
```

Condensed Minutiae Extraction using PreProcess

```
function [output_image] = PreProcess(input_image)
    %%% this function pre-processes an image before being scored for matching,
    %%% based on the pre-processing procedure done in Lab5Fingerprint2.m.
    output_image.imOrig = input_image;
    %%%disp('Segmentation');
    output_image = segmentimage(output_image);
    %%%disp('Orientation array');
    output_image = computeorientationarray(output_image);
    %%%disp('Finding the singularity point');
    output_image = findsingularitypoint(output_image);
    %%%disp('Local frequencies');
    output_image = computelocalfrequency(output_image, output_image.imOrig);
    %%%disp('Filtering');
    output_image = enhance2ridgevalley(output_image);
    %%%disp('Skeleton cleaning');
    output_image = cleanskeleton(output_image);
    %%%disp('Finding minutiae');
    output_image = findminutia(output_image);
end
```

Matching Exercise 1

```
%-------------------------------------------------------------------------
%-------- Select probe fingerprint
%-------------------------------------------------------------------------
% Let's use Andy's 16th print as the probe.
img = AndyThumbs{16};
disp('Processing Matching Exercise 1...');
Fp1 = PreProcess(img);
GaborScores = {9,1};
MinutiaScores = {9,1};

starting_idx = 17;
for i = 1:9
    j = i + starting_idx - 1;
    %-------------------------------------------------------------------------
    %-------- Select comparison fingerprint
    %-------------------------------------------------------------------------
    img = AndyThumbs{j};
    Fp2 = PreProcess(img);

    %-------------------------------------------------------------------------
    %---- Fingerprint comparison using two methods: Gabor and local matching
    %-------------------------------------------------------------------------
    Score1 = MatchGaborFeat(Fp1,Fp2);
    fprintf('Score 1 for print %d using Gabor features: %1.2g\n', i, Score1);

    threshold2=12; % Using 12, as it was arbitrarily suggested in Lab5Fingerprint2.m

    Fp2=align2(Fp1,Fp2);
    Score2=match(Fp1.minutiaArray, Fp2.minutiaArrayAlign, Fp1.imSkeleton, Fp2.imSkeletonAlign,threshold2);
    fprintf('Score 2 for print %d for minutiae : %1.2g\n', i, Score2);
    GaborScores{i} = Score1;
    MinutiaScores{i} = Score2;
end
% Lazy way to print the scores and get them in a column
GaborScores = GaborScores'
MinutiaScores = MinutiaScores'
```

Matching Exercise 2

```
%-------------------------------------------------------------------------
%-------- Select probe fingerprint
%-------------------------------------------------------------------------
% Let's use Andy's 16th print as the probe.
probe = PreProcess(AndyThumbs{16});

% Set up our little database, with one of Andy's prints at the start
database_images{1} = PreProcess(AndyThumbs{18});
database_images{1} = align2(probe, database_images{1});

j = 2
for i = 1:30
    % Grab 10 of Patrick's thumbs
    fprintf('Processing and aligning print number %d\n', i);
    img2 = PreProcess(PatrickThumbs{i});
    % Try and align the image, I suppose
    try
        img2 = align2(probe,img2);
        database_images{j} = img2;
        j = j + 1;
        if(j == 12)
            break;
        end
    catch ME
        fprintf('Could not align print number %d\n', i);
    end
end

MinutiaScores = {11,1};

for i = 1:11
    %-------------------------------------------------------------------------
    %---- Fingerprint comparison using... local matching.
    %-------------------------------------------------------------------------

    threshold2=12; % Using 12, as it was arbitrarily suggested in Lab5Fingerprint2.m

    img2=align2(probe,database_images{i});
    Score_m=match(probe.minutiaArray, img2.minutiaArrayAlign, probe.imSkeleton, img2.imSkeletonAlign,threshold2);
    fprintf('Score for print %d for minutiae : %1.2g\n', i, Score_m);
    MinutiaScores{i} = Score_m;
end
% Lazy way to print the scores and get them in a column
MinutiaScores = MinutiaScores'
```

Matching Exercise 3

```
%-------------------------------------------------------------------------
%-------- Select probe fingerprint
%-------------------------------------------------------------------------
% Let's use Andy's 16th print as the probe.
probe = PreProcess(AndyThumbs{16});

% Set up our little database, with 9 of Andy's prints at the start
j = 1
% Grab 9 of Andy's thumbs
for i = 17:30
    fprintf('Processing and aligning Andys print number %d\n', i);
    img2 = PreProcess(AndyThumbs{i});
    % Align the image
    try
        img2 = align2(probe,img2);
        database_images{j} = img2;
        j = j + 1;
        if(j == 10)
            break;
        end
    catch ME
        fprintf('Could not align Andy print number %d\n', i);
    end
end

% Grab 10 of Patrick's thumbs
for i = 1:30
    fprintf('Processing and aligning Patricks print number %d\n', i);
    img2 = PreProcess(PatrickThumbs{i});
    % Try and align the image, I suppose
    try
        img2 = align2(probe,img2);
        database_images{j} = img2;
        j = j + 1;
        if(j == 20)
            break;
        end
    catch ME
        fprintf('Could not align Patricks print number %d\n', i);
    end
end

MinutiaScores = {19,1};
```

```
MinutiaScores = {19,1};

for i = 1:19
    %-----------------------------------------------------------------------
    %---- Fingerprint comparison using... local matching.
    %-----------------------------------------------------------------------

    threshold2=12; % Using 12, as it was arbitrarily suggested in Lab5Fingerprint2.m

    img2=align2(probe,database_images{i});
    Score_m=match(probe.minutiaArray, img2.minutiaArrayAlign, probe.imSkeleton, img2.imSkeletonAlign,threshold2);
    fprintf('Score for print %d for minutiae : %1.2g\n', i, Score_m);
    MinutiaScores{i} = Score_m;
end
% Lazy way to print the scores and get them in a column
MinutiaScores = MinutiaScores'
```