Fingerprints Recognition

Final Project Report of ECES 681

- Fundamentals of Computer Vision

Shangqi Wu

2016 March

Fingerprints Recognition

Final Project Report

1. Introduction

Fingerprint recognition is a widely adopted fundamental real-world application of computer vision discipline. This technology is well-developed nowadays, and it extensively provides reliable safety measurements in different scenarios, even some cell phones are equipped with fingerprint reader. It is possible to use fingerprints as personal identification, and also use fingerprints as evidence for case investigation and forensics.

As described in final project instruction, the project is proposed under the context of investigation of burglar. With fingerprint database and fingerprint from crime spot collected, it is feasible to infer who is the suspect. Investigators should transform distorted fingerprint image back to an orthodox view, and then extract fingerprint characteristics, and compare the information with those fingerprints in the database. The fingerprints with similarity beyond a certain threshold should be reported and recorded, and persons that the fingerprints belong to should be investigated carefully.

Fingerprints can be analyzed and discriminated by characteristics called minutia points. Minutia points include ridge ending, that is the abrupt end of a ridge, ridge bifurcation, which is a single ridge that divides into two ridges, short ridge or independent ridge, i.e., a ridge that commences and travels a short distance and then ends, island that a single small ridge that bifurcates and reunites shortly afterward to continue as a single ridge, spur, which means a bifurcation with a short ridge branching off a longer ridge, crossover or bridge, which looks like a short ridge that runs between two parallel ridges, delta that is a Y-shaped ridge meeting and core which seems like a U-turn in the ridge pattern.

Since minutia points are highly different from person to person, and there is not likely to exist over two persons that have all same minutia points on the same finger, minutia points are the main evidence of identification of a person.

There are also number of ways to utilize all minutia points information. The most obvious approach is to read a fingerprints image, cut all irrelevant backgrounds and resize it to the same size as images in the database, then calculate coordinate differences among corresponding points. A better approach can preserve relative positions among minutia points, i.e., remain specific shapes consisted by minutia points like convex hull, smashed triangles, etc. Even more sophisticated approaches may also take advantage of local tangent vectors, local gradient, or even texture characteristics. The following image shows an example of how could this general approach work.

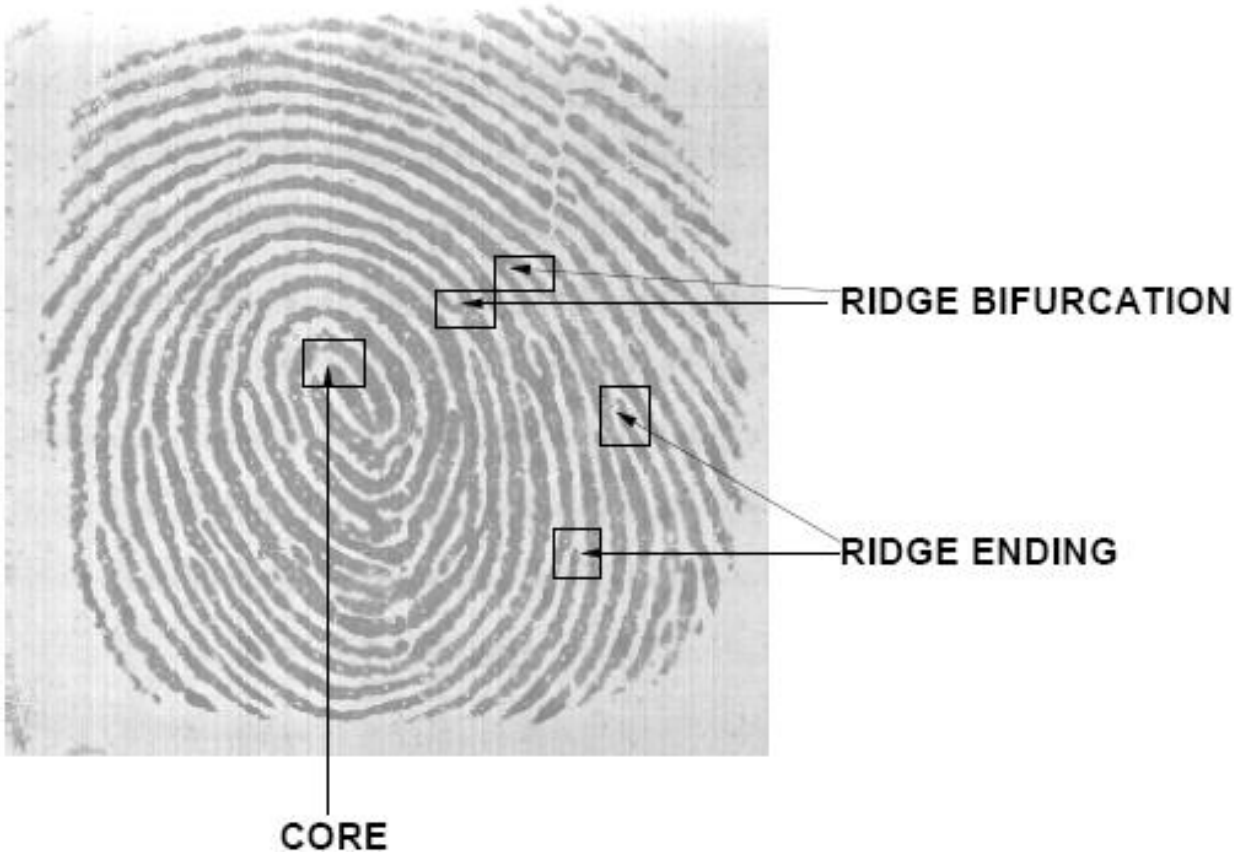

Figure 1.1 Example of minutia points

With so many biometric features that can be extracted for analysis, fingerprint recognition should be an ideal approach to identification security and forensics

investigation. As proven in the following sections of this report, fingerprint recognition works exceptionally well as long as fingerprint images preserves reasonable resolution.

2. Outline of Recognition Process

Since all fingerprint images are provided with project instruction, there is no need to concern about fingerprint acquisition from scanners. With help of Matlab built-in function, it is possible to read in the image file and convert it to gray scale bitmap, so the algorithm will be working with two 8-bit unsigned integer matrices.

The first step should be image pre-processing. This stage should include inverse affine transform of distorted fingerprint, enhancement of fingerprint ridges, and black-white image value conversion. This step mainly involves Fourier transform, histogram equalization, automatic grayscale threshold estimation.

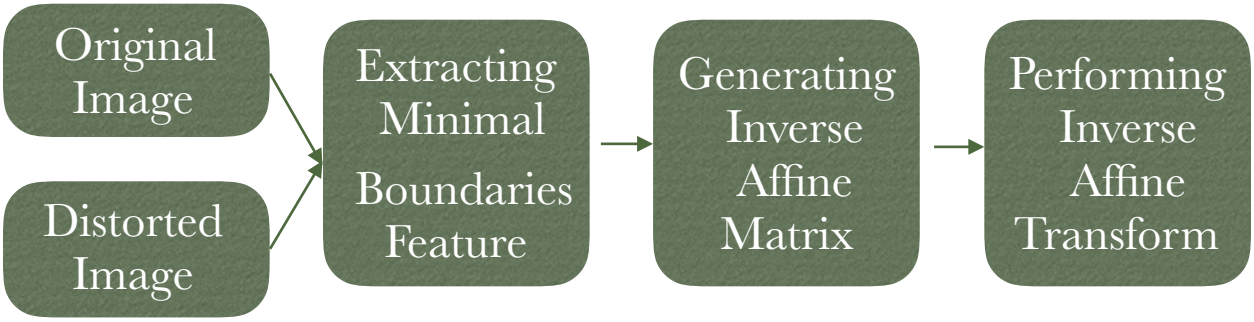

Figure 2.1 Inverse affine transform of distorted image

Figure 2.2 Image pre-processing stage

Next step is supposed to be critical processing stage, which extracts minutia points. At first, the previously processed binary image is retrieved, and then ridges of fingerprints are shrank to one-pixel width while preserving its length and curvatures. After filtering some subtle spikes and breaks next to ridges, the algorithm matches some patterns of certain kinds of minutia points and store points information if there are any match one of those patterns.

Figure 2.3 Critical processing stage

The final step of image processing is review information of all minutia points and remove points which are suspected as false minutia.

Figure 2.4 Post-processing stage

As the last step of the whole process, the information of distorted and original images are compared and a similarity score is computed.

Figure 2.5 Fingerprint matching stage

The discussion and figures above could show how this algorithm works sufficiently. Different stages will be performed in order. Affine transform of distorted fingerprint is performed before the process. Then the pre-process, critical process and post-process are executed consequently. At the last, fingerprint matching algorithm is executed.

A input unknown fingerprint will be compared with every sample existing in the database, the ones generating similarity scores higher than a preset threshold will be reported as suspects.

3. Image Pre-processing

Before this stage starts, an inverse affine transformation is supposed to be applied to unknown fingerprint. Then, the whole stage is executed to all images in database and inverse-transformed input images.

3.1. Inverse Affine Transformation

Since all input images are distorted by some unknown affine transformations, the first step is to calculate inverse affine transform matrix and transform the distorted image back to an orthodox view. This step can be achieved by convex hull of fingerprints aligning, or curve / edge moments processing, however due to the special shapes of fingerprint images, neither of approaches work perfectly.

A combined solution is developed for this step, that is to find a minimal boundary polygon and generate affine transform matrix by moments of polygon edges. This approach worked perfectly for No.1 image in database with No.1 known image provided. The following 2 figures demonstrate outcome of my inverse affine transform algorithm.

Figure 3.1.1 Distorted image

Figure 3.1.2 Inverse transformed image

As we can observe from the recovered image, it provides an orthodox view of the image, which is similar to all standard images in the database. As a matter of fact, it is transformed from No.1 image in database, which can be concluded by visual examination that they preserve the same characteristics.

3.2. Fingerprint Ridge Enhancement

During the experimental runs of the algorithm, it is observed that some subtle noise, which severely affects recognition accuracy later, exists in large background area of distorted fingerprints. As a result, a sub-algorithm is developed which preserve a minimal rectangle contains orthodox fingerprint and cut off area outside the rectangle. Both original images from database and input unknown fingerprints are performed this process for better recognition.

After the image truncation described above, the next task is to enhance resolution of fingerprint image. As a commonly used technique, Fourier transformation (FFT) can be applied to the image, and certain coefficients can be amplified to enhance fingerprint textures. The basic principle to enhance textures is to amplify the dominant FFT coefficient by a certain factor.

Since not all area of fingerprint image contains valid ridge, the FFT coefficients varies in a great range. As a result, this phenomenon may cause suboptimal outcome. To achieve a better enhancement, image is divided into blocks to achieve local optimal coefficient amplification.

The following formula describes the approach to amplify the magnitude of FFT dominant coefficients.

$$g(x,y)=F^{-1}\{F(u,v)\times|F(u,v)|^{K}\}$$

The factor K can be specified by users, the larger K is, the larger white area, which is supposed to represent fingerprint ridges, is expanded. During the experiments with different factors, it is indicated that a factor ranging from 0.2 to 0.4 may generate a reasonable result. The ridges will be thicken and strengthen, meanwhile without merging two distinct ridges.

3.3. Histogram Equalization & Image Binarization

Before converting enhanced fingerprints into black-white binary images, an optimization in image contrast is recommended. It can be accomplished by histogram equalization. The reason of applying histogram equalization is to optimize image quality, by making different luminance level distributed smoothly, so that it will be easier for next step to choose a proper threshold for value binarization.

To convert input gray scale image, the key point is to find an appropriate threshold. The values below this threshold will be marked 0 while values after this threshold will be marked 1 in output binary image.

Similarly, this section is also performed by dividing image into blocks to find local optimal gray scale threshold. The program adopts a number that a little less than local average gray scale value as threshold. The following figures contrast the difference with and without optimizing algorithms, i.e., enhancement and adaptive image binarization.

Figure 3.3.1 Image with optimization

Figure 3.3.2 Image without optimization

Two images above are binary images with and without pre-processing of image No.1 in the database, respectively. The image on the left, which is optimized, shows thicker and clearer image than right image with only Matlab built-in black-white

image conversion function. It can be concluded that this pre-processing algorithm enhances fingerprint ridges effectively.

4. Image Characteristics Extraction

After image pre-processing, the input image is clear enough to examine and extract characters, i.e., minutia points for further steps. This section is focused on minutia points extraction.

4.1. Critical Content Block Map

As for part of fingerprint character, direction of fingerprint ridge texture can play a role in this process. By calculating local gradient of pixel distribution in current block, the program can determine if current block contains trivial contents that may affect final analysis negatively.

The algorithm is to calculate gradients of horizontal and vertical direction, then an approximate local direction can be calculated. After estimating local direction, a general gradient value can be estimated by the following formula. If the value is below a threshold, this block will be marked as trivial, and it will be discarded when extracting minutia points.

$$E=\frac{2\sum_{0}^{x}\sum_{0}^{y}\left(g_x\times g_y\right)+\sum_{0}^{x}\sum_{0}^{y}\left(g_x^2-g_y^2\right)}{W\times W\times\sum_{0}^{x}\sum_{0}^{y}\left(g_x^2+g_y^2\right)}$$

By adapting the function above, blocks with output value less than 0.05 will be marked as trivial in the map and will not being considered for future process. Furthermore, the blocks on the edge of trivial area will also be assigned less weight during consideration of minutia points extraction.

This process allows extraction algorithm focuses on area containing critical ridge information, which is region of interest (ROI) of this project. Blocks within ROI area will be examined with more effort.

The following 2 figures demonstrates the results of local direction and ROI marking result, with the No.1 fingerprint image in the standard image database.

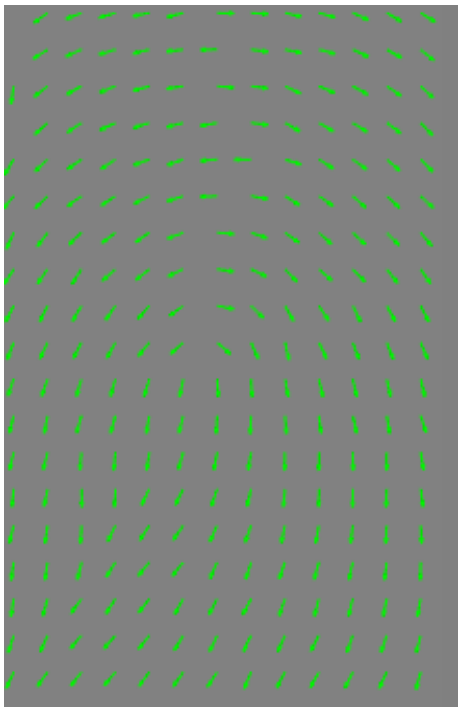

Figure 4.1.1 Direction map

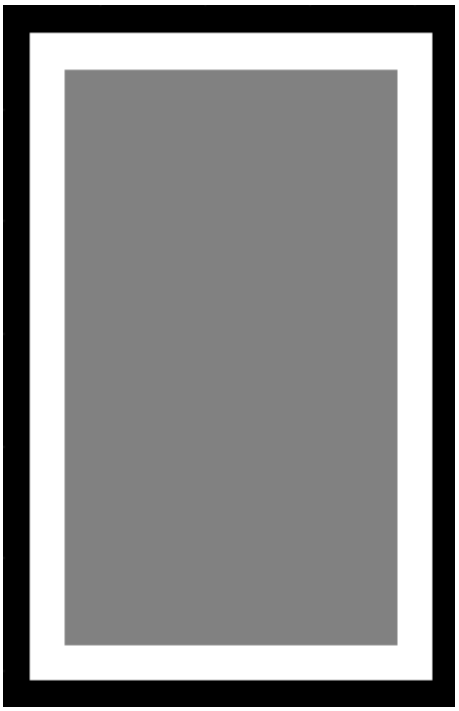

Figure 4.1.2 ROI map

Comparing with fingerprint character, which can be visually detected from figure 3.3.1, direction map matches image to a very good extent. The black area of ROI map stands for trivial area, while white stands for edge and gray is focused area. The output patterns match fingerprint well.

4.2. Ridge Thinning and Spikes Removal

Minutia extraction algorithm works under the assumption that fingerprint ridges are lines or curves, that is, all ridges are of width of one pixel. This stage can be achieved in Morphological approaches, by Matlab built-in functions. The algorithm mark edges of white edges as black iteratively, until the ridge is of width of one pixel.

The thinning algorithm may cause unexpected subtle spikes, H-breaks and islands as noises may interfere analysis in next steps, so they should also be filtered. The filtering algorithm may also be done in a similar way.

The following figures illustrates the difference that thin ridges with and without filtering process.

Figure 4.2.1 Ridges without spike removal

Figure 4.2.2 Ridges with spike removal

From the figures above, right figure shows a smoother fingerprint ridges, spikes or islands are refrained.

4.3. Minutia Extraction

After finished all process described above, minutia extraction can be performed. This stage can be performed by pattern matching with a fixed-size sliding window, the algorithm to recognize is straightforward.

Some of the patterns can be recognized as the following description. The recognition are done by 3 * 3 size window. If the center pixel is part of a ridge and has exactly 3 white neighbor pixels, then this pixel is classified as a ridge branch. If a pixel is white and it has only one white neighbor, it will be regarded as an ending point.

Following 2 figures show examples of how 2 different kinds of minutia points will look like and can be identified within the fixed-size sliding window.

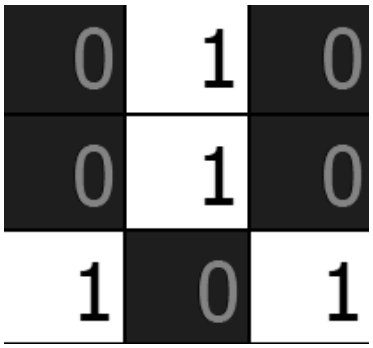

Figure 4.3.1 Pattern of bifurcation

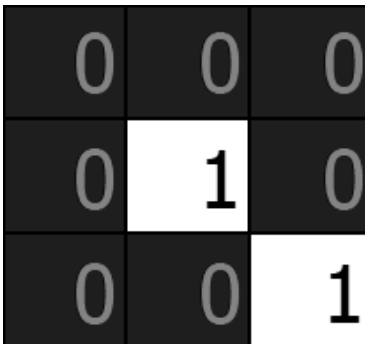

Figure 4.3.2 Pattern of endings

Along with minutia points information, all thin ridges are assigned with an ID for convenience of future steps.

During this stage, estimation of ridge width can be generated. This data can be helpful to false minutia removal. Since distances between ridges of a fingerprints are typically to be constant, so if there are minutia points or ridges which do not follow this distance, one of the points can be removed. To get this value, the algorithm can simply sum up all pixel values in the same row and divide by its length. To pursue better accuracy, column information or even local optimal can be taken into consideration.

5. Information Post-processing

This section mainly focuses on explain why and how to detect and remove false minutia point information.

Since there are several steps during previous 2 image processing stages, each step tries to eliminate noise very hard, however those steps themselves also introduces unexpected noise. Considering an instance as following description, it is possible that after ridge thinning, there is part of one ridge contains to much spikes, and the spike removal stage filters too much part of spikes and introduces artifacts of ridges, thus false ending points are reported.

The algorithm to remove false points are described as following. First, if distance between a branch point and a ending point is too close, i.e., less than inter-ridge width, both of them should be removed. Second, if 2 branch points are closer than ridge width, neither of them should be left on the image. Third, if 2 ending points

are that closer than inter-ridge width, and their directions are nearly the same, both points are to be removed. Last, if a ridge is too short, its ending points should be removed.

After removal, true minutia points will be retained. Following figures are comparison before and after false minutia removal.

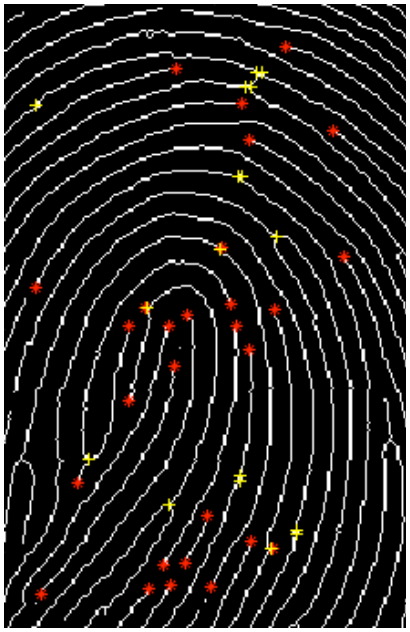

Figure 5.1 All minutia points

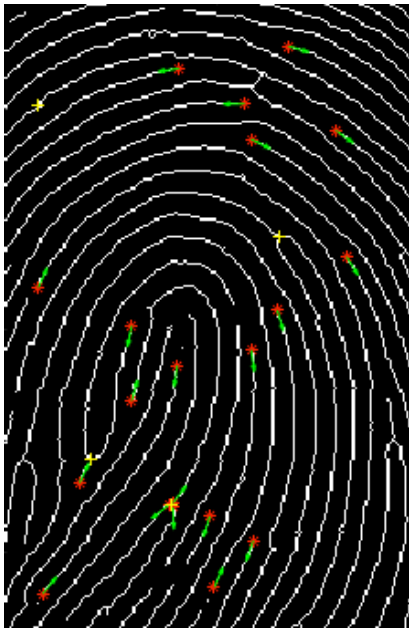

Figure 5.2 Reduced minutia points

6. Fingerprint Matching

The matching stage mainly takes 2 steps, first step is to align minutia points, while the second step is to match and calculate a similarity score of 2 fingerprints.

6.1. Points Alignment

A series of point coordinates represent a ridge containing minutia points. Ridges with normal length will be sampled by 10 points. So that the similarity of 2 ridges can be calculated by formula below with sample points coordinates.

$$S = \frac{\sum_{i=0}^{m} x_i u_i}{\left[\sum_{i=0}^{m} x_i^2 u_i^2 \right]^{0.5}}$$

Then both fingerprints will be tried to estimate a most fit translation and rotation. Since the transformation can be performed by following formula, it is also feasible to calculate the transform matrix.

$$\begin{bmatrix} x_{transform} \\ y_{transform} \\ \theta_{transform} \end{bmatrix} = matrix_{transform} \begin{bmatrix} x_{previous} - x_{reference} \\ y_{previous} - y_{reference} \\ \theta_{previous} - \theta_{reference} \end{bmatrix}$$

So that the transform matrix can be inferred as following. Thus it is possible to perform transformation to minutia points.

$$matrix_{transform} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The procedure is to choose a reference minutia point first, then the coordinate system of all other minutia points will be transformed similar to applying affine transformation. This step uses an angle output generated by trying to keep track short ridges that starting from a minutia point. Then each minutia point can be transformed according to its reference minutia point, and tries to find a best match in the new coordinate system.

6.2. Points Match & Evaluation

The matching algorithm should be automatic, adaptive, fault tolerant and cannot be simply comparing point coordinates, since it is impossible to have exactly the same fingerprint data, even for same image with different pre-processing parameters, the result will not be the same.

The goal can be achieved by adding error tolerance boundary for each minutia point. If the 2 aligned points have a x and y coordinates difference less than the tolerance boundary, they can be regarded as matched points.

Number of matched minutia points will be recorded, and the percentage of matched points out of total number of points will be regarded as the similarity score of 2 fingerprints. In this program, the total number of points is the value of total minutia points found in the first fingerprint. If there are reasonable

percentage of matched minutia points, we can deem 2 fingerprints match to each other. It is also possible to train an appropriate threshold to report hosts of fingerprints as suspects once any test results exceed the threshold.

7. Experiment & Verification

Due to approaching deadline and intensive image processing efforts, this section concentrates on comparison between No.1 unknown fingerprint image, i.e., FP1.png, and each image existing in database.

By visual examination, it is obvious that fingerprint in FP1.png is derived from 01.png, that is, they are fingerprints of the same person.

Figure 7.1 Image 01.png

Figure 7.2 Image FP1.png

After running the program with No.1 unknown fingerprint image with all 10 fingerprint images in the database, all percentage scores are recorded in the following table.

As we can see from the table below, comparing with our previous visual examination result, 01.png and FP1.png successfully matched with each other, with a very high percentage of over 88%, which definitely outreaches common threshold of around 70%.

Image	01.png	02.png	03.png	04.png	05.png	06.png	07.png	08.png	09.png	10.png
FP1 .png	88.2353	30.4348	46.6667	28	31.5789	33.3333	15.3846	30	29.4118	38.8889

Table 7.1 Recognition matching percent with fingerprint database

8. Conclusion

As we can learn from verification result, this program successfully classifies fingerprints come from the same person. The result matches what it is supposed to be.

As for future works, more diverse and reliable minutia points are to be discovered and utilized. This program takes advantage of direction of local blocks, and this part of information can also be adopted to fingerprint recognition. Texture characters are also mentioned to be possible features for recognition.

There is one feature that is not optimal that, the algorithm to perform inverse affine transform is not robust enough. Moments and convex hull are approaches to find transform matrix, and my combined approach has the best performance. However, it may not properly transform distorted images back. This scene happens rare and I cannot state a way to make it always happen.

In general, this approach to recognize fingerprint is good enough, since it accurately reports correct results.