



Digital analysis of fingerprints

Alaa hasan yazbek
Daniel Gabai
Fethi Harkat
Mahtab Khademalhosseini
Thu Thao Huynh

Professor: Christophe Picard

January 2020

Contents

1	Abstract	3
2	Introduction	4
2.1	Related Works	4
2.2	Overview of the implementation	5
2.2.1	Architecture of the code	5
2.2.2	How to launch the code?	6
2.2.3	Tools	6
3	Methodology	8
3.1	Symmetry	8
3.2	Coefficient for Pressure	9
3.2.1	Results and Analysis	10
3.3	Geometrical Warp	11
3.3.1	Find the center of rotation(Centroid)	11
3.3.2	Angle of rotation	11
3.3.3	Make the rotation	13
3.4	Linear Filtering	17
3.4.1	Direct convolution	17
3.4.2	Fast Fourier Transform(FFT)	18
3.4.3	Results and Analysis	19
4	Optimization	20
4.1	Optimization for translation along x axis	20
5	How to recognize a fingerprint?	22
5.1	Image Enhancement	22
5.2	Thinning	24
5.3	Feature extraction	24
5.4	Problems of Minutiae	24
6	conclusion	25
7	Future works	26

Chapter 1

Abstract

This report based on the applicability of mathematical and geometrical functions for proposing the methods solving the problems of fingerprinting browsers based on their behaviour during a handshake. We propose the ideas which are based on simplification and coverage the most important information in images. The algorithms include the mathematical and geometrical sides including bicubics interpolation, trigonometric functions and etc for warp and weak images. Also the practical filters which are included for denoising. We receive the considerable results after test the algorithm by different ways including inverse mapping, angle of rotating, value of loss function.

Keywords: Fingerprints, Image processing, Bi-cubics, Geometric interpretation, Image classification

Chapter 2

Introduction

One of the most important characteristics in humans are their fingerprints. Due to its uniqueness, it has been used in various fields including biometric security, electronic devices, forensic science to identify suspects, victims and etc. While fingerprint analysis helps to make easy ways in such like fields, it has challenges. For instance, classification and identification systems are far from perfection, also the quality of each individual ridge in the particular prints being examined may leads to unreliable results that generally can not be repeatable. In this project we discuss about the images which are come from handshake during the fingerprinting and try to present and improve algorithms by using the mathematical, geometrical and tools of image processing.

2.1 Related Works

The literature in this field has led us to the following information, so we have a brief description of fingerprinting analysis in this part. Authentication is a key role in security systems and many applications are used to recognise the owner and access to information. During decades, methods are created such as the passwords, keys or cards and biometrics features while the third one works better. Biometric systems are included behavioural and physical. The side of physical like the fingerprint and the faceprint, have more secure in front of copy and build[1]. Regarding to the importance of fingerprints in different contexts, its quality is an important issue, therefore poor image quality can lead to inaccurate detection[2]. Indeed, algorithms are written for simulating the artefacts which create during the fingerprint scanning. Initially, the motion models are built by using the mathematical relationships that map pixel coordinates from one image to another[3]. There are many parametric motion models are received from 2D transferring to none planer[Video mosaics for virtual environments]. A range of 2D motion is included Translation, Rotation and Translation, Scaled rotation, Affine, Projective and Hierarchy of 2D Transformations[4]. One of the methods for motion model is called Direct which warps or shifts the images relative to each other and compares how much the pixels are matched [4]. On the opposite side there is feature

based method that corresponds to the features of images like edges, corners and etc[4]. For execution the best direct motion model, we should calculate the Error Metric[4]. In 1998, the method was created by using Zernike moments which could solve for the translational registration, rotational registration(2D and 3D) and zooming all at the same time[5]. For rotating the image we need the the angle, so there are two-stage framework for realising the rotation invariant template matching; first, histograms of orientation codes are employed for approximating the rotation angle of the object and second, matching is performed by rotating the object template by the estimated angle[6].

Another important part in the analysis digital fingerprinting is filters. As we know in the blurred fingerprinting images, finding the ridges is very difficult while valleys in such like images can be found more easy[7]. The extraction of valleys, gives more information for the processes of the matching and post-processing[7]. By using the filters including: 2D filters, Median filter, Gaussian filter, Wiener filter, Kalman filter and Gabor filter, the number of the valleys are extracted, are more than the ridges, So for blurred fingerprint images, the option of extracting valleys instead of ridges is better, because in some blurred fingerprints cannot extract even a single ridge[7]. However, this method takes more time comparing to extract rides, but it is not required here[7].

2.2 Overview of the implementation

2.2.1 Architecture of the code

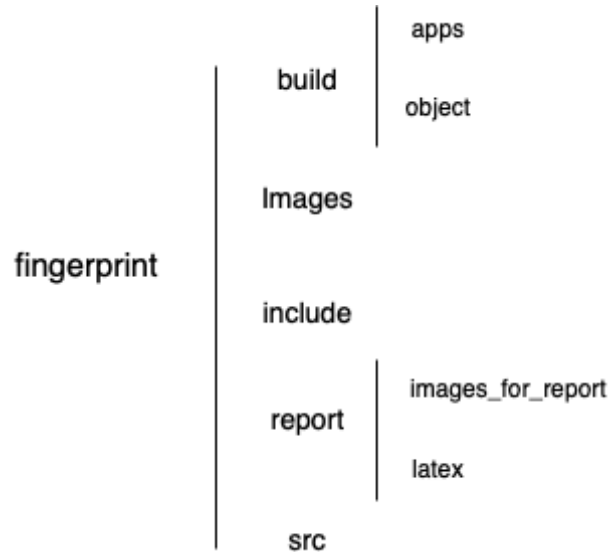


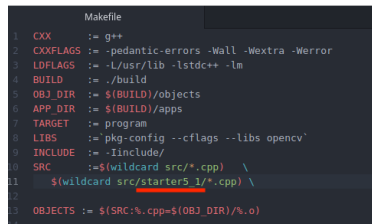
Figure 2.1: Architecture of the fingerprint file

- **build** This directory have two files here which include the apps and object each one provide the program and obejets of our file respectively.
- **images** The directory include the data which are received for this project.
- **include** The hpp format of each directory is received.
- **report** The report of project is available.
- **src** The cpp format of each directory is put. We also have tests in each subfile of src.

2.2.2 How to launch the code?

For launching, you should write the name of the folder of the desired test, to the place which is underlined(2.2). Then, you should go at the root of the project and type in the terminal this line of command : "make clean ; make ; ./build/apps/program".

The code is available in the attached website <https://gitlab.com/gabaid/fingerprint>.



```

Makefile
CXX      := g++
CXXFLAGS := -pedantic-errors -Wall -Wextra -Werror
LDFLAGS  := -L/usr/lib -lstdc++ -lm
BUILD    := ./build
OBJ_DIR  := $(BUILD)/objects
APP_DIR  := $(BUILD)/apps
TARGET   := program
LIBS     := `pkg-config --cflags --libs opencv`
INCLUDE  := -Iinclude/
SRC      := $(wildcard src/*.cpp) \
            $(wildcard src/starter5 1/*.cpp) \
OBJECTS  := $(SRC:%.cpp=$(OBJ_DIR)/%.o)

```

Figure 2.2: The view of Makefile

2.2.3 Tools

We have chosen the library Opencv. The class Mat lets to cast images and the function gives access to each pixel value. Each pixel value is encoded as a *< uchar >*, which is just an integer value in [0,255]. In this project, we do not use any class.

• **Mat()**

```
cv::Mat::Mat ( int rows,
               int cols,
               int type
             )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

rows Number of rows in a 2D array.

cols Number of columns in a 2D array.

type Array type. Use CV_8UC1, ..., CV_8UC4 to create 1-4 channel matrices, or CV_8UC(n), ..., CV_64FC(n) to create multi-channel (up to CV_CN_MAX channels) matrices.

Figure 2.3: Overview of the methods of Interpolation

• **at()**

```
template<typename _Tp>
_Tp& cv::Mat::at ( int row,
                  int col
                )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

row Index along the dimension 0

col Index along the dimension 1

Figure 2.4: Overview of the methods of Interpolation

Chapter 3

Methodology

3.1 Symmetry

This method is based on the symmetry of images and included the steps which use the mathematical functions. Initially, we apply for remap function to do symmetry.

The following matrices are respectively the symmetry along to x axis and the symmetry along to y axis :

$$M_1 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad M_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (3.1)$$

```
◆ remap()  
  
void cv::remap ( InputArray   src,  
                OutputArray  dst,  
                InputArray   map_x,  
                InputArray   map_y,  
                int           interpolation,  
                int           borderMode = BORDER_CONSTANT,  
                const Scalar & borderValue = Scalar()  
              )
```

Figure 3.1: Remap function

Results and Analysis

The images are put below about the original and the symmetry along the y axis.



(a)

(b)

Figure 3.2: (a)The original (b)The symmetry

We get that symmetry and rotation are not the same operation by calculating the determinant of them. The determinant of the symmetry is equal to -1 and determinant for rotation is equal to 1.

3.2 Coefficient for Pressure

We propose a coefficient function for weak pressure(3.2). It decreases when the pixel intensity of the original image as it goes towards zero(the distance is high), or keep it the same when it close to one(the distance is less). We introduce the barycenter as the center of spot and the euclidean distance between it and other pixel where the coefficient needs to be computed. They are basic for receiving the coefficient function. The properties of the function are:

$$c(r)_{r \rightarrow \infty} = 0 \qquad c(0) = 1 \qquad (3.2)$$



Figure 3.3: Weak finger

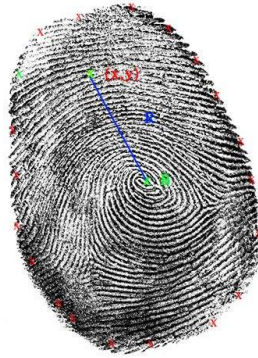


Figure 3.4: The distance between barycenter and the pixel

3.2.1 Results and Analysis

We receive the coefficient function(3.3) and according to the graph(3.5), it goes to zero when the distance is near to infinite.

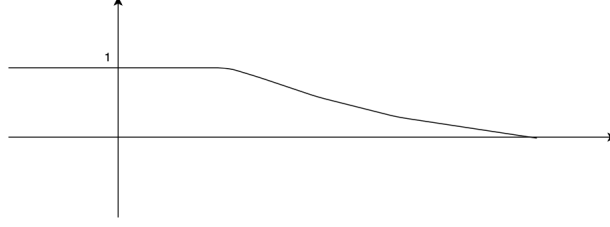


Figure 3.5: The distance between barycenter and the pixel

$$c(x, y) = (1 - \text{Tanh}((x^2 + y^2/\alpha)^p - R))/2 \quad (3.3)$$

3.3 Geometrical Warp

In this section, we propose a mathematical function(3.4) which rotates and translates the image. Therefore, we need the informations about the angle and the center of rotation. Following, it includes three parts to receive the necessary informations.

$$(x', y') = W(x, y; p) \quad (3.4)$$

3.3.1 Find the center of rotation(Centroid)

The idea is suggested about rotating clean image according to the barycenter which is cause to keep more information.

Threshold

Threshold can separate different areas of the image. We do binarization by the domain $[0,100]$ are black and the $[100,255]$ are white.

$$dst(x, y) = \begin{cases} 1 & \text{if } src(x, y) > threshold \\ 0 & \text{if } src(x, y) < threshold \end{cases} \quad (3.5)$$

Barycenter

We find the center of gravity of the image, not the rectangular center that surrounds it. We use the first-order central moment in the x and y directions.

$$C(c_x, c_y) \text{ so that } C_x = \frac{M_{10}}{M_{00}} \text{ and } C_y = \frac{M_{01}}{M_{00}} \quad (3.6)$$

3.3.2 Angle of rotation

After deriving the barycenter, we move to calculate the angle.

Meanfilter

In this part we get the kernel with size $(30 * 30)$ and apply the mean filter. This helps us to get the integrated image.

Threshold

We do threshold on the black and white image and convert it into a binary image.

$$dst(x, y) = \begin{cases} 1 & \text{if } src(x, y) > threshold \\ 0 & \text{if } src(x, y) < threshold \end{cases} \quad (3.7)$$

We suppose that the pixels in $[0, 170]$ are black and in $[170, 255]$ are white. In this part, we find the output image contours.

Sobel

For presenting the changes in intensity, we use Sobel operation that returns edges at those points where the gradient of the considered image is maximum. As a matter of fact, a high changes in gradient indicates a major changes in image.

Finding best ellipse

By using the fitEllipse function, we approximate the best ellipse which is contained all points of contours.

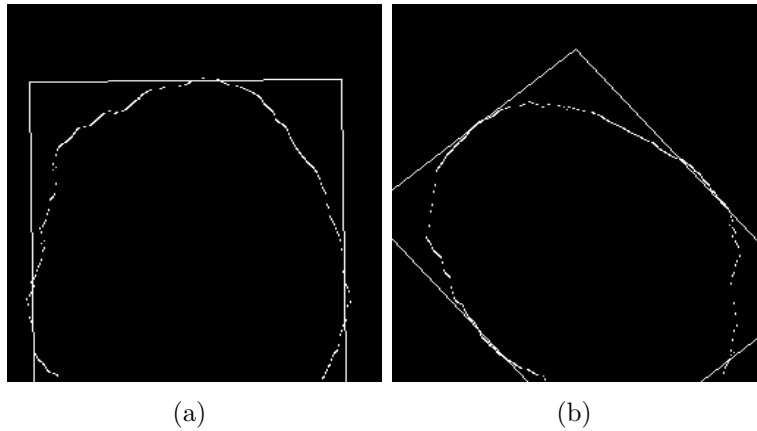
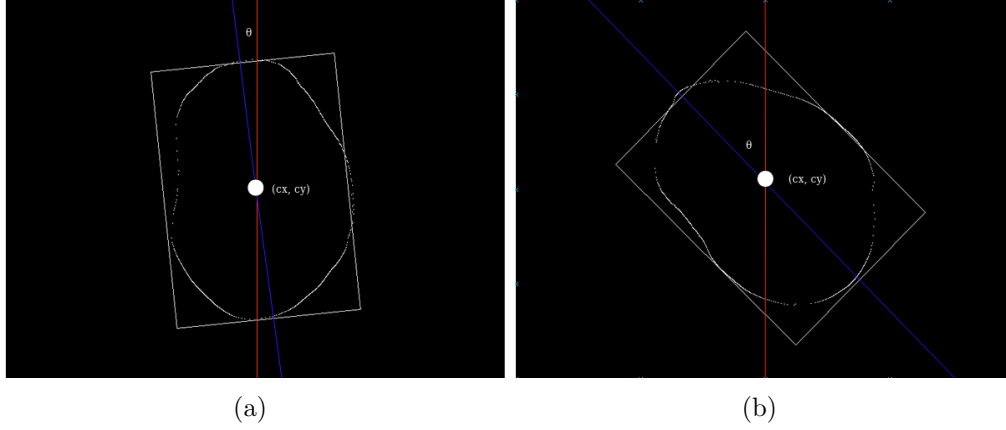


Figure 3.6: (a)The contour of clean finger (b)The contour of warp finger

Receiving the angle

Finally we can receive that the angle between the rectangles which are obtained.



3.3.3 Make the rotation

Following the parameters of center and angle then we get the function of warp, $W(\theta, c_x, c_y)$, and apply to rotate.

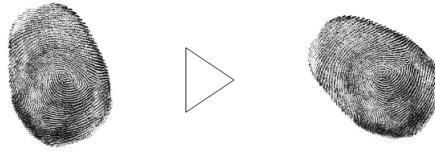


Figure 3.7: The rotation

Inverse mapping

For answering this question “Which pixel coordinate in the source image do we sample for each pixel in the destination image?” we inverse map (3.9). we have the matrix of rotation according to the center of the rotation (3.8).

$$\begin{pmatrix} \alpha & \beta & (1-\alpha)c_x - \beta c_y \\ -\beta & \alpha & \beta c_x + (1-\alpha)c_y \end{pmatrix} \begin{matrix} \alpha = \cos \theta \\ \beta = \sin \theta \end{matrix} \quad (3.8)$$

$$(x, y) = W^{-1}(i, j) \quad (3.9)$$

Interpolation

After the inversing, we have the value of the pixel which is float while we want to get the integer value so we need the interpolation to get it, There are three methods including:

Nearest Neighborhood, Bi-linear and Bicubic. We use the bicubic interpolation by 16 pixels according to the quadratic error which is calculated for each one of the interpolations.

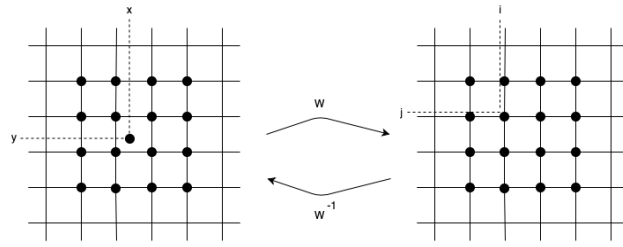


Figure 3.8: The location of pixel

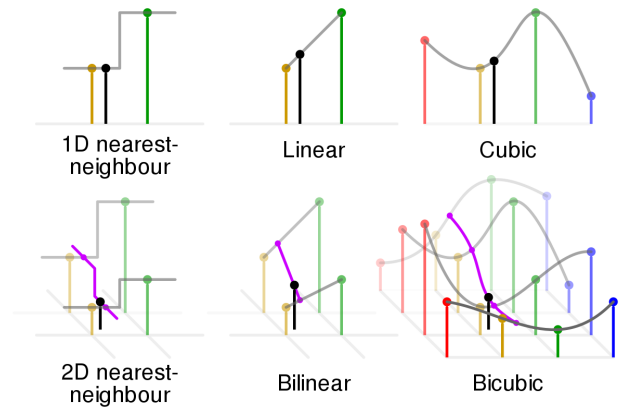


Figure 3.9: Overview of the methods of Interpolation



(a)



(b)



(c)



(d)

Figure 3.10: (a)The original (b)Two times rotation with Nearest Neighbour (c)Two times rotation with Linear (d)Two times rotation with Bicubic

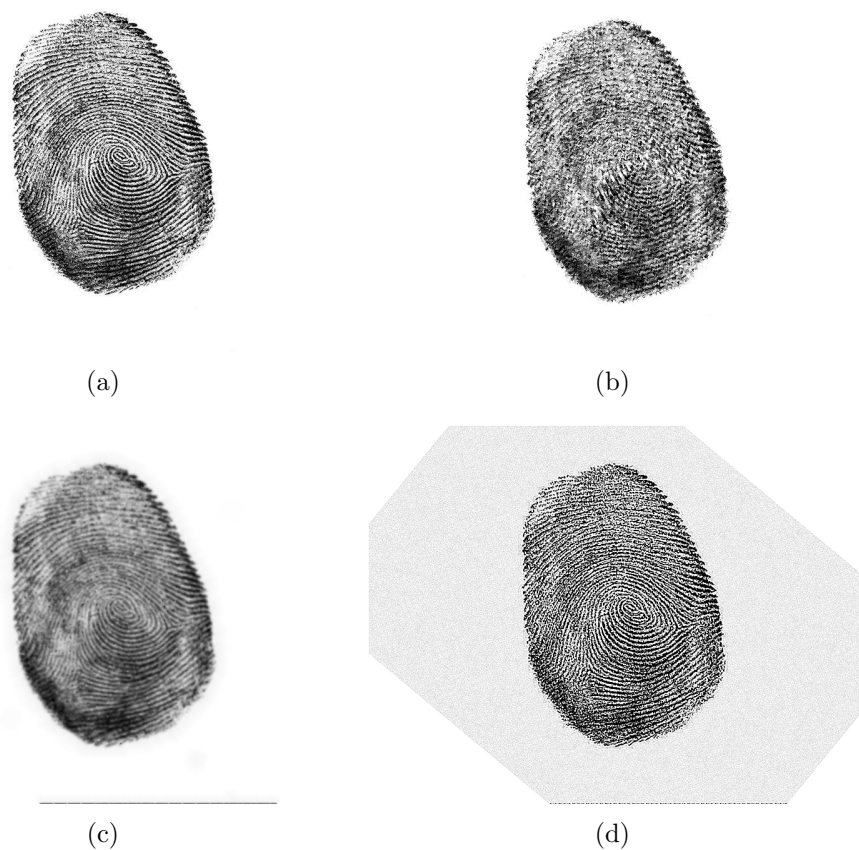


Figure 3.11: (a)The original (b)Twenty times rotation with Nearest Neighbour
(c)Twenty times rotation with Linear (d)Twenty times rotation with Bicubic

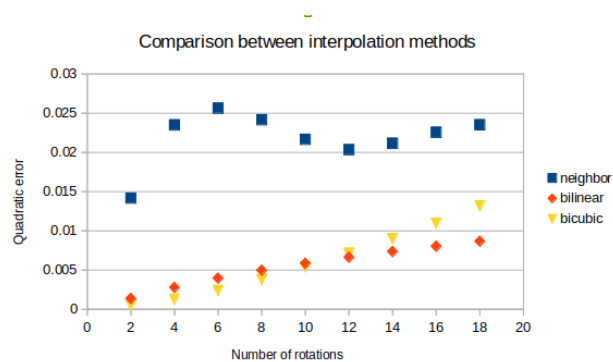


Figure 3.12: Comparison between interpolation methods

3.4 Linear Filtering

In this part we explain which linear filter denoise the blurred images satisfactory. The idea is that by making a noise on the clean image, we get the blurred image, then reverse the relation and obtain the kernel that cleans the image and eliminate the noises. Following, we explain the mechanism of linear filtering.

3.4.1 Direct convolution

For applying a linear filter, we need to have a kernel, then we can do convolve (3.10)(3.13) by kernel on the image. The kernel is also an array of fixed coefficients and has an anchor, which is usually in the center of the array. The anchor is the element of the kernel that is examined on the pixel. We try to keep the size of output image by padding (3.11)(3.14).

$$(f * g)[m, n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(m - i, n - j)g(i, j) \quad (3.10)$$

$$p = \frac{(k - 1)}{2} \quad (3.11)$$

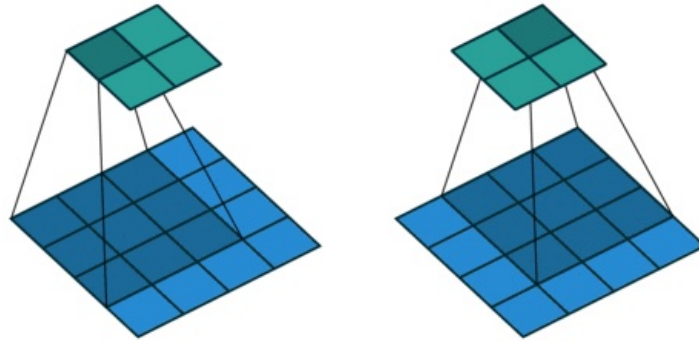


Figure 3.13: Direct convolution

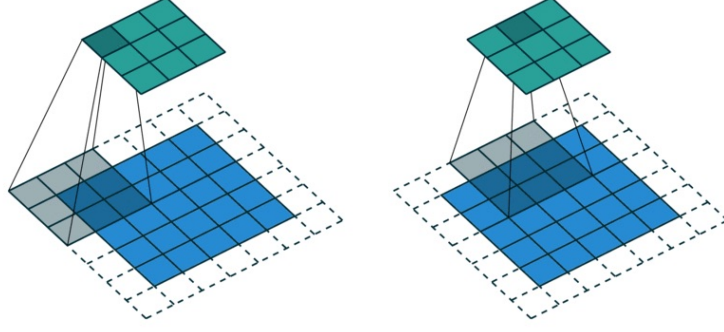


Figure 3.14: Padding

Complexity of algorithm

The complexity of the algorithm is defined how many operations which are used to run and complete the algorithm. Therefore for direct convolution, we can calculate the function (3.12) which (M,N) and (m,n) are the number of rows and columns of the original matrix of image and the matrix of kernel respectively.

$$O((MN)mn) \quad (3.12)$$

3.4.2 Fast Fourier Transform(FFT)

The Fourier transform converts the image into its sinusoidal and cosine components. In other words, it moves the image from a location range to a time or frequency domain. Therefore, if the pixels (elements of image) are in location range, we do direct convolution but on the contrary in the frequency domain, we do convolution by Fast Fourier transform. For convolution in this part, first we apply for the Fourier transform (3.13) and after the inversing and receive the convolution.

$$F(f * g) = F(f).F(g) \quad (3.13)$$

Complexity of algorithm

The complexity of the Fast Fourier transform of the original image $(M * N)$ and kernel $(m * n)$, is calculated by the function below

$$O((M(N\log N)) + O(m(n\log n)) + O(MN) + O(1) + O(M(N\log N)) \quad (3.14)$$

3.4.3 Results and Analysis

The results show the less complexity when we apply for convolution of FFT and it is more fast than direct convolution in the meanwhile, if we have the small kernel, direct convolution is more efficient.

$$\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad (3.15)$$



Figure 3.15: (a)Convolution (b)Convolution FFT (c)Difference

	Convolution	Convolution FFT
Complexity	$O((MN)mn)$	$O((M(N\log N)) + O(m(n\log n)) + O(MN) + O(1) + O(M(N\log N)))$
Efficient	The small kernel	The size of original image and kernel are close

Table 3.1: Comparson between Direct and FFT convolution

Chapter 4

Optimization

In optimisation part, we use the mathematical function which is known as the loss function(4.1) . This formula includes two images, original and translation one. The goal is that receive the optimal parameter. The way is minimising the loss function.

$$l(p) = \sum_x \sum_y (f(x, y) - g(w(x, y; p)))^2 \quad (4.1)$$

4.1 Optimization for translation along x axis

So, we apply for translation the image along the x axis and after compute the optimal value from the loss function.



Figure 4.1: Original image



Figure 4.2: Rotated image

Results and Analysis

In this case, we implement for greedy strategy and get the number 35 for the optimal value for p_x .

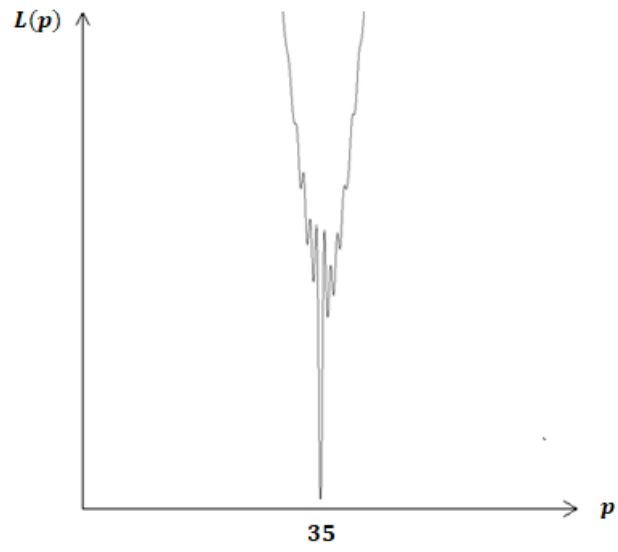


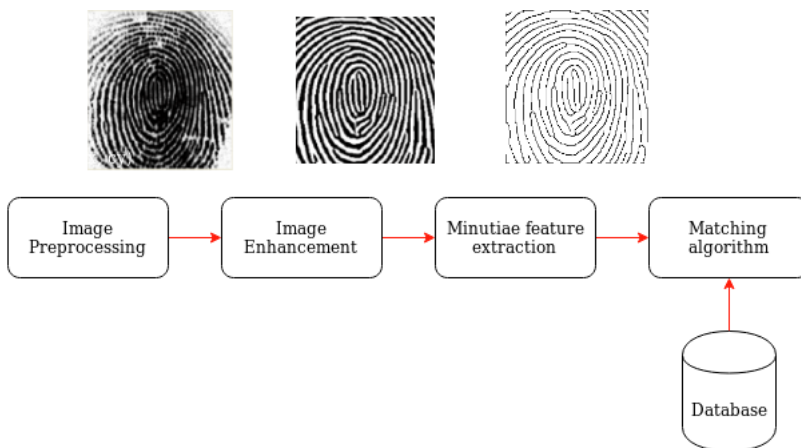
Figure 4.3: The loss function

Chapter 5

How to recognize a fingerprint?

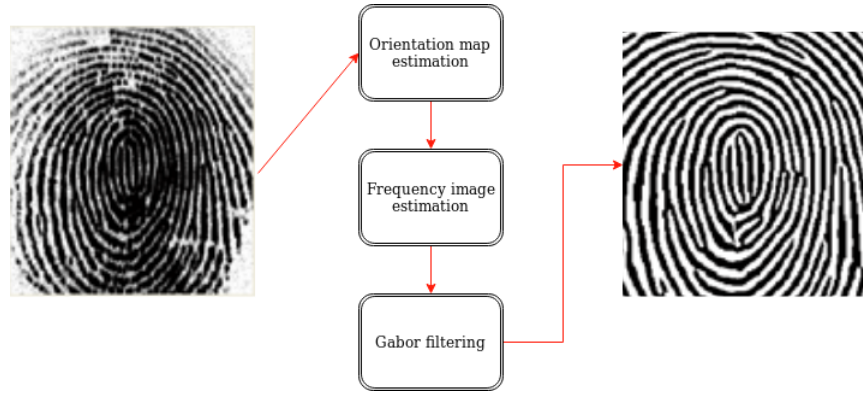
What is the aim of this project? this is the question that we ask ourselves and our answer is that achieve the best algorithm that would capture all the quality states of a fingerprint image and ensure the highest reliability. So, after numerous investigations, we came to the important feature that all fingerprint algorithms are based on, It is "Minutiae". Following, we explain the mechanism of it and give the results.

Minutiae points are the local ridge discontinuities, including two types, ridge endings and bifurcations. The image with around 40 to 100 minutiae, has good quality [9]. Since accurate matching of fingerprints depends largely on ridge structures, the quality of the fingerprint image is of critical importance [8]. In fact that fingerprint image may not always be well defined due to elements of noise that corrupt the clarity of the ridge structures [8].



5.1 Image Enhancement

In this part, we apply for three steps which are Orientation map estimation, Frequency image estimation and Gabor filtering.



Orientation map and frequency estimation

First we do the orientation map estimation to get the local ridge orientation. The orientation map is the direction of the underlying ridge at the minutia location and ridge orientation map represents the local direction of the ridge-valley structure. After we estimate frequency by using the ridge frequency map as the reciprocal of the ridge distance in the direction perpendicular to local ridge orientation.

Gabor filtering

Following, we use Gabor filter by calculator the orientation of each block. So, we can receive the good frequency and an enhanced image.

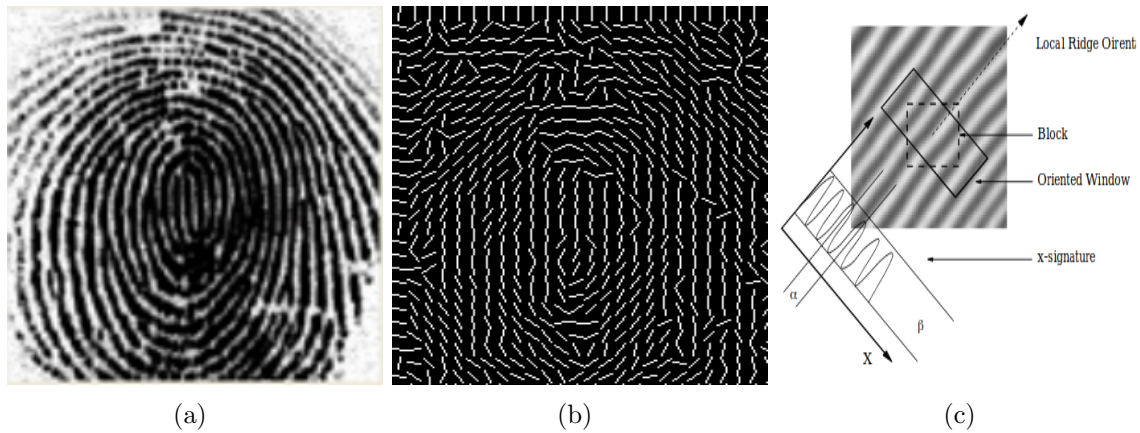
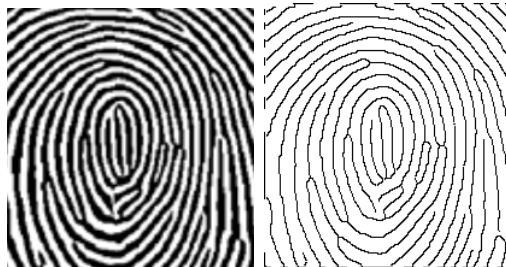


Figure 5.1: (a)The fingerprint image (b)Orientation map (c)The explanation of receiving the frequency

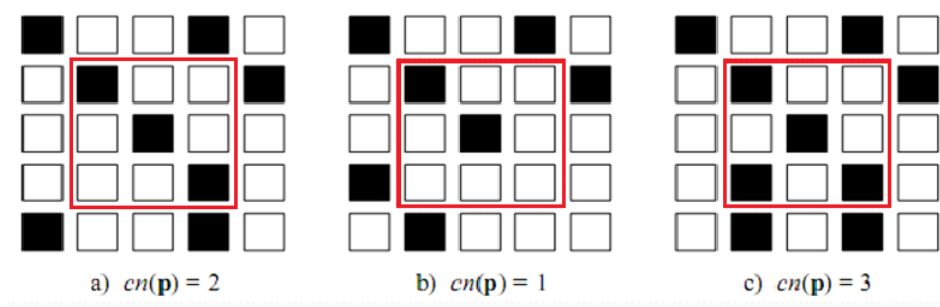
5.2 Thinning

Most fingerprint minutia extraction methods are thinning-based where the skeletonization process converts each ridge to one pixel wide.



5.3 Feature extraction

Minutia points are detected by locating the end points and bifurcation points on the thinned ridge skeleton based on the number of neighboring pixels. The end points are selected if they have a single neighbor and the bifurcation points are selected if they have more than two neighbors.



5.4 Problems of Minutiae

The main problem in the minutiae extraction method using thinning processes comes from the fact that minutiae in the skeleton image do not always correspond to true minutiae in the fingerprint image [8].

Chapter 6

conclusion

In conclusion, we received two functions which give us the weak and warp images. We proposed new idea for the center of rotation and the center of image for coefficient function which are caused to get better results. Besides, we compare two kinds of filter and their efficiency. For optimization, the optimal value is received by using the greedy strategy. Finally, we investigate on extraction the Minutiae of fingerprint and write the an algorithm which is based on.

Chapter 7

Future works

In future, we will focus on the algorithms are based on minutiae features and try to find the best filters that may achieved by combining them.

Bibliography

- [1] abassi, Elham, Charles Wilson, and Craig I. Watson. Fingerprint Image Quality. No. NIST Interagency/Internal Report (NISTIR)-7151. 2004.
- [2] heofanos, Mary, et al. "Does the angle of a fingerprint scanner affect user performance?." Proceedings of the Human Factors and Ergonomics Society Annual Meeting. Vol. 52. No. 24. Sage CA: Los Angeles, CA: SAGE Publications, 2008.
- [3] zeliski, Richard. "Image alignment and stitching: A tutorial." Foundations and Trends® in Computer Graphics and Vision 2.1 (2007): 1-104.
- [4] mage Alignment and Stitching: A Tutorial Richard Szeliski.
- [5] otation and Zooming in Image Mosaieing.
- [6] llah, Farhan, and Shun'ichi Kaneko. "Using orientation codes for rotation-invariant template matching." Pattern recognition 37.2 (2004): 201-209.
- [7] inothkanna, R., and Dr Amitabh Wahi. "A novel approach for extracting fingerprint features from blurred images." International Journal of Soft Computing and Engineering (IJSCE) 2.2 (2012): 231-235.
- [8] ansal, Roli, Priti Sehgal, and Punam Bedi. "Minutiae extraction from fingerprint images-a review." arXiv preprint arXiv:1201.1422 (2011).
- [9] . Hong, Y. Wan, and A. K. Jain, "Fingerprint image enhancement: Algorithms and performance evaluation", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20(8), 1988, pp. 777-789.