The University of Calgary
Department of Electrical and Computer Engineering

ENCM 509 - Fundamentals of Biometric Systems Design

Laboratory Experiment #4 – 5

Fingerprint Biometrics:
Part I (lab 4): Preprocessing and Feature Extraction
Part II (lab 5): Matching for Identification

1 Introduction

The purpose of Part I this laboratory exercise is to investigate the fingerprint processing.

The input data for this lab is your fingerprints collected using the provided DigitalPersona USB devices.

The `.m`-files needed for fingerprint image processing and matching, including Lab4Fingerprint1.m and Lab5Fingerprint2.m, for can be found on N drive in \ENCM\509\lab4-5.

The fingerprint processing and feature (minutia) extraction involves:

- Orientation estimation,
- Ridge orientation,
- Applying Gabor filters on the fingerprint image which have frequency-selective and orientation-selective properties. This enhances the ridges.

These steps will prepare data for fingerprint matching (Part II of this exercise).

The purpose of Part II of this exercise is to investigate a matching algorithm that uses a score to compare images, represented by their feature vectors. The score is a numerical data expressing the similarity of two feature vectors. The idea of matching score is the basis for a simple identification system, performing one-against-many comparisons.

Such a system is created using the following steps:

- Collecting the data for the database ("enrollment"). In this example, we assume we have a data set of 10–30 templates. It should be noted that a database may contain two or more templates of the same person. For example, you can collect 10 fingerprints of your right thumb (or other finger), and 10 of your left one to store as a database (total 20). Or you can have 10 of your fingerprints and 10 of your partner's. You can leave one out to be a "live" (probed) template, or acquire one extra.

- Identification of a newly submitted, or probed, fingerprint against your database of 10 – 20 others. This is performed using so-called "matching", or comparison of the feature vectors, - the database and the probed one (one to be identified). The score is calculated for each pair "database fingerprint – probed fingerprint". Given a database of 20 fingerprints, we have to have 20 pairs, and therefore, 20 numerical scores.

We will use two types of minutiae, ridge end and bifurcation. The template is a feature vector, containing a vector of minutiae, each represented as triplets x, y, θ (x-location, y-location and orientation θ). A fingerprint matching algorithm is used to produce a similarity score between two pairs of fingerprint samples. Traditionally, the fingerprint matching algorithm computes a relative distance (more often, the absolute difference, or Euclidean distance) between two samples of the fingerprint.

The matching can be used in verification or identification. In both applications, you may need to know the threshold for your score value, needed for the appropriate conclusion about the matching. In the identification, you may need to detect the highest score (single answer to matching), or perform ranking, and select several high scores that are above some threshold.

If you have N templates for each of M fingers in the database, you have to calculate $N \times N$ scores within this set of templates, and then choose the best score. All the matches against the same fingers will be True Positives, the mismatches against the same fingers are False Negatives. The matches against different fingers are False Positives, and the mismatches are True Negatives. You will need to record all these numbers in your Report.

There are many fingerprint matching approaches. We will investigate two of them: one based on the Gabor filtered features, and another one based on the Minutiae-Skeleton matching with prior alignment. In both cases, matching is based on calculating the Euclidean distance between the two templates: the database one and the probed one.

2 The laboratory exercise, Part I

2.1 Fingerprint acquisition

The acquisition of the fingerprints will be performed on the USB Digital Persona U.are.U 4500 fingerprint sensors (reader), and the software developed in the Biometric Technologies Laboratory at UofC using the Digital persona SDK. The software creates "raw" data in the `.bmp` (or other) format. Connect the reader it to the computer using a USB port. The device will be ready once red light flashes. Open the program `digitalPersona.exe` on drive N in *\ENCM\509\DigitalPersona* (press "Run"). The acquisition can be performed by placing your finger on the sensor surface, and it can be done as many times as you want to get the good quality print shown in the interface window. To save the image, choose option "File" and then "Save" (save on your drive H or a USB memeory). You can clean the reader with a Scotch tape (yes, press the scotch tape on the top of the sensor and lift it, – as good as new).

To create a database, collect:

- 30 of your left (or right) thumb or other finger images,
- 30 of your other thumb or the other finger.

The quality of data is affected by the sensor quality, the skin condition (dryness, cuts), the quality of the procedure (such as correct placement of the finger), humidity etc. You will need one image of good quality and one of poor quality for analysis of fingerprint processing procedures; the remaining fingerprints will be used for matching.

2.1.1 Reading images in Matlab

Data can be read into Matlab using the `imread` operator. For example, you can read a bmp file (usually, in general RGB form) and convert it to a gray-scale image.

```
img = imread('YourImage');
img=rgb2gray(img);
```

2.1.2 Fingerprint image processing

The fingerprint processing is implemented in the demo `Lab4Fingerprint1.m` file, which calls other functions available on the N drive.

The below algorithm can be traced using the demo (`file Lab4Fingerprint1.m`). After selecting the fingerprint image, it is segmented by isolating the foreground from the background. Then, the ridge orientation is calculated, and the singularity points (core and delta) are localized.

The fingerprint image processing procedure as well as the minutiae extraction involves the following steps:

- Segmentation (`segmentimage.m`): extracts fingerprint area from background and finds the contour.
- Orientation estimation : computes orientation array at every pixel at the original image using (`computeorientationarray.m`).
- Find singularity points : finds core, delta, etc. (`findsingularitypoint.m`)
- Ridge frequency estimation : computes local frequencies (`computelocalfrequency.m` using `frequest.m`) ,
- Gabor filtering : implements directional filters to enhance ridges (`enhance2ridgevalley.m` using `choosefrequency.m`).
- Thinning and skeleton cleaning : (`cleanskeleton.m` and `Thin.`m), and
- Find minutiae : `findminutia.m`.

2.1.3 Image pre-processing exercise

Explore Matlab possibilities to enhance the image.

2.1.4 Intensity enhancement

Modify Lab4Fingeprint1.m demo by adding histogram equalization or modified (adaptive) one:

- `histeq()` ,
- `adapthisteq()`,

To create the pixel intensity histogram for visual inspection, use

- `imhist(img)`

2.1.5 De-noising

Modify Lab4Fingeprint1.m demo by adding de-noising filters such as Matlab functions from Image Processing Toolbox, for example:

- `imadjust()`,
- `wiener2()`, Wiener filter,
- `medfilt2()`, median filter.

Apply them after reading and converting the image to gray-scale (`img=rgb2gray(img)`), for example, using

- `img = imadjust(img);`

Apply various filters and compare the results of minutiae extraction (last step in the demo) for the fingerprint images with and without pre-processing.

2.1.6 Segmentation

Run demo Lab4Fingerprint1.m to investigate the results of segmentation. Note that `segmentimage.m` uses filter `bw = medfilt2(bw, [16 16]);` with parameters [16 16] and `gradDev > 30`. What happens if those parameters are changed?

2.1.7 Orientation and singularity points

Demo Lab4Fingerprint1.m calls `computeorientationarray.m` that uses parameters `blkSize = 10`. What happens if this block size changes?

It also calls `findsingularitypoint.m`. How many singularity points are found in your fingerprint?

2.1.8 Frequency of ridges

Demo Lab4Fingerprint1.m calls `computelocalfrequency.m` that uses parameters `border = 30`. Will decreasing of this parameter influence the outcomes of the further steps (minutia finding)?

2.1.9 Ridge enhancement and skeleton building

Demo Lab4Fingerprint1.m calls `enhance2ridgevalley.m` that calls Gabor filter and also create skeleton used for the next step (cleaning and thinning). The same Gabor filter is also used for feature extraction in Lab 5. Mode details on the Gabor are studied in the next Section of this lab. `cleanskeleton.m` uses different block sized in its morphological operation blkSize to detect holes

```
   skeleton = Cs.skeleton;
for i = 1:5
blkSize = round(Cs.holeBlkSize/i);
skeleton = blkproc(skeleton, [blkSize blkSize],@ removehole);
skeleton = Thin(imcomplement(skeleton), 'Inf');
end
```

Will reducing the size (by using `i` up to 6 or more) improve the skeleton quality?

2.1.10 Gabor filter

Step 5 of the demo involves ridge detection using a 2D Gabor filter. Let us explore this special filter in detail.

A Gabor filter belongs to a tuned band pass filter bank. It has a Gaussian transfer function in the frequency domain. Thus, the Gabor filter is a Gaussian, and in case of 2D, it is modulated by a complex sinusoid (with centre frequencies along x and y-axes respectively), and is characterized by the frequency, `f` and angle, `angle`. Gabor filters are used mostly for shape detecting and feature extracting in image processing.

Example:

```
function gabor = GaborFilter(img, tx, ty, angle, f)
x = [-16:16]; y = [-16:16];
[x, y] = meshgrid(x,y);
xp = sin(angle) .* x + cos(angle) .* y;
yp = sin(angle) .* y - cos(angle) .* x;
gabor = exp(-0.5 * ((xp.^2 / tx^2) + (yp.^2/ty^2))).* cos(2 * pi * f .* xp);
```

Here, the filter is applied to the image `img`, the size of the filter is specified by the "window" we choose for the image processing, `tx` and `ty`, are filter parameters, `angle` specifies an orientation (usually, one of 8), and frequency `f` (computed using local ridge frequencies for the given image).

Function `imfilter` below applies the filter, `gabor`, to the image, `img`:

```
        imgabor = imfilter(img, gabor,'replicate','same');
```

The filter itself can be illustrated by plotting the 2D image for various (8) orientations.

Example:

```
 norient=8;
xsize=32;  % filter size on x
ysize=32;  % filter size on y
dx=8; dy=4; % stabdard deviation of gaussian envelope
f = 0.1;  % frequency
angle=[0:pi/8:pi-pi/8];
n=1;  %initiate the number of filters
figure, hold on
for a=angle
f{n}=GaborFilter(32,32,8, 4, a, f); %build the filter
subplot(1,norient,n),imagesc(f{n}), colormap gray
n=n+1;
end
```

Function `GaborFilter` can be found in the directory on the N drive, *\ENCM\509\lab4*. Apply the Gabor filter to your fingerprint image. Note that the image must be read first (see Lab4Fingeprint1.m demo) and converted to gray-scale double-precision format.

Example:

```
 [namefile,pathname]=uigetfile({'*.bmp','IMAGE Files(*.bmp)'});
[img,map]=imread(strcat(pathname,namefile));
Fp.imOrig = img;
figure(1), imagesc(Fp.imOrig),colormap gray, title('Original Image');
imoriginal = Fp.imOrig;
```

You will have to convert the unsigned gray-scale representation to double-precision: `img = mat2gray(double(imoriginal));`

The example below shows how to draw several images of various angles for the Gabor filter (image test1.bmp was used).

Example:

```
 [namefile,pathname]=uigetfile({'test1.bmp','IMAGE Files (*.bmp)'});
[img,map]=imread(strcat(pathname,namefile));
Fp.imOrig = img;
figure(1), imagesc(Fp.imOrig),colormap gray, title('Original Image');
xsize=32; ysize=32;
tx=4; ty=4;
f=0.11; angle=pi/4;
fi=GaborFilter(xsize,ysize,tx, ty, angle, f);
I1=Fp.imOrig;
If1(:,:)=imfilter(I1,fi);
F1(:,:)=blkproc(squeeze(If1(:,:)),...
...  [xsize/2 ysize/2],inline('abs(mean2(x)-std2(x))'));
figure(2), plot(2), imagesc(fi), colormap gray;
figure(3), plot(2), imagesc(squeeze(F1(:,:))), colormap gray;
```

This will create three figures: the original one (we used `test1.bmp` image for illustration purpose, with four sets of lines: vertical, horizontal and two diagonals), the 2D image of the filter, and the filtered image. As shown in Figure 1, given the angle $pi/4$, the filter extracts the lines oriented along with the diagonal (right-upper corner to left-lower corner).

Use the demo file Lab4Fingerprint2gab.m and modify it to perform Gabor filtering on your fingerprint image, adjust the parameters of the filter. What is the impact of changing the parameters of the Gabor filter?

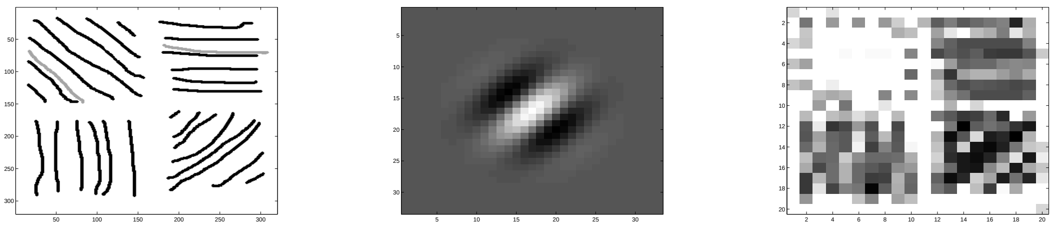

Figure 1: The original image, the filter and the filtered image.

2.2 Fingerprint matching

The matching in this lab is based on two approaches:

- Features obtained based on the Gabor filtering, and
- Feature obtained from localization of the minutiae (two types only: ridge end, and bifurcation).

In the first case, script `MatchGaborFeat.m` is used. Matching score is calculated as the mean of the differences between two feature vector, while the features are calculated using the absolute deviation from the mean on the 16×16 blocks of the Gabor filtered images (there are eight filters, corresponding to eight different angles used to build the Gabor filter, and eight filtered images per one fingerprint).

In the second case, script `match.m` is used. The ridge skeleton, and the coordinates of the minutiae are considered for comparison. For this, image alignment is required (script `align2.m` is used for that). Fingerprint alignment is executed as follows: once a fingerprint is pre-processed, it is necessary to find its corresponding position in the database image in order to make a comparison. The solution consists in calculating the 2-dimensional correlation for every possible location of the fingerprint within the database image. The highest correlation factor determines the part of the reference image that matches the best the probed fingerprint.

Matching score determines the similarity of two fingerprint features Fp1 and Fp2. The minutiae features are compared as follows: a minutia m_i in Fp1 and a minutia m_j in Fp2 are matched if the Euclidian distance between them is smaller than some pre-determined threshold.

We assume that you have 10 impressions of your finger (e.g. left thumb) and 10 impressions of other fingers (yours or your partner).

2.2.1 Matching Exercise 1

Use one of your fingerprints to be used as the "live", or probed one. Use the remaining 9 other impressions of the same finger as a database records. Therefore, we have a database with 9 templates representing the same "individual". Use the demo file Lab5Fingerprint2.m that perform 1 to 1 matching. Develop a simple identification procedure (use loops that repeat 1 to 1 matching several times), to match a probed fingerprint of your finger (e.g. left thumb) against 9 different impressions of the same finger.

For each comparison,

- Record the scores (use a table form) for the matching method based on Gabor filter (it works based on the minimum score). Record the number of matches against the same fingers (True Positives), the mismatches against the same fingers (False Negatives).

- Perform the same exercise for the minutiae based matching (it works based on the maximum score; note the difference compared to the Gabor score). Record True positives and False negatives.

- How the choice of the threshold affects the matching? Use numbers or graphs to illustrate the answer.

2.2.2 Matching Exercise 2

Use one of your fingerprints (e.g. left thumb) to be used as the "live", or probed one. Use the database with one impression of the same finger and 10 of your other finger(s). Therefore, we have a database with 11 templates one of which represents the same "individual", and the rest of templates represent another "individual(s)".

Develop (and code in Matlab) a ranking identification procedure for minutiae-based matching only. A sample of such procedure is as follows:

- Record the matching scores for each fingerprint comparison; use some index corresponding to the fingerprint number. Sort the matching scores: the top score is the maximum score for the Minutia based approach.

- Do you have a true match? It means there is a match of the probed fingerprint against the only one of the same finger. If your true match is not on top of your ranking, which rank is it? Use this rank's score as a "thresholding score" that separated a top-ranked group of the closest matches, and the others.

- Record the number of matches against different fingers as False Positives, and the mismatches as True Negatives. Some false positives may belong to the top ranked group.

- Draw conclusions upon your experiment, threshold choice etc.

2.2.3 Matching Exercise 3

Now, use one of your fingerprints as a probed one, and the database with 9 impression of the same finger and 10 of your other finger(s). Therefore, we have a database with 19 templates, 9 of which represents the same "individual", and the rest of templates represent another "individual(s)".

Implement an identification procedures based on ranking approach for the Minutia score only. For that, you will need to sort the scores from 1 to 19, and rank them. Record all related numbers matches/ mismatches. Draw the conclusions.

3 Laboratory Report (10 marks total)

In your report, include:

- A brief intro and conclusions (0.5 marks).
- Your experiment from part I involving the good quality and poor quality fingerprints. Results of Gabor filtering on your fingerprints using the adjusted (by you) parameters, with illustrations (1 mark).
- Your results and analysis for Matching Exercise 1, illustrated with graphs (2 marks).
- Your identification procedure based on ranking approach and code for Matching Exercise 2, and analysis of the results (2 marks),
- Your identification procedure based on ranking approach and code for Matching Exercise 3, and analysis of the results (2 marks).
- The created Matlab fragments, in the text or as attached .m files where appropriate (2.5 marks).

4 Acknowledgments

We wish to thank research students who contributed to the projects implemented in the Biometric Technologies Laboratory. The acquisition executable `digitalPersona.exe` was created in the BT Lab based on the Digital Persona SDK. The implementation of some fingerprint processing protocols according to NIST (National Institute of Standardization, USA) are credited to K. Kryszczak, EPFL.

Svetlana Yanushkevich.
January 31, 2019