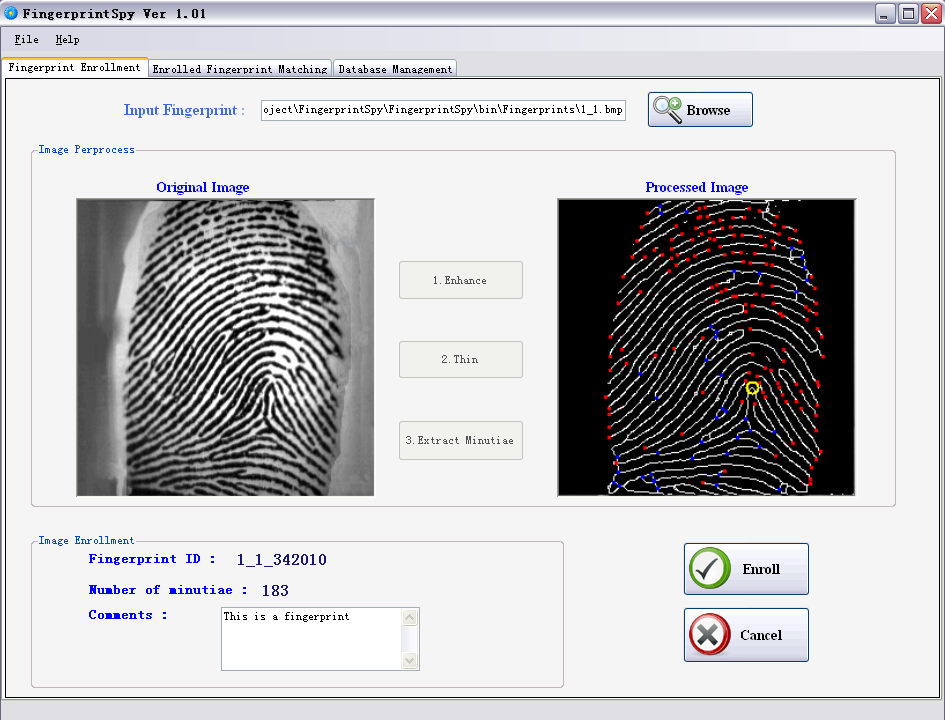


**Project Report**



**Student Name: Wen Liu**

**ID Number: C00105088**

**Project: Fingerprint Recognition**

**Supervisor: Nigel Whyte (M.Sc. MIEEE)**

**Date: 15/04/2010**

**Contents**

1. **Introduction ……………………………………………………………… 2**
2. **Overcome problems …………………………………………………….. 3**

**2.1** Fingerprint scanner **…………………………………………… 3**

2.2 False thinning **…………………………………………… 4**

2.3 Complex math formulas for Core Point detection **…………... 7**

2.4 Processing speed  **……………….…………………..…......... 10**

1. **What I achieved ………………………………………………............... 11**

**3.1 Fingerprint enhancement ……………………………...…… 11**

**3.2 Thinning** **……...……………………………………. 12**

**3.3 Minutiae extraction ..................................................................... 13**

**3.4 Core point detection …………………………………………… 14**

**3.5 Fingerprint enrolment & database management ………………. 15**

**3.6 Fingerprint matching ………………………………............. 16**

**3.7 Fingerprint recognition ...………………………………..…… 17**

3.8 Saving **matching & recognition result ........................................ 18**

**4. What I did not achieved ……………………………………………..…...... 20**

**5. What’s the difference ………………………………………………....... 21**

**5.1 GUI ….…………………………………...………………… 21**

**5.2 Additional functions** **……...……………………………………. 22**

**6. What I learned ………………………………………………....... 23**

**7. What I will do differently if start again ………………………………... 24**

**8. Module description ………………………………………………...... 25**

**9. Data structure ………………………………………………...... 27**

**10. Conclusions …..……………………………………………… 28**

**Bibliography ….………………………………………………. 29**

1. **Introduction**

Fingerprint Recognition is the biggest challenge I have ever met in software project. During the development of this project, problems sometimes encountered, and changes from my earlier research and design reports must be made. I learned a lot from fixing those problems, doing research, designing module and structure, making development schedule and so on.

This document described those problems encountered and how they been solved, what I achieved and did not achieved, what I learned from this project, what I would do differently if starting again, what is the differences from earlier design and research.

It also contains the description of functional modules in the project, data structures and testing used to assess reliability of my project product.

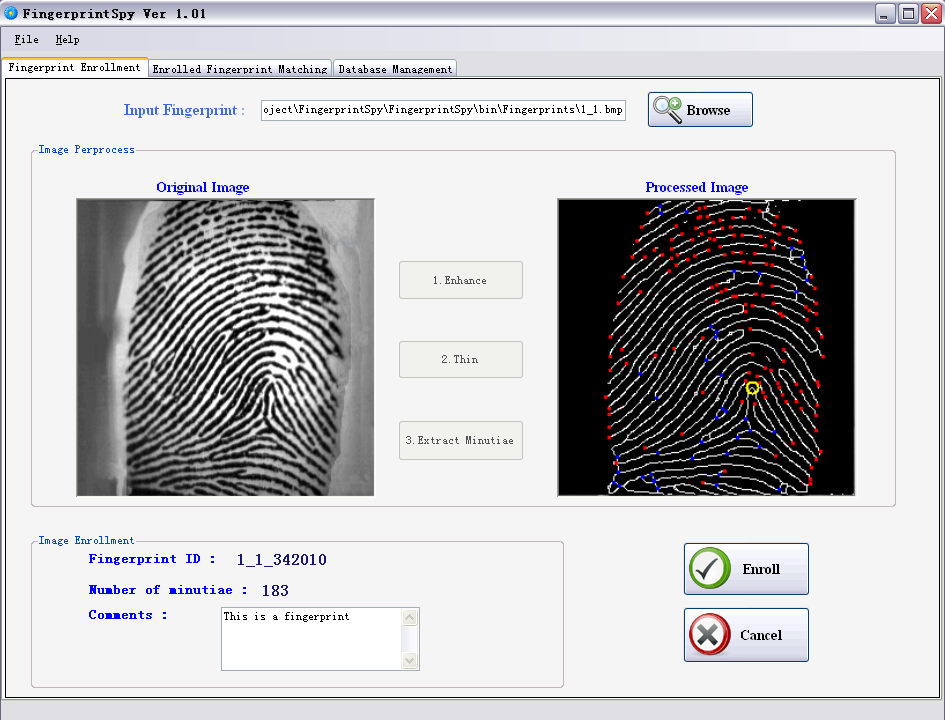
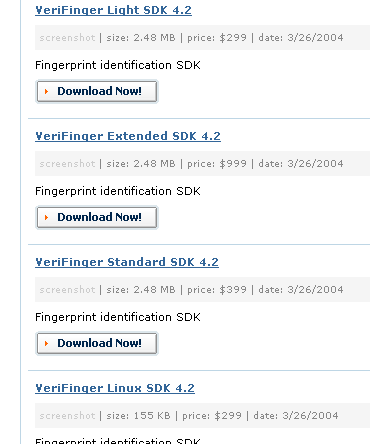


Figure . The GUI of the product – FingerpirntSpy Ver 1.01

1. **Overcome Problems**

***2.1 Fingerprint scanner***

Usually, the fingerprints used to enroll are got from a fingerprint scanner. The recognition application using an interface to connect with the fingerprint and obtains fingerprint image from scanner. Commercial Fingerprint SDKs always contain these interfaces. But one type of interface can only worked on one (sometimes several) particular type of fingerprint scanner.

However, the scanner I got from college does not have an install disk or specification. I spend a month to try to find out the model of that scanner and corresponding software interface. But every SDK cost hundreds of US dollars (see left for a screenshot of a commercial SDK company’s web page) which means I cannot purchase them casually without known the module of the scanner.

On the other hand, even if I purchased the right SDK and get the interface for the scanner, I still cannot use the SDK – the SDK contains all API, functions and class used to develop the fingerprint identification application, which means it cannot be called an “individual project” if the project is full of functions and classes from a commercial SDK.

Figure . The price of SDK is hundreds of US dollars

Fingerprint recognition requires the input fingerprints MUST have the same specification. Fortunately, I found a fingerprint image database from the Internet. There are 24 moderate quality fingerprints in that database. These images are obtained from six different individuals and four different copies of each finger using the same fingerprint scanner. It is means those fingerprint images have the same specification. (size: 256 X 256, bit depth: 24 bits, resolution: 71 dpi)

With this fingerprint database, the fingerprint scanner problem is overcomed and I do not need to keep wasting my valuable time on finding SDK and specification. However, the disadvantage is only 24 fingerprints available, which means the product of my application is reliable when process fingerprints have this specification only. (I tried to find some other fingerprints have the same specification but I did not find any) That is the reason why I list this as a problem I met during development of the product.

***2.2 False Thinning***

At the phase of implementing image preprocessing, the image thinning is the biggest problem which I stuck on it for a while.

Image thinning is the most important step of entire fingerprint recognition project. The quality of thinning is directly affecting the accuracy of minutiae extraction even the final result of the recognition.

In my research report, I mentioned I would like to use Zhang-Suen thinning algorithm as the method to thin the fingerprint. I read the article which publish this algorithm [1] carefully and try to implement it in C# language. I spent one week to get my code done. But for some reason, it does not work correctly.

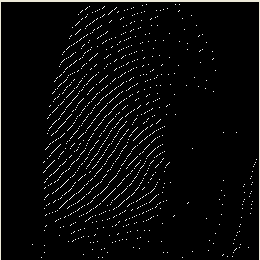


Figure . The original fingerprint image (left) and the output of my false thinning

I double checked the sequence of process and the correctness of the code, but I did not find the reason of this failure.

After searching on the Internet and keep reading the article words by words, I thought I might misunderstood with the sequence of processes and the times of running these processes on the image. I use a picture with black background color and only a capital ‘H’ in white color on that picture to test the code. As expected, the result image is not correct and the error is very straightforward to see.

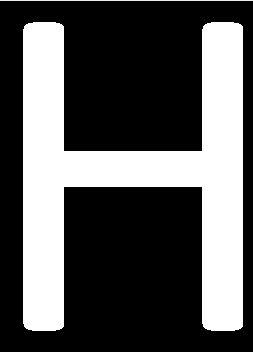
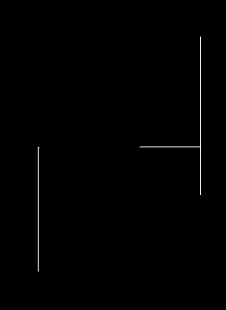
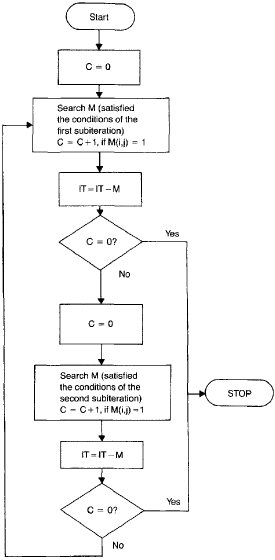
 

Figure . The picture I used to test my thinning algorithm’s failure.

I redesigned my algorithm to meet the Zhang-Suen algorithm’s process sequence.

In the first sub-iteration, only the south-east boundary pixels and the north-west corner pixels which do not belong to an ideal skeleton are marked. Those pixels will be deleted when the first sub-iteration stopped. The second sub-iteration only starts when some pixel been deleted in the first sub-iteration. Similarly, those pixels will be marked and deleted when iteration stopped. If any points been deleted, redo first sub- iteration. Keep doing those two sub- iterations until no more pixels to be deleted.

After redesigning the algorithm, the problem fixed.

Figure The algorithm sequence of Zhang-Suen thinning

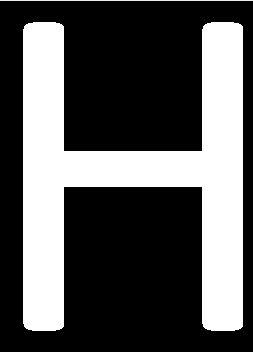
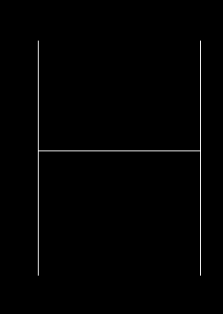
 

Figure Use same picture to test modified algorithm. Problem fixed.

I also tested the fixed thinning algorithm on several fingerprint images. The new algorithm always gave the expected result and I am happy with these results.

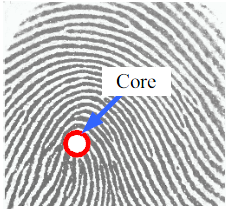


Figure . Two original fingerprints (left upper and left lower corners) and their thinned images after implementing the fixed thinning algorithm on them.

***2.3 Complex math formulas for Core point detection***

As I choose core point base fingerprint matching technique, the core point of a fingerprint is required as the reference point, which is the most inner and highest curved point. [2]

*“The core point, no matter precisely or loosely located, has shown its applications in both fingerprint classification and fingerprint matching using either spatial domain or transformed domain”[3]*



There are four steps for core point detection: image normalization, pixel gradients calculation, orientation field detection and core point detection. Every step has several complex mathematic formulas which I never seen them before.

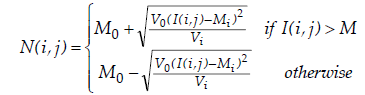
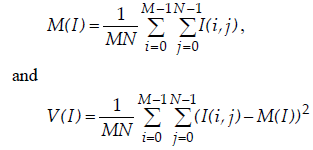
For example, to normalize the unprocessed fingerprint image [3]:

Figure . An example of core point of fingerprint.[2]

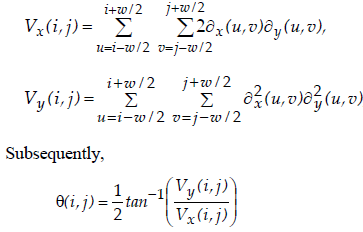
*Here M0 and V0 are the desired mean and variant respectively. The mean and variant of a gray-level fingerprint image with the dimension of M×N pixels, are defined respectively as:*

To detect the orientation field of the fingerprint after gradients of every pixel been calculated [3]:

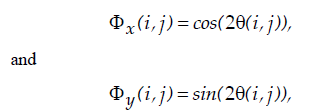
*1) Divide the input image I into non-overlapping blocks with size w×w .*

*2) Compute the gradients ∂x(i, j) and ∂y(i, j) at each pixel (i, j) which is the center of the block. The gradient operator can be chosen according to the computational complexity.*

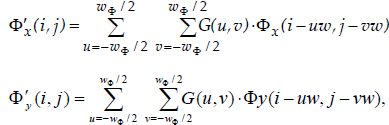
*3) Estimate the local orientation using the following equations*



*4) Assumed that the local ridge orientation varies slowly in a local neighborhood where no core point appears. The discontinuity of ridge and valley due to noise could be softening by applying a low pass filter. However, to apply a low pass filter the orientation image must be converted to a continuous vector field. The continuous vector field, which its x and y components are defined as Φx and Φy respectively.*



*5) With the resulted vector field, the two dimensional low-pass filters G with unit integral is applied. The specified size of the filter is wΦ ×wΦ . As a result,*



*6) The smoothed orientation field (local ridge orientation at (i, j) ) can then be computed as follows:*

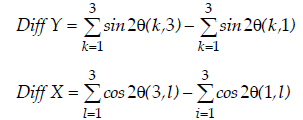


To detect the core point of the fingerprint after orientation field been established [3]:

*1) Compute the local orientation θ(i, j) by using equation (3) above. The input block size could be small as w = 3 , ie, k × l = 3× 3 pixels.*

*2) Smooth the orientation field θ′(i, j) by using equation (6) above.*

*3) In every progressive block, the difference of direction components is computed.*



*4) The curvature point (X) could be located at the corresponding (i, j) where Diff X and Diff Y are negative.*

I was stuck with these mathematic formulas for at least two weeks. I tried many times to implement those formulas in C# but the results are either crashed compiling or false outputs. I realized I could not go any further without figure out the meaning of those formulas. I researched all materials (articles, web pages, replies of the questions I posted on the BBS) I got, but still no progress.

Finally, I contacted with one mathematic lecture of college, Mr. Joseph Bennett. This gentleman gives me great help on understanding those formulas and really helps me to find my confidence back. After several discussions with Joe, the biggest problem I met during the development is solved.

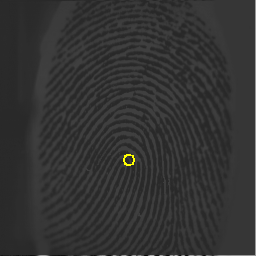
I use several different fingerprints to test the algorithm and the results are always correct.

Figure original fingerprint (left) and its normalized image (right) with core point indicated

***2.4 Processing speed***

After the success of fingerprint image preprocessing (enhancement, thinning, normalization, orientation filed detection and core point extraction), I found although the results are always correct but the processing speed is a little bit slow.

The reason of that problem is my thinning algorithm – I double check the image with Zhang-Suen thinning technique to get the best thinning result. The thinning function will keep checking the image to eliminate pixels which not a part of skeleton of the fingerprint ridge until no more pixels to erase. In most case, the elimination loops will be executed at least twice to get the skeleton of ridges.

The thinning process usually takes 7 seconds – that is the best I can do after several times compaction. To deal with this problem, at the beginning, I tried to use a process bar to show the detail of thinning process. But this method involves multithreading technique and the times of elimination loops are run must known, which means I must redesign the structure of my image preprocessing steps and research into the multithreading technique. It will take me at least one week to do that and I do not have such time.

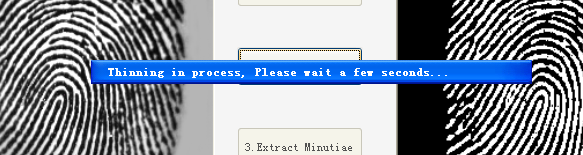
The solution I accepted is using a message bar to show the thinning algorithm is in process. The message bar will be disposed when the thinning process completed.

Figure The message bar shows the thinning function is in process.

1. **What I achieved**

***3.1 Fingerprint enhancement***

As the first step of image preprocessing, the fingerprint enhancement takes me one week to get it done. I did not meet any noticeable difficulty in this step. The purpose of enhancement is to change the input fingerprint image into binary format. This step prunes the fingerprint first and distinguishes fingerprint ridges from valleys and marks ridges as white color and valleys as black, background color.

The only problem is setting the value of threshold – if the gray scale of the pixel is greater than the threshold the pixel will be set to black color, otherwise it will be set to white color. I use 100 as the threshold value after about 20 times tests.

This step also shows the same specification of input fingerprints is required. The threshold value I set is only works on the image with specification is size 256X256, bit depth 24 bits, resolution 71dpi. If the input image is not has that specification, the result may not correct.

I tested my algorithm on every fingerprint in my image database and the results are always correct.

Figure The original image (left) and its enhanced image (right)

***3.2 Thinning***

As I mentioned in the first chapter of this paper, I got some problems in this step. But finally I overcomed these problem and get the correct result I want.

The thinning step works just fine after problem fixed and I even tried some image rather than fingerprint to test the algorithm (e.g. picture of number plate) – of course have the same specification, the result are also correct.





Figure The enhanced fingerprint (left upper) and its thinned image (right upper); the enhanced number plate (left lower) an its thinned image (right lower)

***3.3 Minutiae extraction***

Minutiae extraction step, also it is not simple, but because I did a lot of previous research, it did not bother me much.

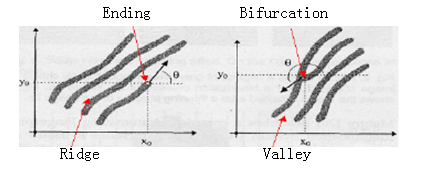
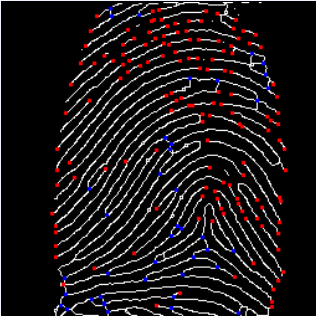
I implemented the algorithm I mentioned in research report, and it works just fine. There are only two kinds of minutiae I care about in fingerprint recognition – ridge ending and ridge bifurcation.

Figure Ridge ending and bifurcation of a fingerprint

The algorithm I used can find out all those minutiae correctly. I marked ridge endings with red dots and bifurcations with blue dots. In fact, I do not need such number of minutiae for the matching and recognition. I implement false minutiae (as I mentioned in research report, they are spike, bridge, hole, break, spur, ladder structures) elimination algorithm and I get the results which I expected.



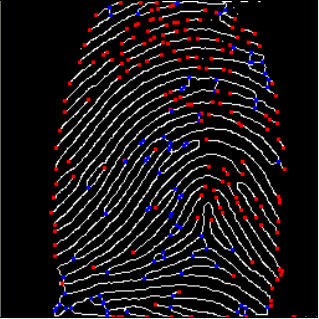


Figure Thinned fingerprint with all minutiae indicated (left) and the image with false minutiae eliminated.

I also use the number plate image to test the minutiae extraction algorithm and I got the correct result as I expected.



Figure Thinned number plate image (left) and the image with minutiae indicated.

***3.4 Core point detection***

As I mentioned in the Overcome problems chapter, I met some difficulties when I implement the formulas into codes. After overcome these problem, I found the core point algorithm works perfect for my program. I used more than 20 different fingerprints to test the algorithm and I am happy with the results.

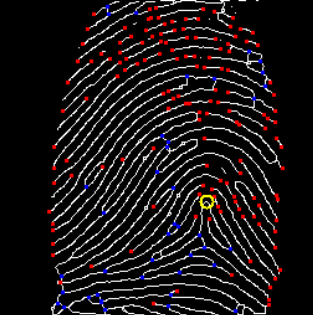
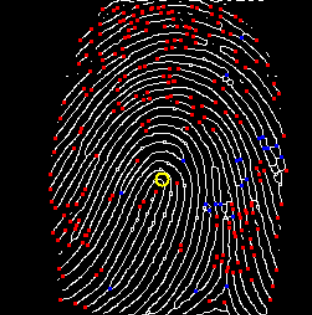
These fingerprint images above are ready to enroll into databases.

Figure Three fingerpint images with minutiae and core pint (yellow circle) are indicated.

***3.5 Fingerprint enrolment & database management***

In my design manual, I mentioned that there are two databases in my application – image database and details database. There is no big difference in my product.

The image database is a hidden folder under the application installation path. It contains all enrolled (processed) fingerprint images.

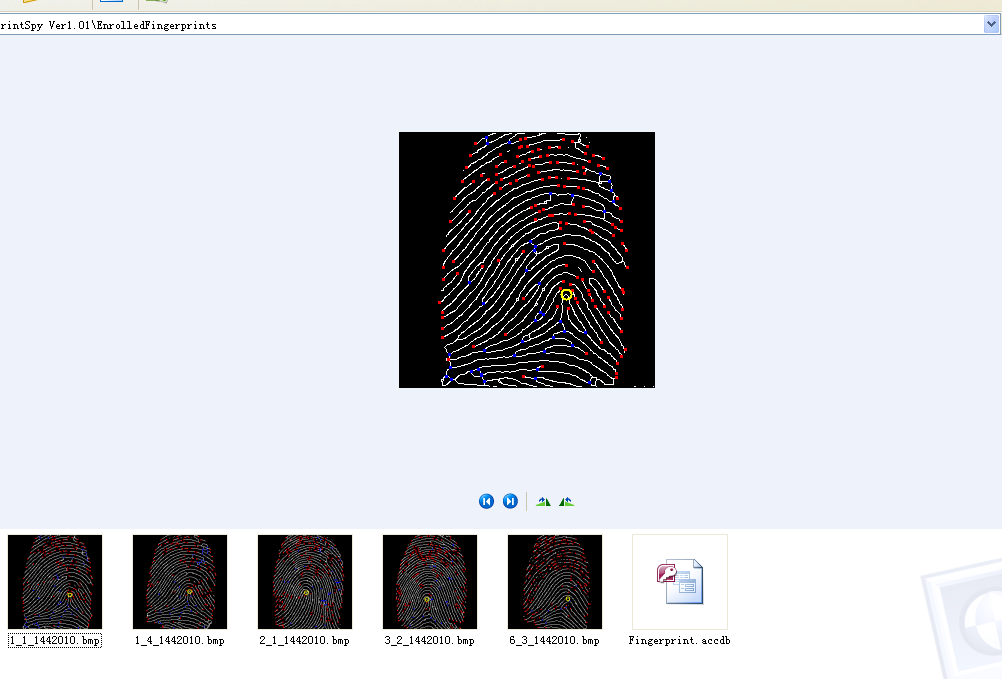
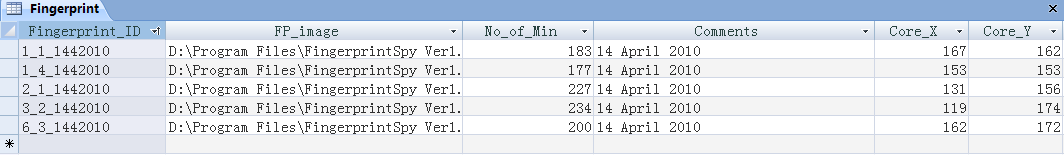
The details database is an Access database which contains two tables – fingerprint table and minutiae table and protected by password (Password is: 081021). Fingerprint table contains details of the enrolled fingerprints.

Figure Image database of the application

Figure Fingerprint table of details database

Minutiae table contains details of the minutiae of each enrolled fingerprint. it is sorted by the Fingerprint\_ID.

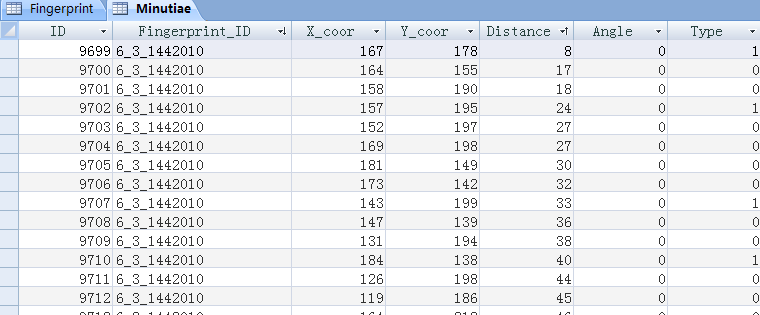


Figure Minutiae table of the details database

I successfully connect databases with my application and user can insert, read and delete records from the databases. Because of success of implementing minutiae extraction, core point detection and database operation, the fingerprint matching and recognition can be done without difficulties.

***3.6 Fingerprint matching***

The good and reliable results of previous stages make fingerprint matching a straightforward step to do.

After user selected two enrolled fingerprint from the database, the program set up a connection with database to get all details about these fingerprints from Fingerprint table and Minutiae table. The minutiae details will be assigned into two 2-dimentional arrays – template array and sample array. Because of the distances from core point to the minutiae and rotation angle differences between minutiae and core point been calculated already before fingerprint enrollment, the matching algorithm is only simple mathematics.

The algorithm takes every minutia record from template array and matches them with records in sample array. If two minutiae (one from template and the other from sample) are the same type (ending or bifurcation) with distance difference less than 6 pixels and angle difference less than 6 degrees, they will be marked as a pair of matched minutiae. The algorithm keep doing this until all minutiae records in template array been processed.

The algorithm finished with returning a string array which contains number of template minutiae, matched pairs and similarity rate. This information will be showed on the GUI with result comments which based on the similarity.

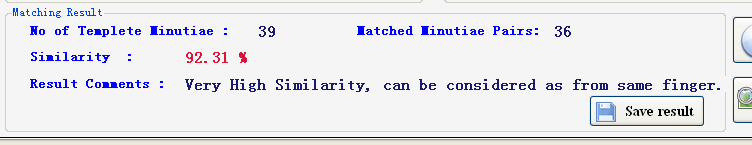


Figure Matching result be showed on the GUI with a result comment

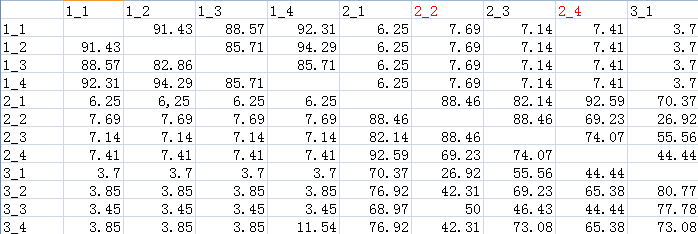
I use about 100 different pairs of fingerprints to test the algorithm and the accuracy of the matching is more than 90%. I am very happy with this result.

Figure The similarity table of fingerprints I used to test the matching algorithm. The first digit of image name shows whether the fingerprint image is come from same finger; the second digit is the copy no. of the finger, which means the images with same first digit are got from same finger.

***3.7 Fingerprint recognition***

This function is not mentioned in my previous report. My supervisor, Mr. Nigel Whyte suggested I should add this function into my product. I found it is a very useful function which saves user a lot of time to find out the most similar fingerprint record of an unenrolled fingerprint.

The algorithm behind this function is very similar with fingerprint matching. The only different is the algorithm will run image preprocessing, minutiae extraction and core point detection steps on the input (unprocessed) fingerprint image first, records the minutiae and core point details of the image and then keep reading the fingerprints and their minutiae records from the database and matching them with the input fingerprint’s corresponding record until all records in the fingerprint database be processed or a record’s similarity is greater than 95.0%. The algorithm finished with retuning a result string array which contains the similarity rate of the most similar fingerprint record of the input image and the details of that matched fingerprint. This information will be showed on the GUI as well.

I also run many tests on the recognition algorithm and the results are satisfactory.

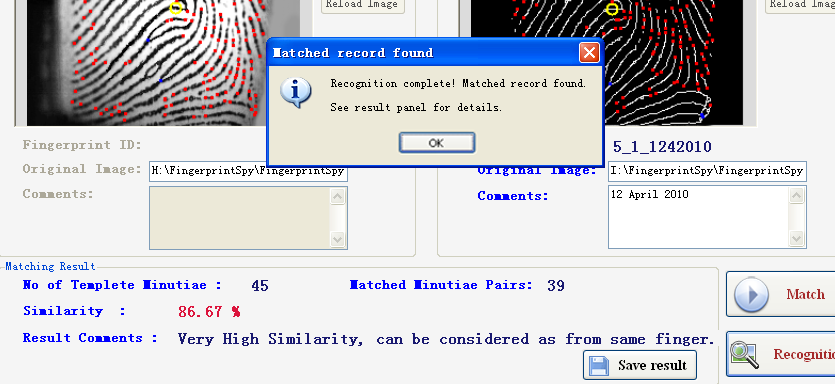


Figure The screenshot of the GUI when recognition finished.

***3.8 Saving matching & recognition result***

The application also has a function to save the matching & recognition result. This is another function I did not mention in the previous reports.

User click  button and application will then generate a text file (\*.txt) which contains details of the match. This text file will be saved under the “Match Results” folder in the installation path of the application.

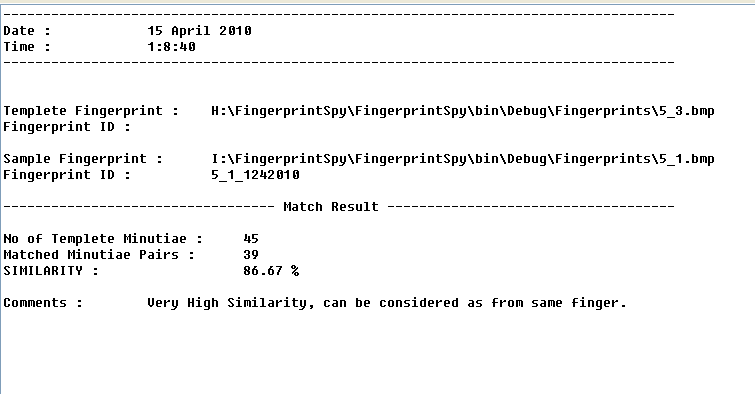


Figure A screenshot of the match result save file generated by the application.

This function is simple to do but it is useful. It allows user to review the result of fingerprint matching they did previously. That is the reason why I add this function into the application at the end of development.

1. **What I did not achieved**

I would like to say I achieved all essential functions required by the project proposal and I also achieved functions which suggested by my supervisor. But there are still some things I am not very satisfy with.

Although the connecting with a fingerprint scanner is not one of the targets of the project and I found a substituting solution, I also list it as a function I did not achieved. If the fingerprint scanner is available, I am sure that I can make my application even more functional and powerful.

The processing speed is another point I am not very satisfy with. As I mentioned in the previous chapter, the application takes 7 - 10 second to thin the fingerprint image. I think possibly there is another algorithm can do the thinning process faster. But Zhang-Suen thinning algorithm is the one gives me the best thinning result.

The last thing I would like to mention is the matching algorithm used in my application. The matching algorithm I used is a core point based matching scheme. It is easy to understand and make logical sense, but it is not a common matching algorithm used in commercial fingerprint recognition systems. In the research stage, I read a lot of articles about fingerprint recognition algorithms. The algorithm gives highest accuracy result and fastest matching speed is called Aligned minutia matching algorithm [4]. This algorithm will align the minutia based on the ridge and then using a size-flexible window to search the matching pairs on the other fingerprint [4]. It is very powerful but it is to complex and time consuming that cannot be done by one person. The algorithm I used gives me good result and an acceptable accuracy, but I can imagine that this algorithm is not very reliable when dealing with poor quality or fragmentary fingerprints.

1. **What’s the difference**

As I mentioned in the previous chapters, some new useful functions been added into my product (e.g. Fingerprint recognition). I also did some change on GUI of the application.

In fact, because I did the research adequately in research stage (read more than 55 related articles, PPTs, and journals), there is no big difference between final product and previous design, specification reports. In this chapter, I would like to introduce these differences I made.

***5.1 GUI***

At the beginning, I set 4 picture boxes to show the output of every process step in Fingerprint Enrollment stage. But latterly I found this is not an efficient way – 4 picture boxes make every processed image small and make differences less distinct. I redesigned the GUI and now it has only one processed image box. The image in the box upgraded after every process step to show the output image – it is bigger and easier to see the differences.

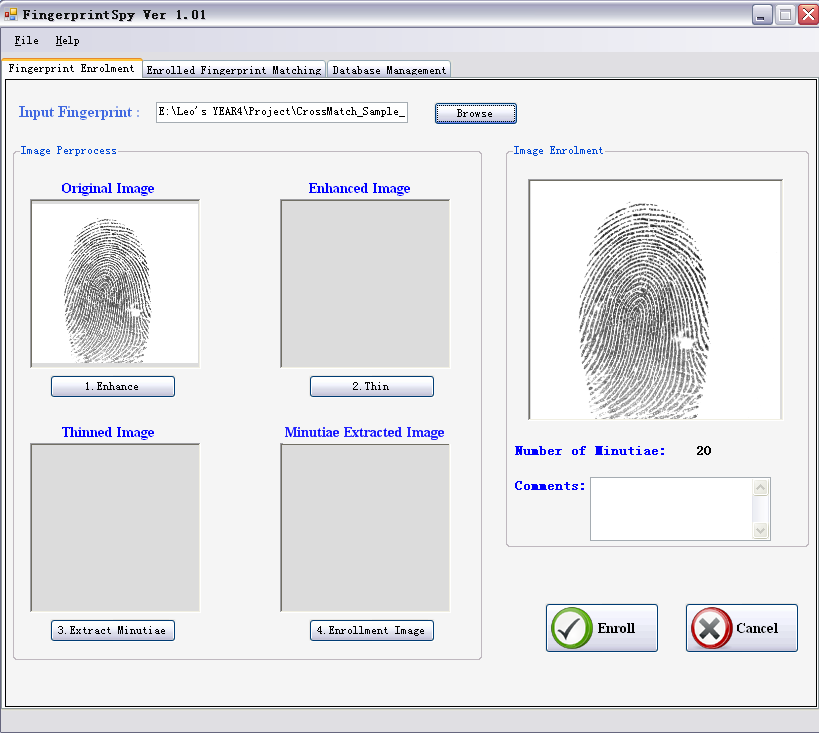


Figure The GUI in my Function specification report (left) and the one of the final product (right)

The Fingerprint matching stage is also different. Because of the additional functions, I added 2 new buttons, and onto the GUI and removed “Print Result” function and its button – there is no more reason to put it on the GUI anymore since a save function been added. The new GUI is more compact, friendly and straightforward.

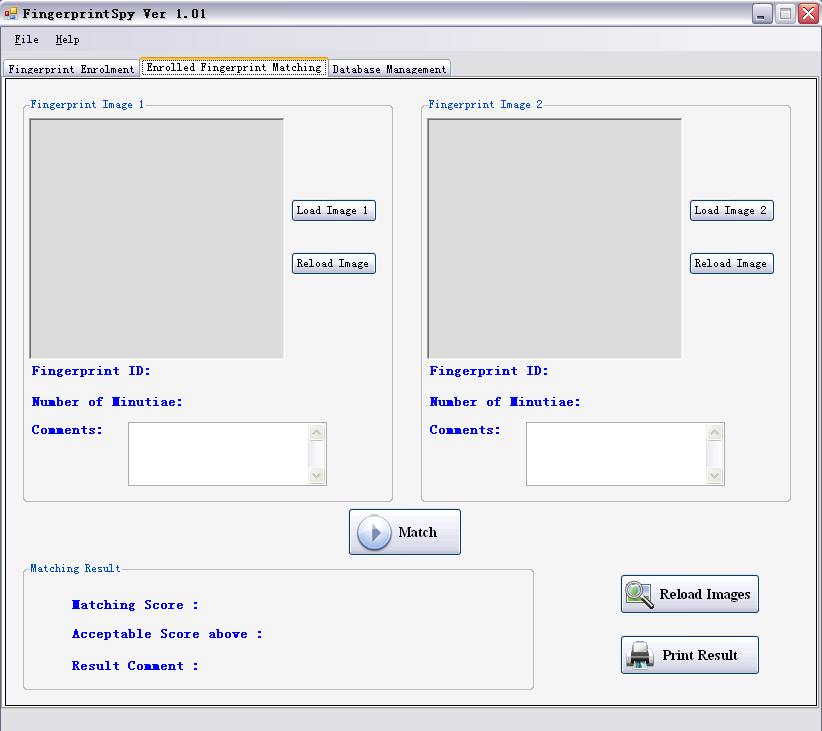
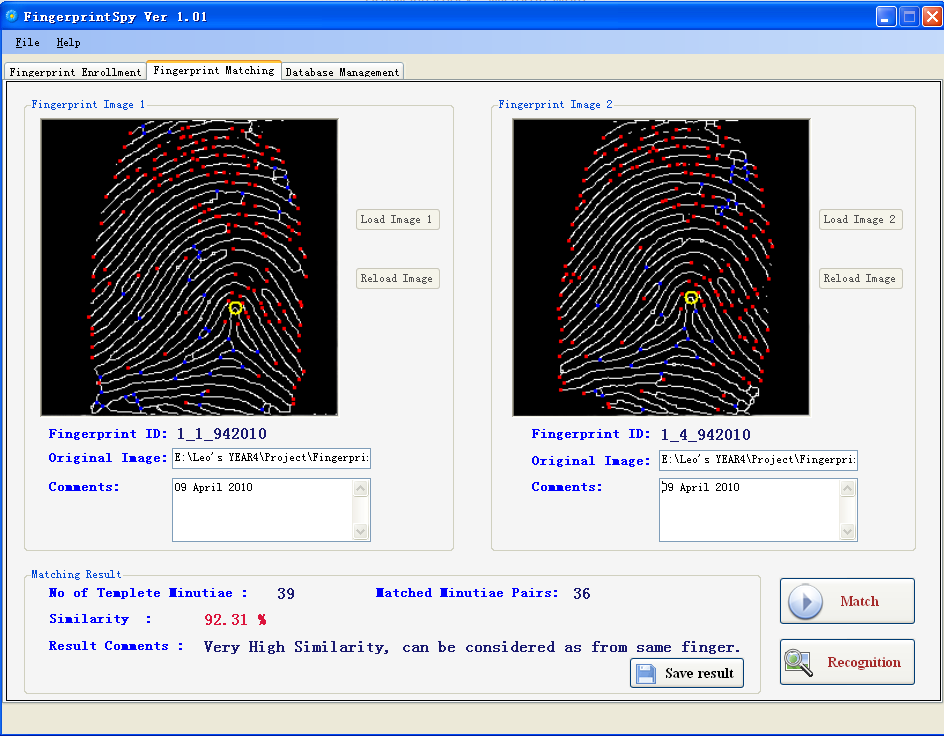


Figure The GUI in my Function specification report (left) and the one of the final product (right)

***5.2 Additional functions***

I mentioned about the fingerprint recognition function and save matching & recognition result function been added into final product in the previous chapter. I will not do any redundant description in this section. Details about these two functions are in the chapter 3, called “3.7 Fingerprint recognition” and “3.8 Save matching & recognition result”.

1. **What I learned**

Firstly, I learned the importance of research, both high level and low level for developing a project and how to doing research about an area which I am not familiar with. I would like to say the research stage is extremely important and I cannot achieve the result I got at this stage without doing two months detailed researches.

Secondly, I am more experienced on Object Oriented Programming by doing this project. The essential concepts of OOP are used in everywhere of my project. Classes can communicate with each other and those functions inside the classes can also be refers to functions in another class. It saves me a lot time and makes the application more efficient and faster.

Thirdly, I got experiences on developing program under C# environment. I never touched C# language before this project. The reason of why I pick C# as the develop language is the C# intended to be a simple, modern, general-purpose, Object-Oriented Programming language. The syntax of C# is very similar to Java which another OOP language I used. C# supports GDI+ will make image process easier to program. In C#, database connection is also easy and straight forward. C# has a well designed library. All classes and their usage and explanation can be found in it. With Microsoft Visual C# IDE, GUI programming and design will also easier to be done.

A tons of knowledge on the image processing and biometric technique is another gain I got after doing this project. I believe most of this knowledge will do great helps in the future when I developing projects relevant to image processing and identity verification.

This tough project topic also taught me: Be strong and never give up. Sometimes I got upset when I encountered some extremely unreadable formulas or algorithms. But I found no matter how difficult, you will find the solutions if you keep searching for the answer, not giving up, roll over and die. I very enjoyed the fulfillment after overcome a tough problem.

In one word, this is a great experience which I have learned so such from and I will never forget it.

1. **What I will do differently if start again**

The time constraint is a restriction of the project development. If I have a chance to redo the project from the beginning:

1. I will try to contact with the fingerprint scanner retailer/ manufacturer to get technique supports on connecting fingerprint scanner with application.
2. I will try to finish my research/design finish as soon as possible with a high level quality.
3. I will try to start coding immediately after get the concept of fingerprint technique – get my hands dirty as early as possible.
4. I will try to get more complex and accuracy fingerprint matching technique implemented.
5. I will try to finish coding earlier and leave more time for application testing and debugging.
6. I will keep recording problems I met in the development in details and keep updating the development diary.
7. I will make the application more intelligent, for example run image preprocessing steps automatically when user input a fingerprint.
8. I will make the user interface more friendly, for example add processing bars on the GUI to show exactly time left for thinning process.

Along the development of the project, I found there are several additional functions can be added onto my application. However, time condition would allow me to get all those functions done. I have confidence to implement those functions if I got some more time – I can make my product a perfect fingerprint recognition system.

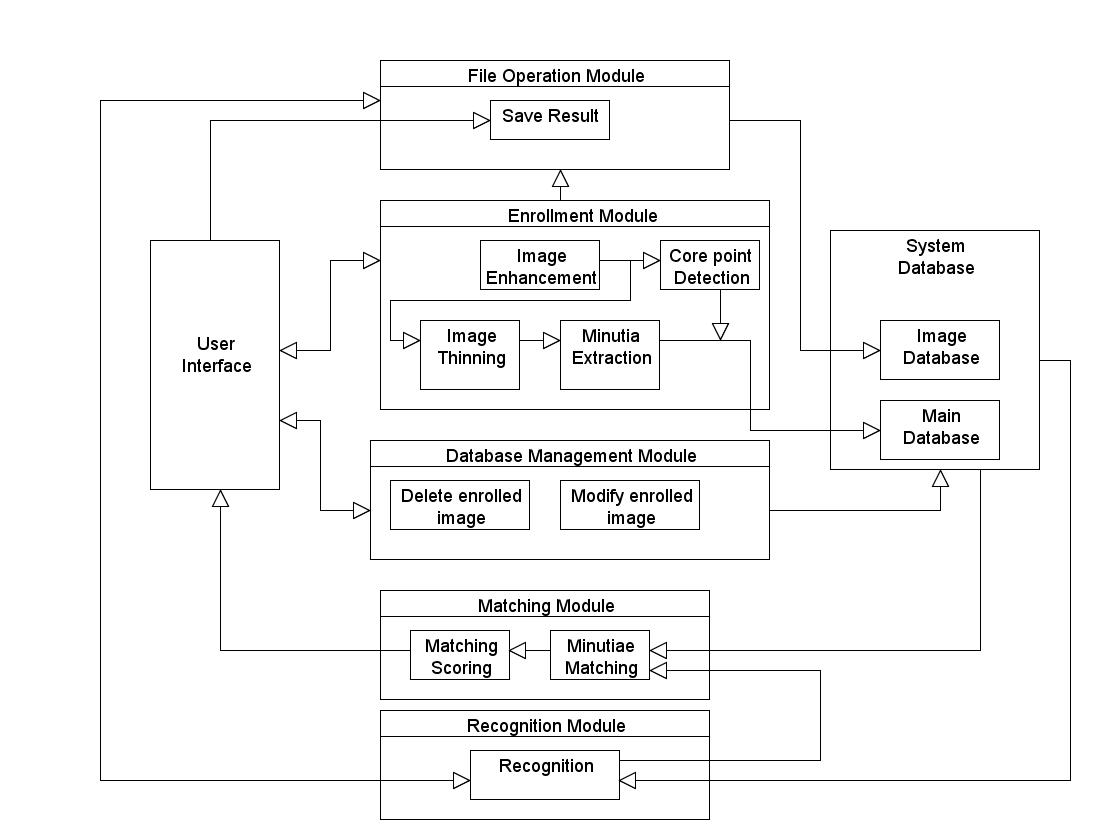
1. **Module description**

Figure The module diagram of the system.

There are five main modules in the system. Enrollment module, Database management module, Matching module, Recognition module and File operation module.

* **Enrollment Module**

The module contains four classes: Enhance, Thinning, Corepoint and Minutiae. This module runs image processing steps and detects the core point and all minutiae of the input fingerprint. The result images will be returned and displayed on the user interface. The result will also be saved into image and main database.

* **Database Management Module**

This module handles database management actions from user interface. Two functions – Delete record function will remove related records of selected fingerprint from image database and main database. Modify comments function will update main database with new comments of the selected fingerprint.

* **Matching Module**

The matching module contains a class called matching used to run matching algorithms on the two enrolled fingerprint images. The matching score will be generated after matching completed and it will be returned to the user interface.

* **Recognition Module**

This module has a recognition class which will process the inputted unenrolled fingerprint image first and then assign details of two fingerprints (the other is enrolled fingerprint got from database) into two matching arrays, send these arrays to matching module to get matching score. The functions keep doing that until the highest matching score been found. The score will then be returned to user interface.

* **File Operation Module**

The module handle file selecting actions by generates open file dialogs. It also generated matching result save files and save these file into “Matching Result” folder under the installation path of the application.

Please look into Code Listings document for details explanation about those modules. Every classes are well commented in that document.

1. **Data structure**

There are two databases been used in the system: main database and image database. As I mentioned in the previous document, the image database is a simple hidden folder which holds all processed image of enrolled fingerprints. The main database is an Access database file which contains two tables: Fingerprint table and Minutiae table

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Data type** | **Sample Input** | **Comments** |
| Fingerprint\_ID | string | “1\_1\_1642010” | The enrolled ID generated by the system. |
| FP\_image | string | “C:\image\1\_1.bmp” | The filename of original fingerprint image |
| No\_of\_Min | int | 201 | Amount of minutiae in the fingeprint |
| Comments | string | “John’s fingerprint” | User comments of the fingerprint |
| Core\_X | int | 145 | X coordinate of the core point |
| Core\_Y | int | 260 | Y coordinate of the core point |

( Table 1. Data structure of Fingerprint table)

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Data type** | **Sample Input** | **Comments** |
| ID | int | 1002 | The id of the minutiae |
| Fingerprint\_ID | string | “1\_1\_1642010” | The id of fingerprint which the minutia belongs to |
| X\_coor | int | 201 | The row index of pixel where the minutiae at. |
| Y\_coor | int | 87 | The column index of pixel where the minutiae at. |
| Angle | int | 45 | The rotation degree of the minutiae |
| Distance | int | 12 | The distance to the core point |
| Type | Int | 1 | The type of the minutia. 0 for a ridge ending and 1 for a bifurcation. |

( Table 2. Data structure of Minutiae table)

The system also has a data structure for the save file (holds matching result) which is very straightforward. The save file of the system is a text format file (\*.txt). I have described the save file in “3.8 Saving matching & recognition result” section in chapter 3 of this document.

1. **Conclusion**

01 : 06 AM, 16th April, 2010. The biggest challenge I ever met in my programming life is finished. As I said, I have learned a lot from doing this project – it is not only a excellent practice of the software engineering skills I have learned in the last four years but also I life experience taught me how to dealing with adversity and pressure.

Although my product application is not a perfect solution of this project topic, but I am still happy with the achievement I got. I spend almost all my after- class time on this project and I feel great satisfaction after complete all required tasks. The experience I got from this project is priceless to me.

I really appreciate the helps and suggestions from my supervisor Mr. Nigel Whyte and Mr. Joseph Bennett. I cannot achieve this step without those valuable helps. Thank you very much.

***Bibliography***

[1] T.Y. Zhang and C.Y. Suen, *A Fast Parallel Algorithm for Thinning Digital Patterns,* Communication of the ACM, Vol.27 No.3. pp 236, Mar 1984.

[2] Tomohiko Ohtsuka, Daisuke Watanabe, Hiroyuki Aoki, *Fingerprint Core and Delta Detection by Candidate Analysis*, MVA2007 IAPR Conference on Machine Vision Applications, May 16-18, 2007, Tokyo, JAPAN

[3] Choomchuay, Atipat Julasayvake and Somsak. 2007. *AN ALGORITHM FOR FINGERPRINT CORE POINT DETECTION*. s.l. : IEEE, 2007.

[4] Luo Xiping, Tian Jie, *A Minutia Matching algorithm in Fingerprint Verification.* AILAB, Institute of Automation, The Chinese Academy of Sciences, Beijing,100080