**Chapter 1: Introduction**

**1.1. What is A Fingerprint?**

A fingerprint is the feature pattern of one finger (Figure 1.1). It is believed with strong evidences that each fingerprint is unique. Each person has his own fingerprints with the permanent uniqueness. So fingerprints have being used for identification and forensic investigation for a long time.



Figure 1.1 A fingerprint image acquired by an Optical Sensor

A fingerprint is composed of many ridges and furrows. These ridges and furrows present good similarities in each small local window, like parallelism and average width.

However, shown by intensive research on fingerprint recognition, fingerprints are not distinguished by their ridges and furrows, but by Minutia, which are some abnormal points on the ridges (Figure 1.1.2). Among the variety of minutia types reported in literatures, two are mostly significant and in heavy usage: one is called termination, which is the immediate ending of a ridge; the other is called bifurcation, which is the point on the ridge from which two branches derive.

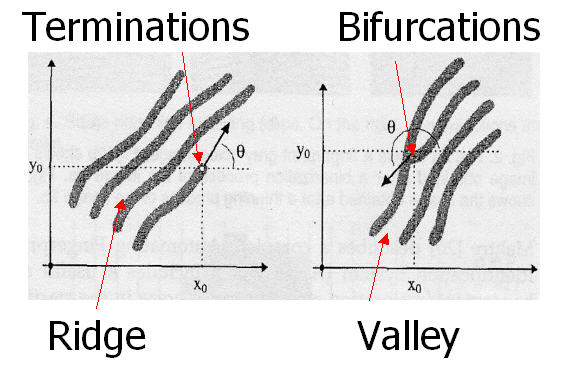


Figure 1.2 Minutiae. (Valley is also referred as Furrow, Termination is also called Ending, and Bifurcation is also called Branch)

**1.2. What is Fingerprint Recognition?**

The fingerprint recognition problem can be grouped into two sub-domains: one is fingerprint verification and the other is fingerprint identification (Figure 1.3). The fingerprint recognition here is referred as AFRS (Automatic Fingerprint Recognition System), which is program-based.

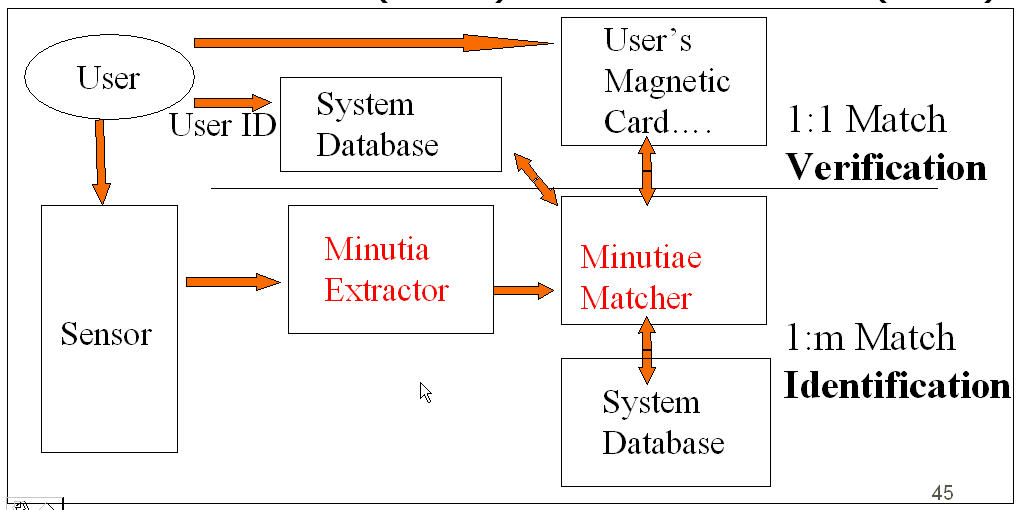


Figure 1.3 Verification vs. Identification

Fingerprint verification is to verify the authenticity of one person by his fingerprint. The user provides his fingerprint together with his identity information like his ID number. The fingerprint verification system retrieves the fingerprint template according to the ID number and matches the template with the real-time acquired fingerprint from the user. Usually it is the underlying design principle of AFAS (Automatic Fingerprint Authentication System).

Fingerprint identification is to specify one person’s identity by his fingerprint(s). Without knowledge of the person’s identity, the fingerprint identification system tries to match his fingerprint(s) with those in the whole fingerprint database. It is especially useful for criminal investigation cases. And it is the design principle of AFIS (Automatic Fingerprint Identification System).

However, all fingerprint recognition problems, either verification or identification, are ultimately based on a well-defined representation of a fingerprint. As long as the representation of fingerprints remains unique and simple, the fingerprint matching, either for the 1-to-1 verification case or 1-to-m identification case, is straightforward and easy.

**Chapter 2: Literature Surveyed**

**2.1. Designs**

**2.1.1 System Level Design**

A fingerprint recognition system constitutes of fingerprint acquiring device, minutia extractor and minutia matcher [Figure 2.1].

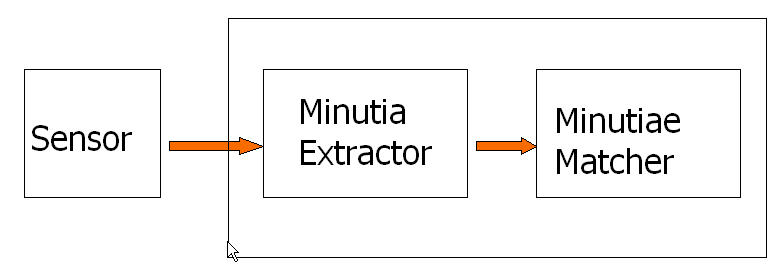


Figure 2.1 Simplified Fingerprint Recognition System

For fingerprint acquisition, optical or semi-conduct sensors are widely used. They have high efficiency and acceptable accuracy except for some cases that the user’s finger is too dirty or dry. However, the testing database for our project is from the available fingerprints provided by FVC2002 (Fingerprint Verification Competition 2002). So no acquisition stage is implemented.

The minutia extractor and minutia matcher modules are explained in detail in the next part for algorithm design and other subsequent sections.

**2.1.2. Algorithm Level Design**

To implement a minutia extractor, a three-stage approach is widely used by researchers. They are preprocessing, minutia extraction and post-processing stage [Figure 3.2].

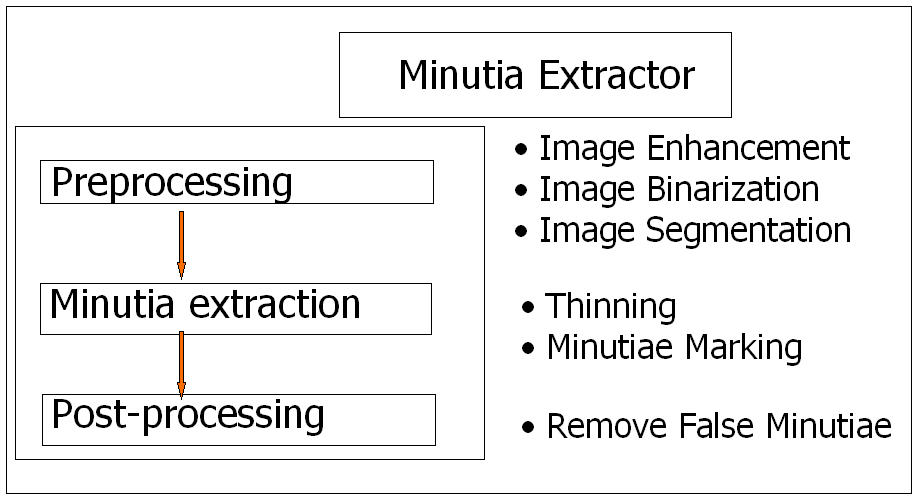


Figure 2.2 Minutia Extractor

For the fingerprint image preprocessing stage, we use Histogram Equalization and Fourier Transform to do image enhancement. And then the fingerprint image is binarized using the locally adaptive threshold method. The image segmentation task is fulfilled by a three-step approach: block direction estimation, segmentation by direction intensity and Region of Interest extraction by Morphological operations. Most methods used in the preprocessing stage are developed by other researchers but they form a brand new combination in our project through trial and error. Also the morphological operations for extraction ROI are introduced to fingerprint image segmentation by us.

For minutia extraction stage, three thinning algorithm are tested and the Morphological thinning operation is finally bid out with high efficiency and pretty good thinning quality. The minutia marking is a simple task as most literatures reported but one special case is found during our implementation and an additional check mechanism is enforced to avoid such kind of oversight.

For the post-processing stage, a more rigorous algorithm is developed to remove false minutia. Also a novel representation for bifurcations is proposed to unify terminations and bifurcations.

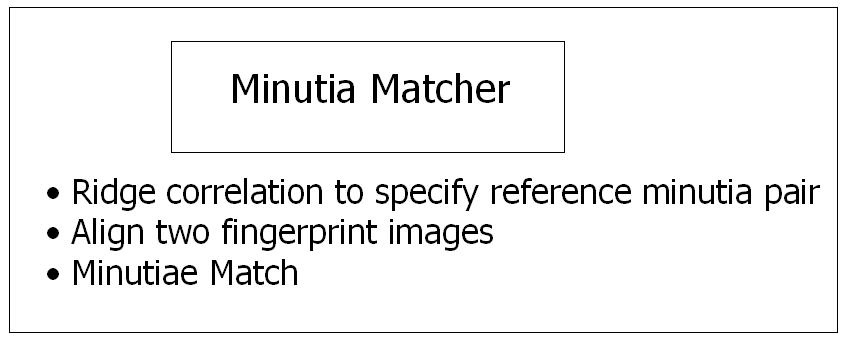


Figure 2.3 Minutia Matcher

The minutia matcher chooses any two minutia as a reference minutia pair and then match their associated ridges first. If the ridges match well, two fingerprint images are aligned and matching is conducted for all remaining minutia [Figure 2.3].

**2.2. Fingerprint Image Pre-Processing**

**2.2.1 Fingerprint Image Enhancement**

Fingerprint Image enhancement is to make the image clearer for easy further operations. Since the fingerprint images acquired from sensors or other medias are not assured with perfect quality, those enhancement methods, for increasing the contrast between ridges and furrows and for connecting the false broken points of ridges due to insufficient amount of ink, are very useful for keep a higher accuracy to fingerprint recognition.

Two Methods are adopted in our fingerprint recognition system: the first one is Histogram Equalization; the next one is Fourier Transform.

**2.2.2. Histogram Equalization:**

Histogram equalization is to expand the pixel value distribution of an image so as to increase the perceptional information. The original histogram of a fingerprint image has the bimodal type [Figure 2.4], the histogram after the histogram equalization occupies all the range from 0 to 255 and the visualization effect is enhanced [Figure 2.4].

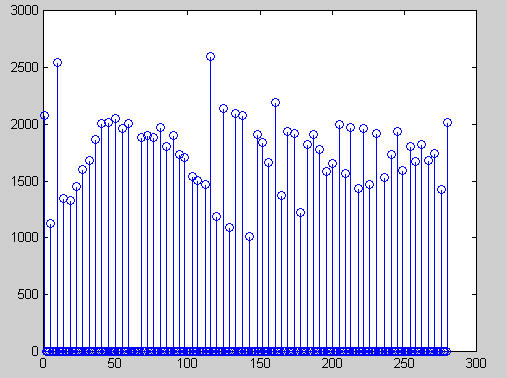
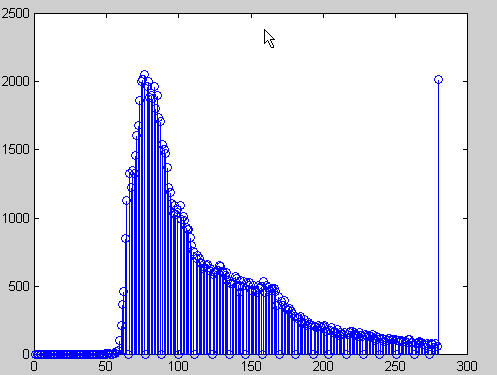


Figure 2.4 The right side of the following figure is the output after the histogram equalization.

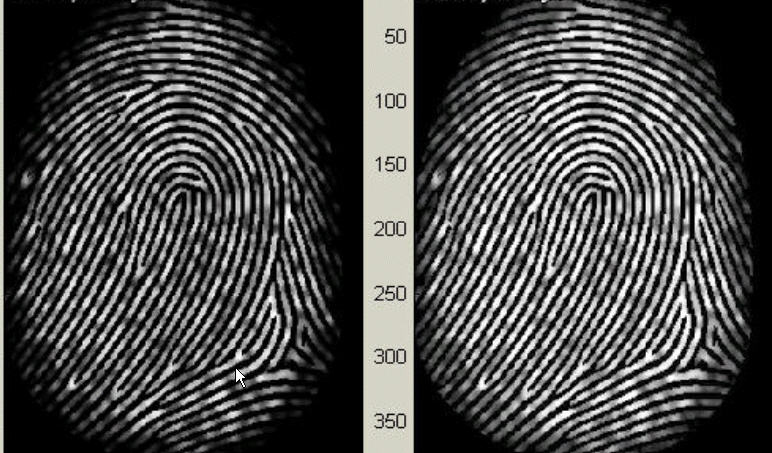


Figure 2.5 Histogram Enhancement. Original Image (Left). Enhanced image (Right)

**2.2.3 Fingerprint Enhancement by Fourier Transform**

We divide the image into small processing blocks (32 by 32 pixels) and perform the Fourier transform according to:

http://ise0.stanford.edu/class/ee368a_proj01/dropbox/project22/finger/pre.ht8.gif     (1)

for u = 0, 1, 2, ..., 31 and v = 0, 1, 2, ..., 31.

In order to enhance a specific block by its dominant frequencies, we multiply the FFT of the block by its magnitude a set of times. Where the magnitude of the original FFT = 

Get the enhanced block according to

http://ise0.stanford.edu/class/ee368a_proj01/dropbox/project22/finger/pre.ht9.gif  (2) ,

where F-1(F(u,v)) is done by:

http://ise0.stanford.edu/class/ee368a_proj01/dropbox/project22/finger/pre.ht10.gif    (3)

for x = 0, 1, 2, ..., 31 and y = 0, 1, 2, ..., 31.

The k in formula (2) is an experimentally determined constant, which we choose k=0.45 to calculate. While having a higher "k" improves the appearance of the ridges, filling up small holes in ridges, having too high a "k" can result in false joining of ridges. Thus a termination might become a bifurcation. Figure 2.6 presents the image after FFT enhancement.

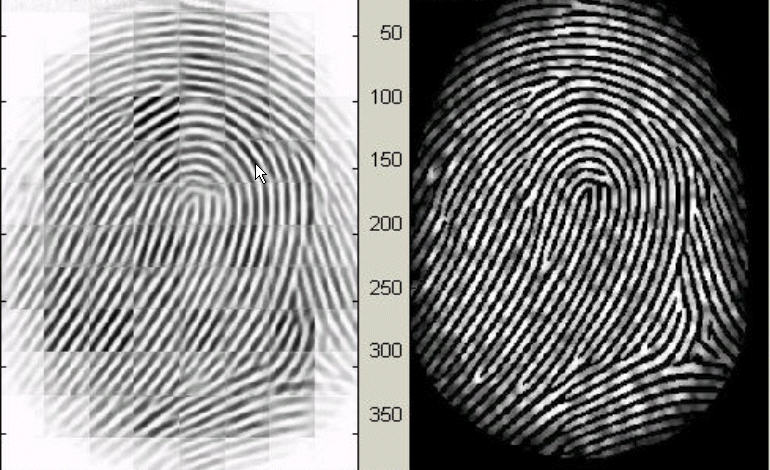


Figure 2.6 Fingerprint Enhancement by FFT Enhanced image (left), Original image (right)

The enhanced image after FFT has the improvements to connect some falsely broken points on ridges and to remove some spurious connections between ridges. The shown image at the left side of figure 2.6 is also processed with histogram equalization after the FFT transform. The side effect of each block is obvious but it has no harm to the further operations because I find the image after consecutive binarization operation is pretty good as long as the side effect is not too severe.

**2.3 Fingerprint Image Binarization**

Fingerprint Image Binarization is to transform the 8-bit Gray fingerprint image to a 1-bit image with 0-value for ridges and 1-value for furrows. After the operation, ridges in the fingerprint are highlighted with black color while furrows are white.

A locally adaptive binarization method is performed to binarize the fingerprint image. Such a named method comes from the mechanism of transforming a pixel value to 1 if the value is larger than the mean intensity value of the current block (16x16) to which the pixel belongs [Figure 2.7].

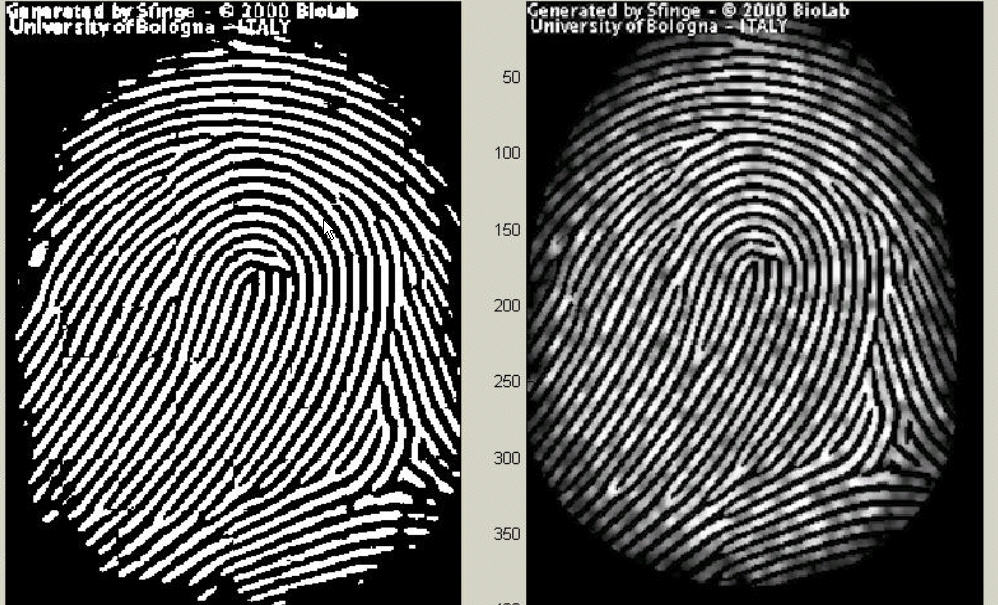


Figure 2.7 the Fingerprint image after adaptive binarization  
Binarized image(left), Enhanced gray image(right)

**2.4. Fingerprint Image Segmentation**

In general, only a Region of Interest (ROI) is useful to be recognized for each fingerprint image. The image area without effective ridges and furrows is first discarded since it only holds background information. Then the bound of the remaining effective area is sketched out since the minutia in the bound region are confusing with those spurious minutia that are generated when the ridges are out of the sensor.

To extract the ROI, a two-step method is used. The first step is block direction estimation and direction variety check [1], while the second is intrigued from some Morphological methods.

**2.4.1. Block direction estimation**

1.1 Estimate the block direction for each block of the fingerprint image with WxW in size (W is 16 pixels by default). The algorithm is:

1. Calculate the gradient values along x-direction (gx) and y-direction (gy) for each pixel of the block. Two Sobel filters are used to fulfill the task.
2. For each block, use Following formula to get the Least Square approximation of the block direction.

 for all the pixels in each block.

The formula is easy to understand by regarding gradient values along x-direction and y-direction as cosine value and sine value. So the tangent value of the block direction is estimated nearly the same as the way illustrated by the following formula.



1.2 After finished with the estimation of each block direction, those blocks without significant information on ridges and furrows are discarded based on the following formulas:



For each block, if its certainty level E is below a threshold, then the block is regarded as a background block.

The direction map is shown in the following diagram. We assume there is only one fingerprint in each image.

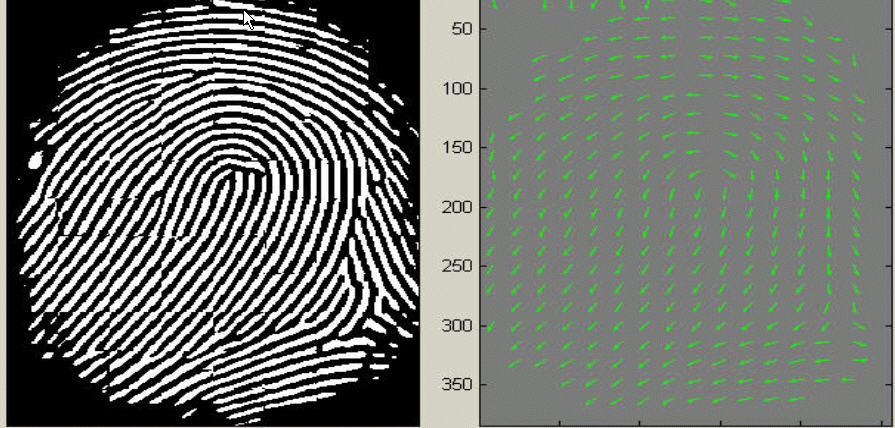


Figure 2.8 Direction map. Binarized fingerprint (left), Direction map (right)

**2.4.2. ROI extraction by Morphological operations**

Two Morphological operations called ‘OPEN’ and ‘CLOSE’ are adopted. The ‘OPEN’ operation can expand images and remove peaks introduced by background noise [Figure 3.9.2]. The ‘CLOSE’ operation can shrink images and eliminate small cavities [Figure 2.9.3].

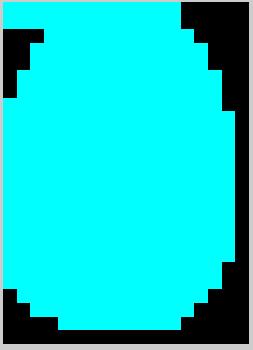
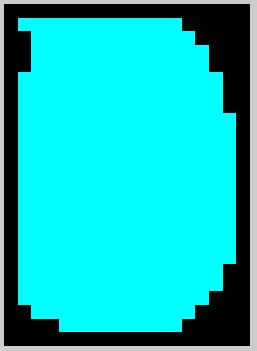


Figure 2.9.1 Original Image Area Figure 2.9.2 After CLOSE operation

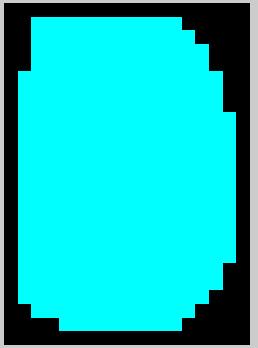
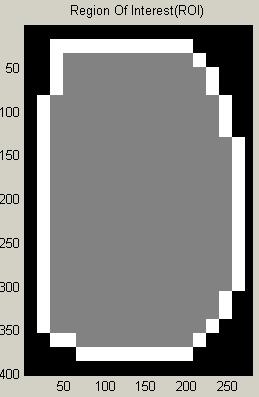


Figure 2.9.3 After OPEN operation Figure 2.9.4 ROI + Bound

Above Figures show the interest fingerprint image area and its bound. The bound is the subtraction of the closed area from the opened area. Then the algorithm throws away those leftmost, rightmost, uppermost and bottommost blocks out of the bound so as to get the tightly bounded region just containing the bound and inner area.

**2.5. Minutia Extraction**

**2.5.1. Fingerprint Ridge Thinning**

Ridge Thinning is to eliminate the redundant pixels of ridges till the ridges are just one pixel wide. Paper uses an iterative, parallel thinning algorithm. In each scan of the full fingerprint image, the algorithm marks down redundant pixels in each small image window (3x3). And finally removes all those marked pixels after several scans. In our testing, such an iterative, parallel thinning algorithm has bad efficiency although it can get an ideal thinned ridge map after enough scans. We use a one-in-all method to extract thinned ridges from gray-level fingerprint images directly. Their method traces along the ridges having maximum gray intensity value. However, binarization is implicitly enforced since only pixels with maximum gray intensity value are remained. Also in our testing, the advancement of each trace step still has large computation complexity although it does not require the movement of pixel by pixel as in other thinning algorithms. Thus the third method is bid out which uses the built-in Morphological thinning function in MATLAB.

**2.5.2. Minutia Marking**

After the fingerprint ridge thinning, marking minutia points is relatively easy. But it is still not a trivial task as most literatures declared because at least one special case evokes our caution during the minutia marking stage.

In general, for each 3x3 window, if the central pixel is 1 and has exactly 3 one-value neighbors, then the central pixel is a ridge branch [Figure 2.10.1]. If the central pixel is 1 and has only 1 one-value neighbor, then the central pixel is a ridge ending [Figure 2.10.2].

1

0

0

0

1

0

0

0

0

0

0

1

1

1

0

0

1

0

1

0

1

0

1

0

0

1

0

Figure 2.10.1 Bifurcation Figure 2.10.2 Termination Figure 2.10.3 Triple counting branch

Figure 2.10 illustrates a special case that a genuine branch is triple counted. Suppose both the uppermost pixel with value 1 and the rightmost pixel with value 1 have another neighbor outside the 3x3 window, so the two pixels will be marked as branches too. But actually, only one branch is located in the small region. So a check routine requiring that none of the neighbors of a branch are branches is added.

Also the average inter-ridge width D is estimated at this stage. The average inter-ridge width refers to the average distance between two neighboring ridges. The way to approximate the D value is simple. Scan a row of the thinned ridge image and sum up all pixels in the row whose value is one. Then divide the row length with the above summation to get an inter-ridge width. For more accuracy, such kind of row scan is performed upon several other rows and column scans are also conducted, finally all the inter-ridge widths are averaged to get the D.

Together with the minutia marking, all thinned ridges in the fingerprint image are labeled with a unique ID for further operation. The labeling operation is realized by using the Morphological operation: BWLABEL.

**2.6. Minutia Post-processing**

**2.6.1. False Minutia Removal**

The preprocessing stage does not totally heal the fingerprint image. For example, false ridge breaks due to insufficient amount of ink and ridge cross-connections due to over inking are not totally eliminated. Actually all the earlier stages themselves occasionally introduce some artifacts which later lead to spurious minutia. These false minutia will significantly affect the accuracy of matching if they are simply regarded as genuine minutia.

Seven types of false minutia are specified in following diagrams:

m1 m2 m3 m4

m5 m6 m7

[Figure 3.11. False Minutia Structures. m1 is a spike piercing into a valley. In the m2 case a spike falsely connects two ridges. m3 has two near bifurcations located in the same ridge. The two ridge broken points in the m4 case have nearly the same orientation and a short distance. m5 is alike the m4 case with the exception that one part of the broken ridge is so short that another termination is generated. m6 extends the m4 case but with the extra property that a third ridge is found in the middle of the two parts of the broken ridge. m7 has only one short ridge found in the threshold window.]

It only handles the case m1, m4, m5 and m6. The referred papers have not false minutia removal by simply assuming the image quality is fairly good. It has not a systematic healing method to remove those spurious minutia although it lists all types of false minutia shown in Figure except the m3 case.

Our procedures in removing false minutia are:

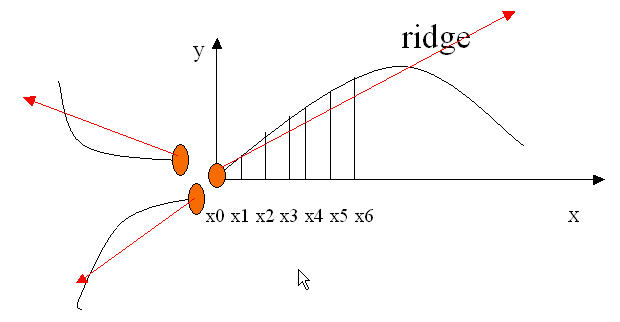
1. If the distance between one bifurcation and one termination is less than D and the two minutia are in the same ridge(m1 case) . Remove both of them. Where D is the average inter-ridge width representing the average distance between two parallel neighboring ridges.
2. If the distance between two bifurcations is less than D and they are in the same ridge, remove the two bifurcations. (m2, m3 cases).
3. If two terminations are within a distance D and their directions are coincident with a small angle variation. And they suffice the condition that no any other termination is located between the two terminations. Then the two terminations are regarded as false minutia derived from a broken ridge and are removed. (case m4,m5, m6).
4. If two terminations are located in a short ridge with length less than D, remove the two terminations (m7).

Our proposed procedures in removing false minutia have two advantages. One is that the ridge ID is used to distinguish minutia and the seven types of false minutia are strictly defined comparing with those loosely defined by other methods. The second advantage is that the order of removal procedures is well considered to reduce the computation complexity. It surpasses the way adopted by [12] that does not utilize the relations among the false minutia types. For example, the procedure3 solves the m4, m5 and m6 cases in a single check routine. And after procedure 3, the number of false minutia satisfying the m7 case is significantly reduced.

**2.6.2. Unify terminations and bifurcations**

Since various data acquisition conditions such as impression pressure can easily change one type of minutia into the other, most researchers adopt the unification representation for both termination and bifurcation. So each minutia is completely characterized by the following parameters at last: 1) x-coordinate, 2) y-coordinate, and 3) orientation.

The orientation calculation for a bifurcation needs to be specially considered. All three ridges deriving from the bifurcation point have their own direction, [9] represents the bifurcation orientation using a technique proposed in the paper. It simply chooses the minimum angle among the three anticlockwise orientations starting from the x-axis. Both methods cast the other two directions away, so some information loses. Here I propose a novel representation to break a bifurcation into three terminations. The three new terminations are the three neighbor pixels of the bifurcation and each of the three ridges connected to the bifurcation before is now associated with a termination respectively [Figure 3.12].



*1*

0

0

0

1

*1*

*1*

0

0

Figure 2.12 A bifurcation to three terminations Three neighbors become terminations (Left) Each termination has their own orientation (Right)

And the orientation of each termination (tx,ty) is estimated by following method [book ch2]:

Track a ridge segment whose starting point is the termination and length is D. Sum up all x-coordinates of points in the ridge segment. Divide above summation with D to get sx. Then get sy using the same way.

Get the direction from: 

**2.7. Minutia Match**

Given two set of minutia of two fingerprint images, the minutia match algorithm determines whether the two minutia sets are from the same finger or not.

An alignment-based match algorithm partially derived from the Paper is used in our project. It includes two consecutive stages: one is alignment stage and the second is match stage.

1. Alignment stage. Given two fingerprint images to be matched, choose any one minutia from each image, calculate the similarity of the two ridges associated with the two referenced minutia points. If the similarity is larger than a threshold, transform each set of minutia to a new coordination system whose origin is at the referenced point and whose x-axis is coincident with the direction of the referenced point.
2. Match stage: After we get two set of transformed minutia points, we use the elastic match algorithm to count the matched minutia pairs by assuming two minutia having nearly the same position and direction are identical.

**2.7.1. Alignment Stage**

1. The ridge associated with each minutia is represented as a series of x-coordinates (x1, x2…xn) of the points on the ridge. A point is sampled per ridge length L starting from the minutia point, where the L is the average inter-ridge length. And n is set to 10 unless the total ridge length is less than 10\*L.

So the similarity of correlating the two ridges is derived from:



where  and  are the set of minutia for each fingerprint image respectively. And m is minimal one of the n and N value. If the similarity score is larger than 0.8, then go to step 2, otherwise continue to match the next pair of ridges.

1. For each fingerprint, translate and rotate all other minutia with respect to the reference minutia according to the following formula:

where (x,y,θ) is the parameters of the reference minutia, and TM is

The following diagram illustrates the effect of translation and rotation:



The new coordinate system is originated at minutia F and the new x-axis is coincident with the direction of minutia F. No scaling effect is taken into account by assuming two fingerprints from the same finger have nearly the same size.

Our method to align two fingerprints is almost the same with the one used by [1] but is different at step 2. Lin’s method uses the rotation angle calculated from all the sparsely sampled ridge points. Our method use the rotation angle calculated earlier by densely tracing a short ridge start from the minutia with length D. Since I have already got the minutia direction at the minutia extraction stage, obviously our method reduces the redundant calculation but still holds the accuracy.

Also Lin’s way to do transformation is to directly align one fingerprint image to another according to the discrepancy of the reference minutia pair. But it still requires a transform to the polar coordinate system for each image at the next minutia match stage. Our approach is to transform each according to its own reference minutia and then do match in a unified x-y coordinate. Therefore, less computation workload is achieved through our method.

**2.7.2. Match Stage**

The matching algorithm for the aligned minutia patterns needs to be elastic since the strict match requiring that all parameters (x, y, θ) are the same for two identical minutia is impossible due to the slight deformations and inexact quantizations of minutia.

Our approach to elastically match minutia is achieved by placing a bounding box around each template minutia. If the minutia to be matched is within the rectangle box and the direction discrepancy between them is very small, then the two minutia are regarded as a matched minutia pair. Each minutia in the template image either has no matched minutia or has only one corresponding minutia.

The final match ratio for two fingerprints is the number of total matched pair over the number of minutia of the template fingerprint. The score is 100\*ratio and ranges from 0 to 100. If the score is larger than a pre-specified threshold, the two fingerprints are from the same finger.

However, the elastic match algorithm has large computation complexity and is vulnerable to spurious minutia.

**Chapter 3: Motivation and Approach**

**3.1 Problem Statement**

In a common style, this will be a difficulty to know and differentiate the types of fingerprint. In this world, there are many types of marble and have a little different among them in the decorative design. We could not able to know the type of fingerprint which have the feature had been needed.

By using ANN toolbox, We able to assign an input pattern or train the network. This ability of the network is to recognize the data features that were extracted from the image with a little differentiate. Adding with using Graphical User Interface (GUI) almost inside the MATLAB is to be user friendly when using this system.

**3.2 Proposed Method**

Applications of fingerprint recognition can be found in security, tracking, multimedia, and entertainment domains. We have demonstrated how a fingerprint recognition system can be designed by a Artificial neural network, to capture the minutiae characteristics of fingerprint, to simulate the human visual system, although artificial neural network designed as a tool for modeling biological neural networks, the level of performance obtained with artificial neural network is such that in a variety of tasks, processing architectures developed using artificial neural network can perform at least as well and in many cases substantially better than more conventional image processing and pattern recognition techniques.

**Neural Networks:**

A neural network, also known as a parallel distributed processing network, is a computing solution that is loosely modeled after cortical structures of the brain. It consists of interconnected processing elements called nodes or neurons that work together to produce an output function. The output of a neural network relies on the cooperation of the individual neurons within the network to operate. Processing of information by neural networks is characteristically done in parallel rather than in series (or sequentially) as in earlier binary computers or Von Neumann machines.

Neural network theory is sometimes used to refer to a branch of computational science that uses neural networks as models to simulate or analyze complex phenomena and/or study the principles of operation of neural networks analytically. It addresses problems similar to artificial intelligence (AI) except that AI uses traditional computational algorithms to solve problems whereas neural networks use 'networks of agents' (software or hardware entities linked together) as the computational architecture to solve problems. Neural networks are trainable systems that can "learn" to solve complex problems from a set of exemplars and generalize the "acquired knowledge" to solve unforeseen problems as in stock market and environmental prediction. I.e., they are self-adaptive systems. The term 'Neural Network' has two distinct connotations:

i. Biological neural networks are made up of real biological neurons that are connected or functionally-related in the peripheral nervous system or the central nervous system. In the field of neuroscience, they are often identified as groups of neurons that perform a specific physiological function in laboratory analysis.

ii. Artificial neural networks are made up of interconnecting artificial neurons (usually simplified neurons) designed to model (or mimic) some properties of biological neural networks. Artificial neural networks can be used to model the modes of operation of biological neural networks, whereas cognitive models are theoretical models that mimic cognitive brain functions without necessarily using neural networks while artificial intelligence are well-crafted algorithms that solve specific intelligent problems (such as chess playing, pattern recognition, etc.) without using neural network as the computational architecture.

An artificial neural network (ANN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical model or computational model for information processing based on a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network.

Conventional approaches have been proposed for solving these problems. Although successful applications can be found in certain well-constrained environments, none is flexible enough to perform well outside its domain. ANNs provide exciting alternatives, and many applications could benefit from using them.

**Graphical User Interface (GUI)**

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand. The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders.

In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components. The applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work.

**3.3. Organization of Report**

The report consists of seven chapters.

Chapter 1: Introduction to the evolution of fingerprint recognition.

Chapter 2: It consists of literature survey made on the project.

Chapter 3: It encapsulates the problem statement and proposed methodology.

Chapter 4: It focuses on system analysis, design and algorithm details.

Chapter 5: It includes system implementation and the screenshots.

Chapter 6: It shows the experiment details and conclusion.

Chapter 7: It highlights on future scope of the system.

**Chapter 4: System Analysis and Design**

**Analysis and Algorithm Details**

In our project, we will try to do fingerprint classification by using self-organization map (SOM). Mainly guided by the paper, both a conventional and a modified SOM algorithm are used to fulfill the task. Also different approaches from the literature for retrieving the feature vectors of fingerprints and for fully making use of the available SOM toolbox play a critical role to keep my classifier system efficient and robust.

The SOM is proved to be truly useful for classifying the difficult fingerprint classification problem. Also the classifier has the strength of being extended by introducing more complicated SOM architectures or further utilizing advanced statistical method like principal component analysis (PCA).

**4.1. What is SOM?**

A **self-organizing map (SOM)** or self-organizing feature map (SOFM) is a type of [**artificial neural network**](http://en.wikipedia.org/wiki/Artificial_neural_network) that is trained using [unsupervised learning](http://en.wikipedia.org/wiki/Unsupervised_learning) to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map. Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the [topological](http://en.wikipedia.org/wiki/Topology) properties of the input space.

This makes SOMs useful for [visualizing](http://en.wikipedia.org/wiki/Scientific_visualization) low-dimensional views of high-dimensional data, akin to [multidimensional scaling](http://en.wikipedia.org/wiki/Multidimensional_scaling). The model was first described as an artificial neural network by the [Finnish](http://en.wikipedia.org/wiki/Finland) professor [Teuvo Kohonen](http://en.wikipedia.org/wiki/Teuvo_Kohonen), and is sometimes called a **Kohonen map**.

Like most artificial neural networks, SOMs operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called [vector quantization](http://en.wikipedia.org/wiki/Vector_quantization). Mapping automatically classifies a new input vector.

A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. The usual arrangement of nodes is a regular spacing in a [hexagonal](http://en.wikipedia.org/wiki/Hexagonal) or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The procedure for placing a vector from data space onto the map is to first find the node with the closest weight vector to the vector taken from data space. Once the closest node is located it is assigned the values from the vector taken from the data space.

While it is typical to consider this type of network structure as related to [feed forward networks](http://en.wikipedia.org/wiki/Feedforward_neural_networks) where the nodes are visualized as being attached, this type of architecture is fundamentally different in arrangement and motivation.

Useful extensions include using [toroidal](http://en.wikipedia.org/wiki/Torus) grids where opposite edges are connected and using large numbers of nodes. It has been shown that while self-organizing maps with a small number of nodes behave in a way that is similar to [K-means](http://en.wikipedia.org/wiki/K-means_algorithm), larger self-organizing maps rearrange data in a way that is fundamentally topological in character.

It is also common to use the [U-Matrix](http://en.wikipedia.org/wiki/U-Matrix). The U-Matrix value of a particular node is the average distance between the node and its closest neighbors (ref. 9). In a square grid for instance, we might consider the closest 4 or 8 nodes (the [Von Neumann neighbourhood](http://en.wikipedia.org/wiki/Von_Neumann_neighbourhood) and [Moore neighbourhood](http://en.wikipedia.org/wiki/Moore_neighbourhood) respectively), or six nodes in a hexagonal grid.

Large SOMs display properties which are emergent. In maps consisting of thousands of nodes, it is possible to perform cluster operations on the map itself.

**4.2. Learning Algorithm:**

The goal of learning in the self-organizing map is to cause different parts of the network to respond similarly to certain input patterns. This is partly motivated by how visual, auditory or other [sensory](http://en.wikipedia.org/wiki/Sense) information is handled in separate parts of the [cerebral cortex](http://en.wikipedia.org/wiki/Cerebral_cortex) in the [human brain](http://en.wikipedia.org/wiki/Human_brain).

The weights of the neurons are initialized either to small random values or sampled evenly from the subspace spanned by the two largest [principal component](http://en.wikipedia.org/wiki/Principal_component) [eigenvectors](http://en.wikipedia.org/wiki/Eigenvectors). With the latter alternative, learning is much faster because the initial weights already give good approximation of SOM weights.

The network must be fed a large number of example vectors that represent, as close as possible, the kinds of vectors expected during mapping. The examples are usually administered several times as iterations.

The training utilizes [competitive learning](http://en.wikipedia.org/wiki/Competitive_learning). When a training example is fed to the network, its [Euclidean distance](http://en.wikipedia.org/wiki/Euclidean_distance) to all weight vectors is computed. The neuron with weight vector most similar to the input is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU. The update formula for a neuron with weight vector ***Wv***(t) is



where α(t) is a [monotonically decreasing](http://en.wikipedia.org/wiki/Monotonically_decreasing) learning coefficient and **D**(t) is the input vector. The neighborhood function Θ (v, t) depends on the lattice distance between the BMU and neuron *v*. In the simplest form it is one for all neurons close enough to BMU and zero for others, but a [Gaussian function](http://en.wikipedia.org/wiki/Gaussian_function) is a common choice, too. Regardless of the functional form, the neighborhood function shrinks with time.[[3]](http://en.wikipedia.org/wiki/Self-organizing_map#cite_note-Haykin-2) At the beginning when the neighborhood is broad, the self-organizing takes place on the global scale. When the neighborhood has shrunk to just a couple of neurons the weights are converging to local estimates.

This process is repeated for each input vector for a (usually large) number of cycles **λ**. The network winds up associating output nodes with groups or patterns in the input data set. If these patterns can be named, the names can be attached to the associated nodes in the trained net.

During mapping, there will be one single *winning* neuron: the neuron whose weight vector lies closest to the input vector. This can be simply determined by calculating the Euclidean distance between input vector and weight vector.

While representing input data as vectors has been emphasized in this article, it should be noted that any kind of object which can be represented digitally and which has an appropriate distance measure associated with it and in which the necessary operations for training are possible can be used to construct a self-organizing map. This includes matrices, continuous functions or even other self-organizing maps.

**4.3. Training Algorithms:**

**4.3.1. Conventional SOM:**

1. Construct a  SOM, initialize all the weights
2. Input a fingerprint vector: *X*
3. Find the winning node  where:  = 
4. Update the weight vectors:



Where N is the neighborhood function corresponding to the SOM node topology

1. Repeat 2-4 till Update is not significant.

**4.3.2. Modified SOM:**

Note: Each fingerprint is associated with a certainty vector C

1. Construct a  SOM, initialize all weights
2. Input a fingerprint vector: *X* = *C\*X + (1-C)\*Xavg;*
3. Find the winning node  where:  = 
4. Update the weight vectors:

 *C*

Where N is the neighborhood function corresponding to the SOM  
 node topology

1. Repeat 2-4 till Update is not significant.

Figure 4.1 Finding the Winning Node

**4.4. Classification of Fingerprints:**

We divide fingerprint into five classes - arch or tented arch, left loop, right loop, whorl and unclassified.

We are going to use Self Organizing Maps (SOM) Toolbox for the Classification of Fingerprints.

Figure 4.2 Fingerprint Classification

**4.5. Feature vector Generation**

A 256-dimension feature vector for each fingerprint image is generated. The feature vector generation stage is an important part for keeping the accuracy of the classification system. Nearly half time is paid to this part.

The idea can be concluded into four steps. Firstly the block directional image and the certainty value associated with each block are calculated. Then the segmentation of the fingerprint image so that noisy and corrupted parts that do not carry valid information are deleted. Meanwhile the core point to be taken as the reference center is extracted. Then a 16x16 block directions surrounding to the core are composed into the feature vector.

1. ***Block Direction Estimation***

The block direction estimation program operates on the gray level fingerprint image and then obtains an approximate direction for each image block with size WxW (16x16 by default). The method used in our project is originated from the paper, which calculates the horizontal and vertical gradients of each pixel and then combines all the gradients within the block to get an estimated direction. It has large consistent with the ridge flows of the original fingerprint image in most testing cases. The other reason We chose this algorithm rather than the one used by the paper is that the certainty values can be generated simultaneously during the direction estimation, which definitely reduces the computation cost comparing with the two independent procedures adopted by the paper. All the directions are then normalized into the domain from 0 to pi, and the certain values are in the interval from 0 to 1.

###### **Segmentation**

An inevitable problem after step A is that we cannot eliminate the background noise completely although we have already discarded the image areas that have too small certainty values. Also we need to extract the tightly bounded fingerprint region from the image to benefit the consequent steps.

A series of complicated operations are used to fulfill the segmentation task, which are also totally different from those ones used by the paper, but still keep the efficiency and accuracy high. They include Histogram Equalization, Image enhancement and coarse segmentation by Fourier Transform, Image binarization, Interest region locating by Morphological operations. They are from multiple sources such as literatures, student projects and my own implementation for the honors project. They will not be explained in detail here since we concentrate on the SOM part.

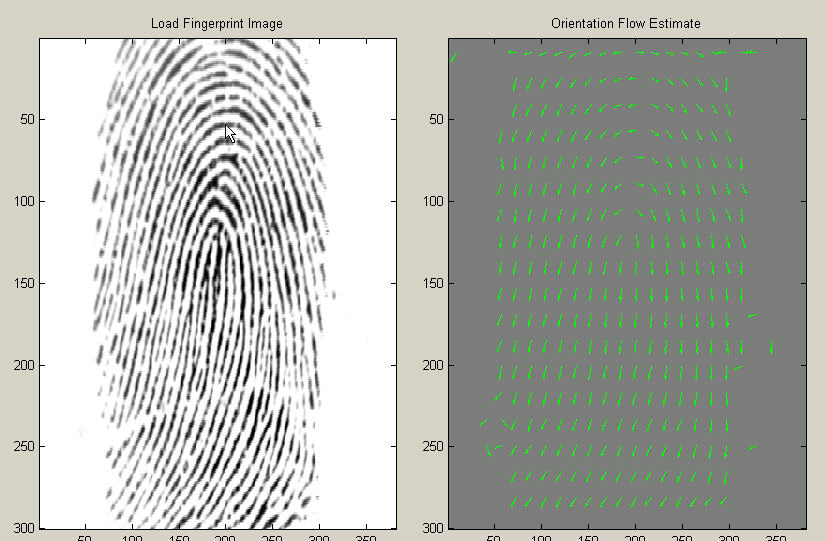


Figure 4.3 Extract Fingerprint region(left, Extract Effective Region(right)

###### **C) Core point detection**

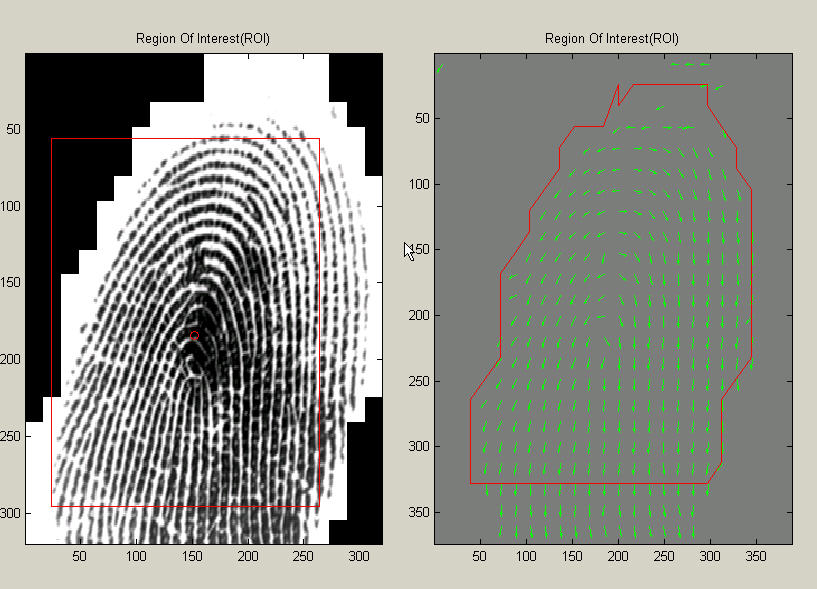


Figure 4.4 Core point detection

The one remaining fundamental step before classification is the core point extraction, that is, the automatic detection of the core point of the fingerprint. This step is particularly important since a reference center is required in order to correctly compare two fingerprints.

But several assumptions and restrictions have to be clarified first. Automated core detection can only find the most likely center of the image without regard whether there is a meaningful core exists or not. Furthermore, the alignment according to the core point only partially remedies the misalignment of two fingerprints since we do not consider the scaling, rotation, and clipping factors.

An algorithm is used to detect the core point but with a slight difference. maps all the block directions to the interval from –0.5 to 0.5 and then simply regards the value 0.5 corresponds to the core. But in my test, values through such a mapping do not have an exact 0.5 in most case, so the region with a largest value in center, which also continuously connects with a largest neighborhood area before the value reduces to a threshold, is used to locate the core. Proved by testing, 90% of fingerprint images can be located the core correctly.

###### **Regulate the feature vector**

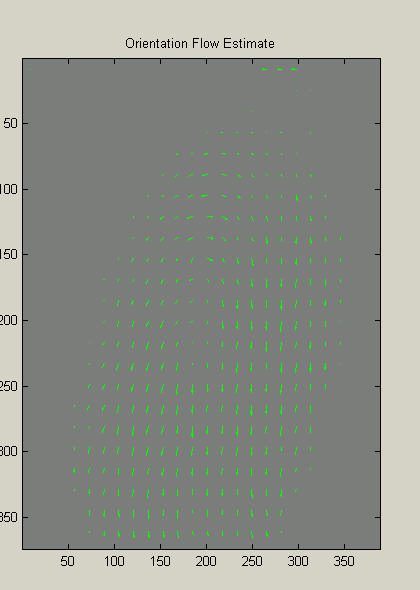


Figure 4.5 Regulate the Feature Vector

Preparing for the feature vector is actually a trivial task after successfully conducting the above three steps. Simply we extract a 16x16 block central at the core and then reconstruct it as a 1x256 vector. For the parts of the block in the background region or outside the image region, we padding *NaN* as the value, which will not be regarded effect vector value in further process. The same operations are enforced to the certainty vector.

###### **4.6. SOM Construction and Training**

A free SOM toolbox is adopted to assist in the project. So the discussion of SOM at the algorithm level is combined with the introduction of the SOM toolbox to make the explanation specific.

1. **The Conventional SOM:**

1. Construct a SOM with size  and assign random values to all the weights corresponding to the  map neurons. It is easily achieved by calling the routine of SOM\_RANDINIT.
2. Sequentially train feature vectors:

Total N feature vectors each with 256 dimensions in size are fed into the SOM. For simplicity, all the N vectors are trained one by one, so N trainings going through each vector one time are regarded as a run of training. Such kind of run is taken K times.

For each input vector, the detailed learning procedures are as follows:

Find the winning output node *Dmin* by the following criterion:

*Dmin = min{||x-wj||}*

Where ||.|| denotes the Euclidean norm and wj is the weight vector connecting input nodes to output node j;

Adjust the weight vectors according to the following update formula:



Where *Wij* is *jth* component of the weight vector , *L(t)* is the learning rate and *N(j,t)* is the neighborhood function.

The neighborhood function is a window centered around the winning node *dmin* , whose radius decreases with time. In the implementation, the radiuses at the beginning and the end can be set explicitly when calling the function SOM\_SEQTRAIN. It is simply set to decrease from the map size m to 1 during all the K runs. The learning rate function L(t) is also a decaying function. It is kept large at the beginning of training and decreased gradually as learning proceeds.

**B. Modified SOM (MSOM)**

The modified SOM accepts the certainty vector as well and considers the certainties during the training stage. The toolbox does not provide such a modified SOM, so some manual work has to be done to implement the MSOM.

The detailed MSOM is like that:

1. Initialize a  SOM. For reducing the variation, all weights of SOM neurons are set to zero.
2. Construct a structure containing a feature vector x and its certainty vector c. And then input the structure into the SOM.

As described in, the current step reforms the feature vector *x* to *xc* accordingwhere is the vector holding the average values xk (k from 1-256) over the whole training sample space.

But in our approach, such a distortion of input data is not enforced, because the Xc after the interpolation might be drastically reformed. It actually cannot be used to “stimulate the components of the input vectors of which we are certain and inhibit the less certain ones” as declared as the paper because attenuating the block with a small certainty or raising the block direction with a good certainty do not ensure the best-matched winning node will be associated due to the large variety of weight values of SOM neurons.

our way to bind the certainty values is explained in the later step.

Find the winning output node Dmin by the following criterion:



Where ||.|| denotes the Euclidean norm and wj is the weight vector connecting input nodes to output node j.

Note that the criterion for Dmin is slightly different from the one for the conventional SOM case, because the calculation for the Euclidean distance is now weighted by the certainty values. It is more reasonable than the simply distortion the input vector by interpolating certainty values.

Adjust the weight vectors of the SOM neurons in the neighborhood area according to the following formula:



Where *Wij* is jth component of the weight vector *Wj*, L(t) is the learning rate and N(j,t) is the neighborhood function.

**4.7. Design**

**4.7.1. System Architecture:**

Figure 4.6 System Architecture

The Architecture consists of four components:

1. User interface
2. System database
3. Enrollment module
4. Authentication module.

* The user interface provides mechanisms for a user to indicate his/her identity and input his/her fingerprints into the system.
* The system database consists of a collection of records, each of which corresponds to an authorized person that has access to the system. Each record contains the following fields which are used for authentication purpose:

User name of the person, Minutiae templates of the person’s fingerprint, and other information (e.g., specific user privileges).

* The task of enrollment module is to enroll persons and their fingerprints into the system database. When the fingerprint images and the user name of a person to be enrolled are fed to the enrollment module, a minutiae extraction algorithm is first applied to the fingerprint images and the minutiae patterns are extracted. A quality checking algorithm is used to ensure that the records in the system database only consist of fingerprints of good quality, in which a significant number (default value is 25) of genuine minutiae may be detected. If a fingerprint image is of poor quality, it is enhanced to improve the clarity of ridge/valley structures and mask out all the regions that cannot be reliably recovered. The enhanced fingerprint image is fed to the minutiae extractor again.
* The task of authentication module is to authenticate the identity of the person who intends to access the system. The person to be authenticated indicates his/her identity and places his/her finger on the fingerprint scanner; a digital image of his/her fingerprint is captured; minutiae pattern is extracted from the captured fingerprint image and fed to a matching algorithm which matches it against the person’s minutiae templates stored in the system database to establish the identity.

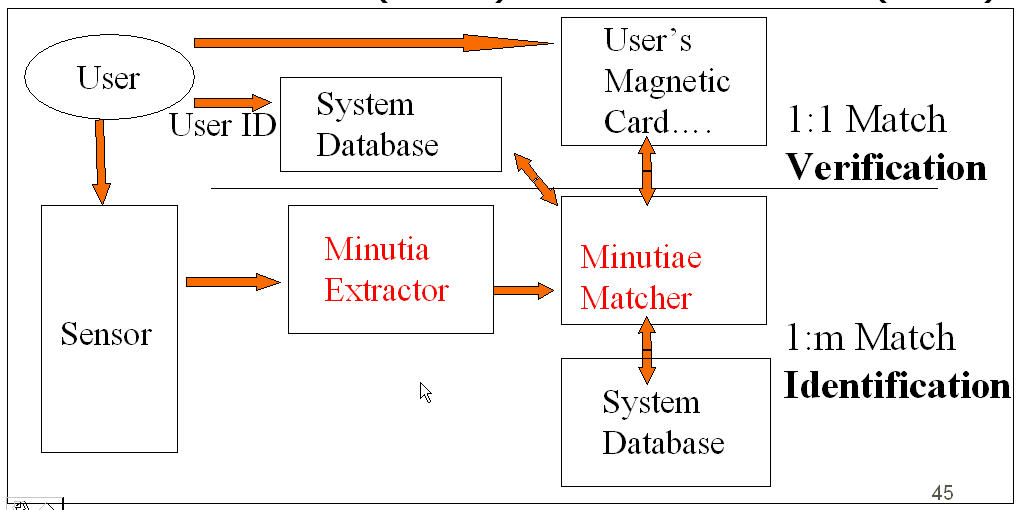


Figure 4.7 Identification & Verification

**Chapter 5: System Implementation**

* 1. **Details of Hardware & Software**

We are going to use the Fingerprint Verification Database (FVC 2002), which will store the images of fingerprints.

We will use Softwares like MATLAB 7.8 and SOM (Self organizing Map) toolbox for using Artificial Neural Networks.

Fingerprint Matching will be done by core point detection using SOM toolbox.

* 1. **Screenshots:**

1. Type command **start\_gui\_single\_mode** in MATLAB 7.8

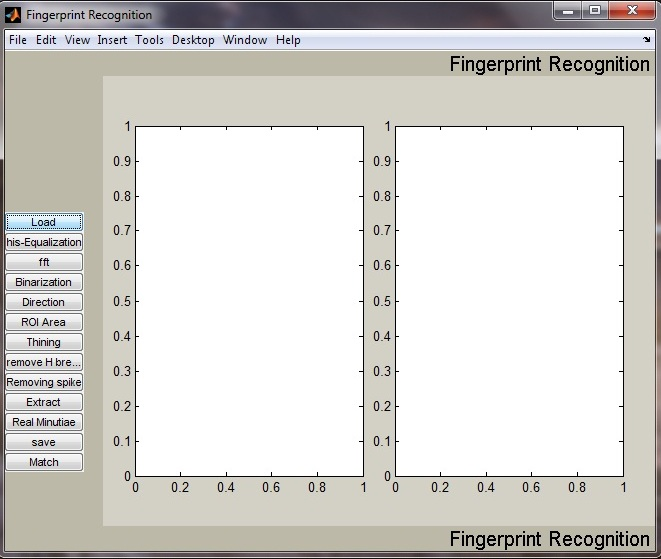


Figure M.1 the User Interface of the Fingerprint Recognition System. The series of buttons on the left side will be invoked sequentially in the consequent demonstration. The two blank areas are used to show the fingerprint image before and after a transaction respectively.

2. Click **Load** Button

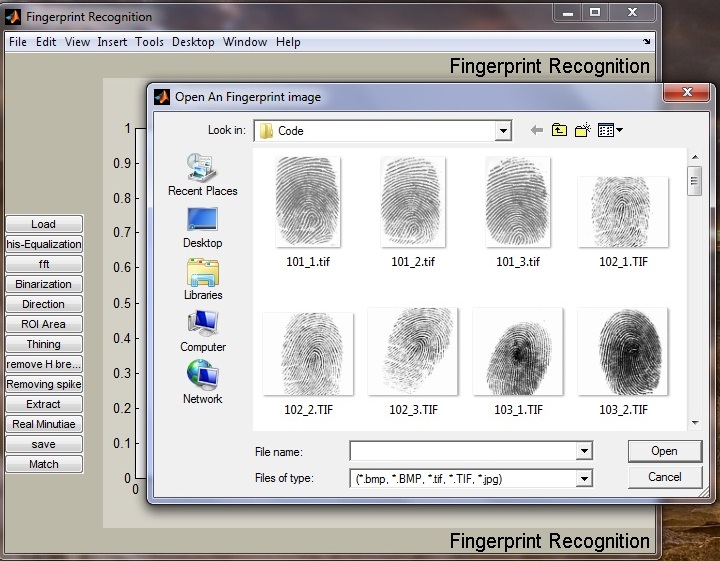
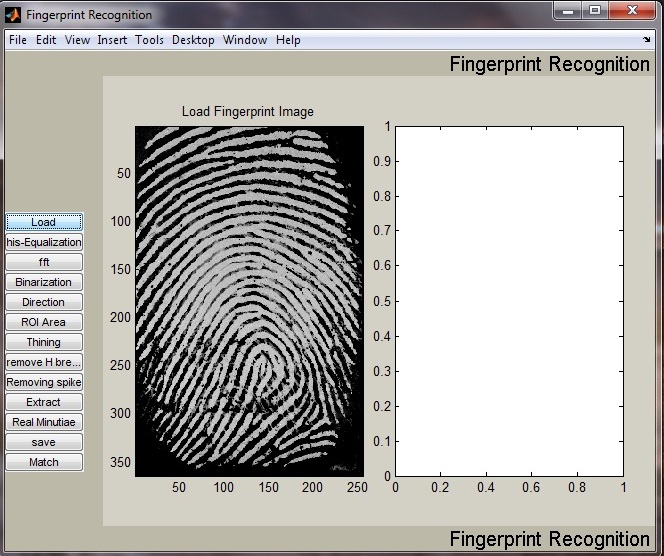
 

Figure M.2 Load a gray level fingerprint image from a drive specified by Users. Multiple formats are supported and the image size is not limited.

3. Click **his-Equalization** Button

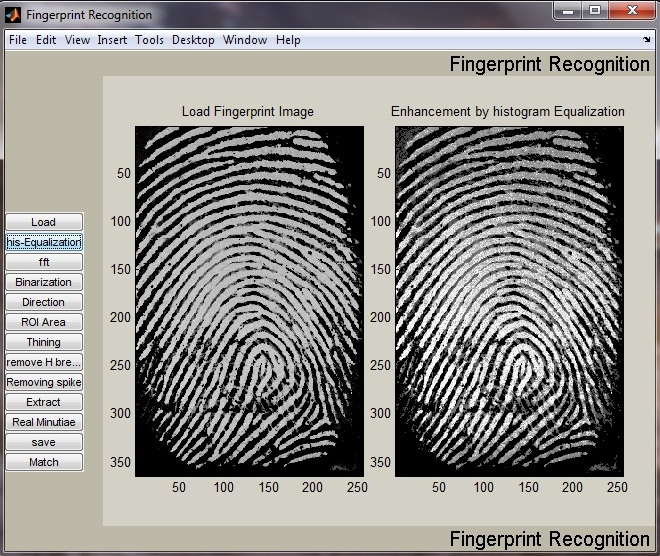


Figure M.3 After Histogram Equalization. The image on the left side is the original fingerprint. The enhanced image after the Histogram Equalization is shown on the right side.

4. Click **fft** Button

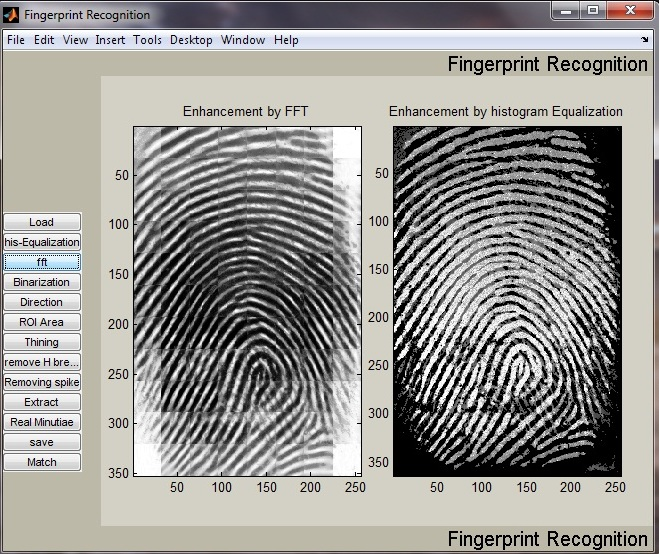
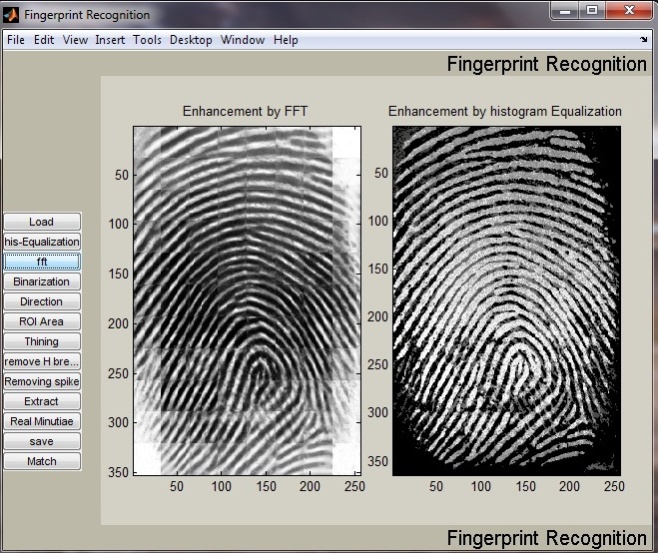


Figure M.4 Captured window after click ‘FFT’ button. The pop-up dialog accepts the parameter k (please refer the formula 2). The experimental optimal k value is 0.45. The enhanced image will be shown in the left screen box, which however is not shown here.

5. Click **Binarization** Button



1. Click **Direction** Button

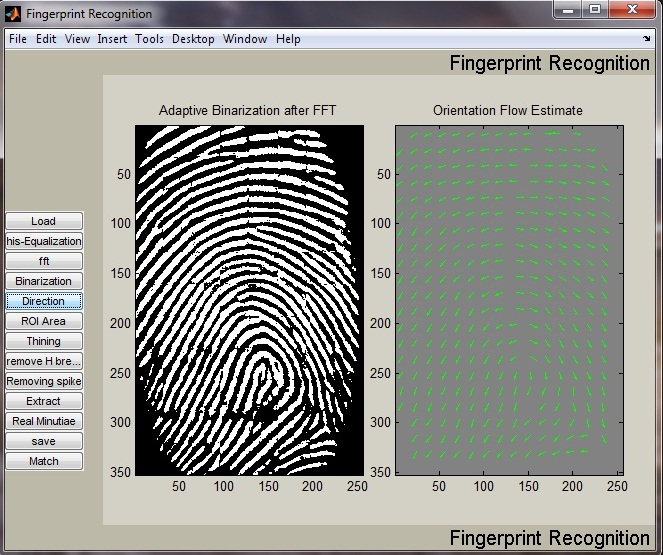


Figure M.5.6 Screen capture after binarization (left) and block direction estimation (right).

7. Click **ROI Area** Button

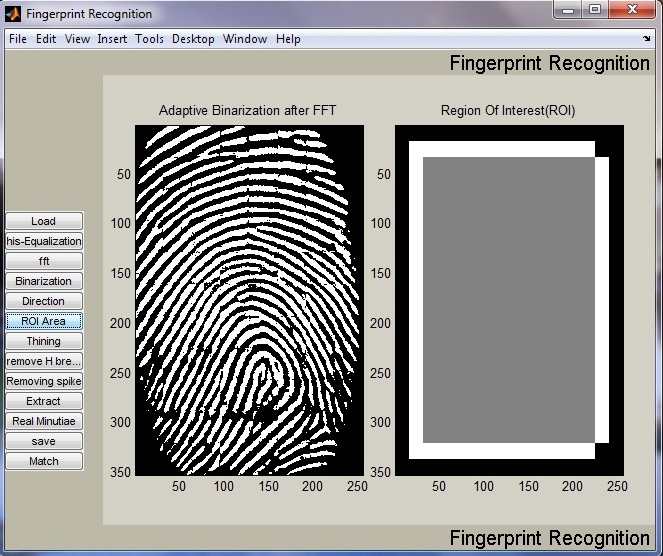
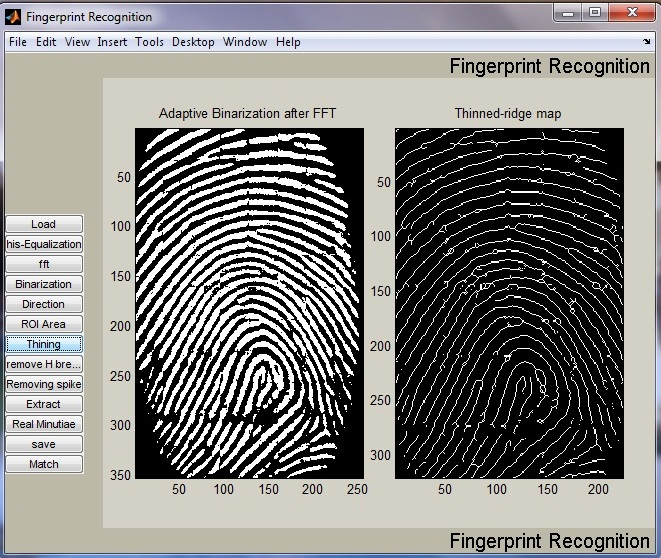
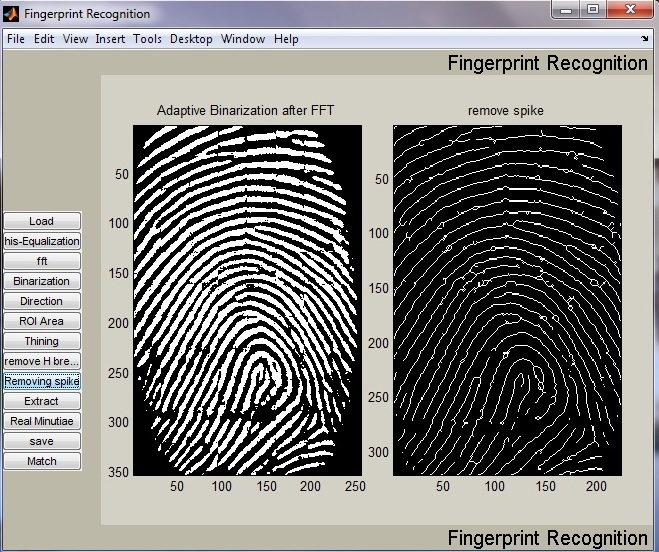


Figure M.7 ROI extraction(right). The right screen box shows the final region of interest of the fingerprint image. The subsequent operations will only operate on the region of interest.

8. Click **Thinning** Button



9. Click **Remove H breaks** Button



10. Click **Remove spikes** Button

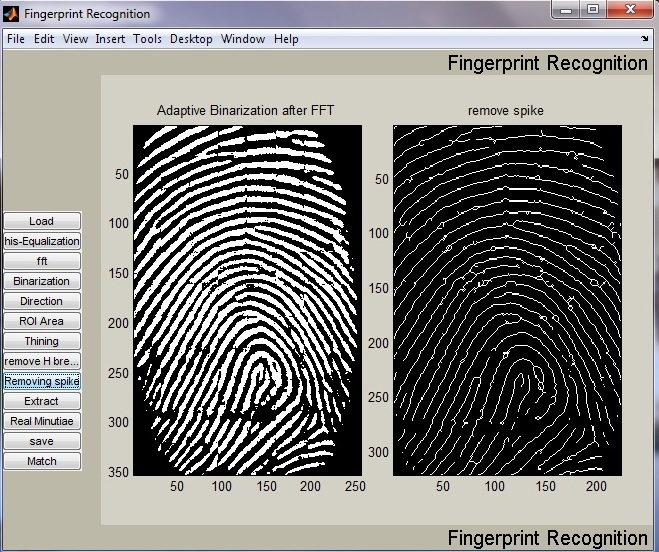
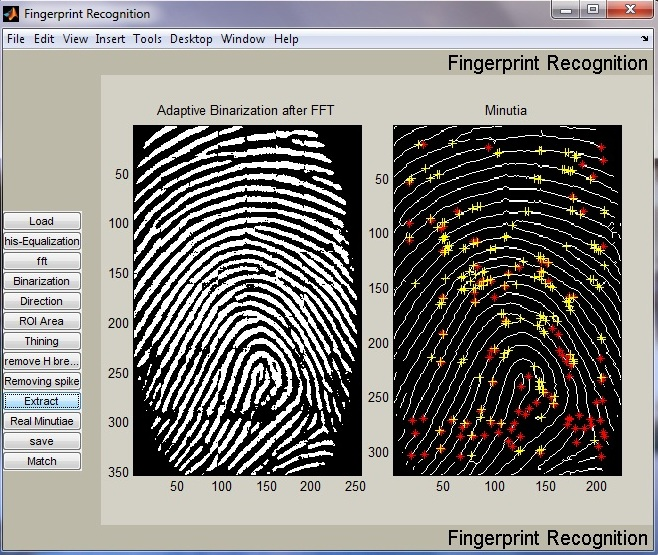


Figure M.10 the Fingerprint image after thinning, H breaks removal, isolated peaks removal and spike removal.(right).

1. Click **Extract** Button



12. Click **Real Minutia** Button

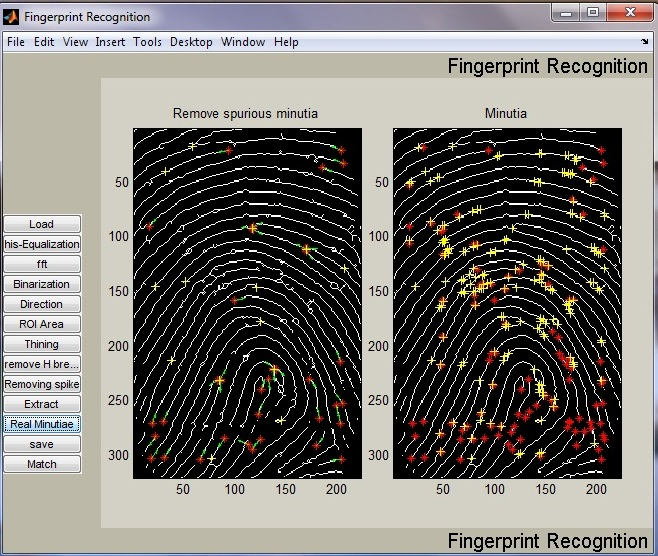


Figure M.12 Minutia Marking (right) and False Minutia Removal (Left). Bifurcations are located with yellow crosses and terminations are denotes with red stars. And the genuine minutia (left) are labeled with orientations with green arrows.

13. Click **Save** Button

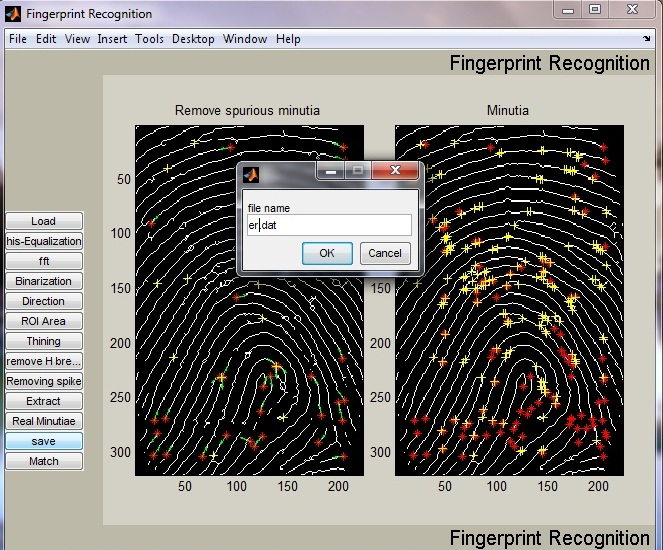


Figure M.13 Save minutia to a text file. The saved text file stores the information on all genuine minutia. The exact format of the files are explained in the source code.

14. Click **Match** Button

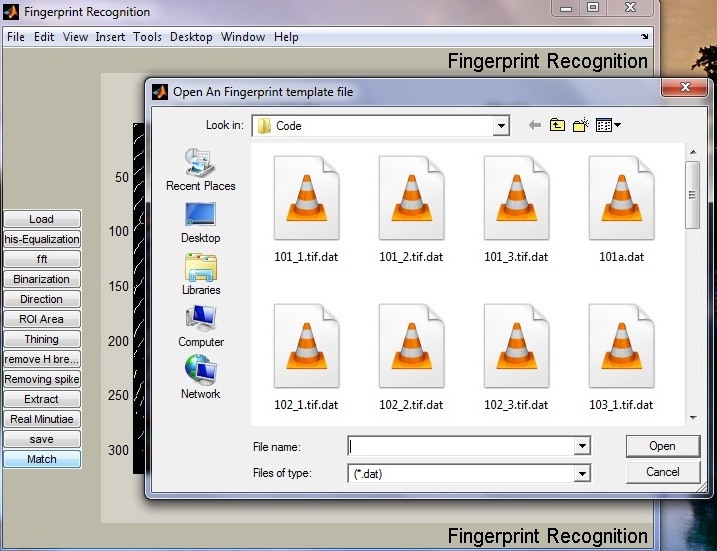


Figure M.14 Load two minutia files and do matching. Users can open two minutia data files from the dialog invoked by clicking the ‘Match’ button. The match algorithm will return a prompt of the match score. But be noted that matching in the GUI mode is not encouraged since the match algorithm relies on heavy computation. Unpredicted states will happen after a long irresponsive running time. Batch testing is prepared for testing match. Please refer the source files for batch testing.

### **Chapter 6: Experiment Details**

**6.1. Experiment Results**

A fingerprint database from the FVC2000 is used to test the experiment performance. My program tests all the images without any fine-tuning for the database. The experiments show my program can differentiate imposturous minutia pairs from genuine minutia pairs in a certain confidence level. Furthermore, good experiment designs can surely improve the accuracy as declared by [10]. Further studies on good designs of training and testing are expected to improve the result.

Here is the diagram for Correct Score and Incorrect Score distribution:

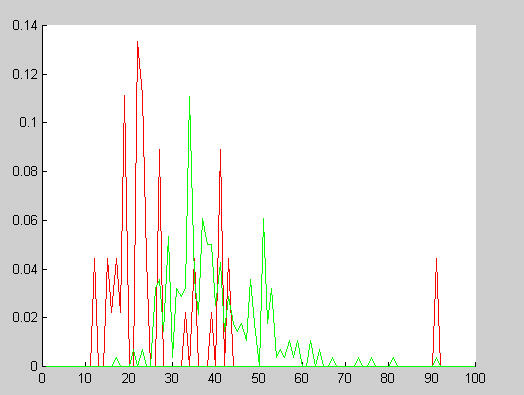


Figure 7.2.1 Distribution of Correct Scores and Incorrect Scores   
Red line: Incorrect Score   
Green line: Correct Scores

It can be seen from the above figure that there exist two partially overlapped distributions. The Red curve whose peaks are mainly located at the left part means the average incorrect match score is 25. The green curve whose peaks are mainly located on the right side of red curve means the average correct match score is 35. This indicates the algorithm is capable of differentiate fingerprints at a good correct rate by setting an appropriate threshold value.

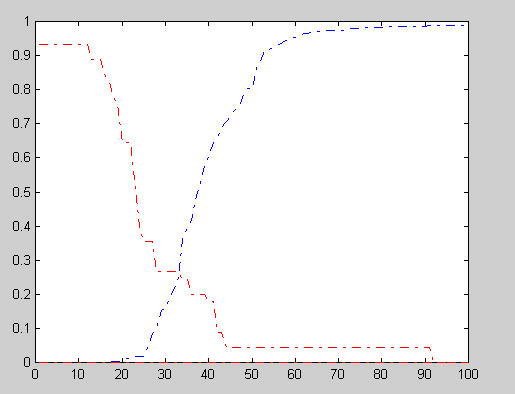


Figure 7.2.2 FAR and FRR curve

Blue dot line: FRR curve   
Red dot line: FAR curve

The above diagram shows the FRR and FAR curves. At the equal error rate 25%, the separating score 33 will falsely reject 25% genuine minutia pairs and falsely accept 25% imposturous minutia pairs and has 75% verification rate.

The high incorrect acceptance and false rejection are due to some fingerprint images with bad quality and the vulnerable minutia match algorithm.

* 1. **SOM Experiment Details:**

*Fingerprint images Format*: TIF.

*Source:* All the fingerprint images are from FVC2000 and FVC2002.

*Database size*: 60.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Search% |  | Recognition% |  |  |  |
|  |  | **3x3** | **4x4** | **5x5** | **8x8** | **10x10** |
| M | 10 | 12.1 | 18.8 | 28.8 | 91.8 | 100 |
| S | 20 | 26.0 | 38.7 | 54.0 | 100 |  |
| O | 30 | 40.1 | 61.3 | 80.5 |  |  |
| M | 40 | 55.9 | 86.6 | 100 |  |  |
|  | 50 | 71.9 | 100 |  |  |  |
|  | 60 | 88.5 |  |  |  |  |
|  | 70 | 100 |  |  |  |  |
|  | 80 |  |  |  |  |  |
|  | 90 |  |  |  |  |  |
|  | 100 |  |  |  |  |  |
|  | 10 | 19.6 | 16.4 | 26.4 | 40.0 | 62.7 |
| S | 20 | 39.6 | 38.1 | 53.1 | 100 | 100 |
| O | 30 | 57.3 | 60.9 | 82.5 |  |  |
| M | 40 | 72.9 | 84.8 | 100 |  |  |
|  | 50 | 86.3 | 100 |  |  |  |
|  | 60 | 97.5 |  |  |  |  |
|  | 70 | 100 |  |  |  |  |
|  | 80 |  |  |  |  |  |
|  | 90 |  |  |  |  |  |
|  | 100 |  |  |  |  |  |
|  |  |  |  |  |  |  |

Each fingerprint has three different rolling. Two rolling of each fingerprint are used as training samples, the other one is used as testing sample.

From the above Training algorithms and experiments, we can also observe that MSOM performs better than SOM for most cases. Also the results improve with the increasing network size.

#### Conclusion:

#### Our project has combined many methods to build a minutia extractor and a minutia matcher. The combination of multiple methods comes from a wide investigation into research paper. Also some novel changes like segmentation using Morphological operations, minutia marking with special considering the triple branch counting, minutia unification by decomposing branch into three terminations, and matching in the unified x-y coordinate system after a two-step transformation are used in this project which enhances the test results. Also a program coding with MATLAB going through all the stages of the fingerprint recognition is built. It is helpful to understand the procedures of fingerprint recognition and demonstrate the key issues of fingerprint recognition.

**Chapter 7: Future Scope**

This project is to design and implement a pattern recognition system by using a graphical user interface (GUI) called NNTOOL. In this project, there is main target to achieve at the end of this project:

1. To design and train the network in learning algorithm and weight initialization.

2. To study about using graphical user interface (GUI) include inside Neural Network

toolbox.

3. To adapt of using the features extracted from the image which compare and produce a

description about the image.

4. To increase the layers of SOM so that there won’t be too many vectors in a single class.

**References**

[1] L. Hong., “*Automatic Personal Identification Using Fingerprints*”. PhD thesis, Michigan State University, 1998.

[2] SOM MATLAB Tool Box. http://www.cis.hut.fi/projects/somtoolbox/

[3] L.C. Jain, U. Halici, I. Hayashi, S.B. Lee and S. Tsutsui, *“Intelligent biometric techniques in fingerprint and face recognition”*. 1999, the CRC Press.

[4] Anil Jain, Sharath Pankanti, “*Fingerprint Classification & Matching”, Exploratory Computer vision Group*, IBM T.J. Watson Research Center.

[5] “*Intelligent Fingerprint Recognition System*” , Sy mohd syathir bin sy ali zainol abiding, University of malaysia Pahang.

[6] “*Liquid State Machine Based Fingerprint Identification*”, Australian Journal of Basic and Applied Sciences, 5(5): 857-865, 2011 ISSN 1991-8178.

[7] L. Hong, Y. Wan , and Anil K. Jain., *”Fingerprint Image Enhancement : Algorithm and performance algorithm”*. IEEE Transactions on Pattern Analysis and

Machine Intelligence, 20(8):777 { 789, May 1998.

[8] Halici U., Ongun G. *“Fingerprint Classification through Self Organization Maps Modified to Treat Uncertainties”*, Proceedings of the IEEE, Vol84, No10, pp1497 1512, October 1996.

[9] C.J. Lee and S.D. Wang, *“Fingerprint feature extraction using Gabor filters".* Electronic Letters, 35(4):288 { 290, 1999.

[10]*“Biometric Authentication System”*, Philippe C. Cattin Swiss Federal Institute of Technology Zurich, 2002.

[11]*“Fingerprint Identification in Biometric Security Systems”,* Mary Lourde R. and Dushyant Khosla, International Journal of Computer and Electrical Engineering, Vol. 2, No. 5, October, 2010 1793-8163.

[12] *“Fingerprint Identification and Verification System by Minutiae Extraction Using Artificial Neural Network”,* Atanu Chatterjee, Shuvankar Mandal, G. M. Atiqur Rahaman, and Abu Shamim Mohammad Arif.

[13] Wikipedia, “Self- organizing Maps (SOM)”,

[www.wikipedia.com/self\_organising\_maps/](http://www.wikipedia.com/self_organising_maps/)

[14] Mary Jane and Aliman Manalo. *“Development of a Fingerprint Classification Scheme For Improved Fingerprint Identification".* Technical report, University of the Philippines, Diliman.

[15] Raymond Thai, *“Fingerprint Image Enhancement and Minutiae Extraction".* Technical report, The University Of Western Australia.

**List of Publications and Achievements**

1. Got first prize in Technical paper presentation in “Perifereria: The Tech Fest” conducted in Pillai’s HOC College of Engineering and Technology on 30th & 31st March.
2. Certificate of Participation in “Techniquest 2012”, A National Level Technical Paper Presentation held in Shah and Anchor Kutchhi Engineering College on 7th April, 2012.
3. Certificate of Participation in “INNOVISION 2012”, A National Level Technical Symposium held in Rajiv Gandhi Institute Of Technology, Mumbai on 26th of March, 2012.
4. Our paper had been shortlisted for next round of “TECHNODOX 12- Technical Paper Presentation at Fervor” in Vidyalankar Institute of Technology.