Large Scale Fingerprint Identification Systems

A Project Report
Submitted in partial fulfilment of the
requirements for the Degree of
Master of Engineering
in
Computer Science and Engineering

by

Kushal Wadhwani

Computer Science and Automation
Indian Institute of Science
BANGALORE – 560 012

JUNE 2011

TO

Family

and

Friends

Acknowledgements

I express my gratitude and sincere thanks to Prof. Jayant Haritsa for his guidance, supervision and feedback for the progress of this project. I would also like to thank all my lab mates for constantly supporting and motivating all through out the course of this project.

Abstract

The Unique Identification(UID) project is introduced, with the mandate of providing a unique identity to all Indian residents [1]. In our work we have designed a system that simulates the working of UID project and derived results that can be scaled up to the magnitude of UID project.

We have implemented a two stage procedure for fingerprint identification. First stage involves continuous classification of fingerprint database using global features which reduces the search space by orders of magnitude. In the second stage exhaustive fingerprint matching is applied using local features on the reduced set of fingerprints.

For continuous classification, directional field feature which captures orientation of ridges in fingerprint image and fingercode feature which captures texture information of fingerprint image are used. To make features noise robust multi space KL transform is applied on directional field and fingerprint features. To further reduce the search space both directional field and fingercode are combined to produce final set for matching. Minutia points are used as local features for matching and minutia match algorithm is used for matching. We have discussed the performance of our classification and matching on time and error matrices over different noise levels. Whole system is built on PostgreSQL open source database system.

Contents

List of Tables

List of Figures

Keywords

Biometric , Fingerprints, Identification System , Continuous Classification , Minutia Matching, Multi-space KL Transform.

Chapter 1

Introduction

1.1 Biometric

Biometrics consists of methods for uniquely recognizing humans based upon one or more intrinsic physical or behavioral traits. Identification of humans on the basis of biometric data is famous and well studied problem in computer science. Fingerprints are the most reliable and widely used biometric data used for identification. Fingerprints attracted many researchers and lot of work has been done in this field.

1.2 UID project

UID project which is introduced with the mandate of providing a unique identity to all Indian residents uses fingerprints as biometric data for ensuring uniqueness. UID project stands out from other identification solutions because of its prodigiousnessestimated database size for UID project an order of magnitude larger than the existing largest biometric database that is IAFIS [18]. Two major functions involved in the functioning of UID are as follows.

De-duplication De-duplication is the processing to restrict multiple instances of same fingerprint in the database. De-duplication, involves matching fingerprint during the time of insertion with all the fingerprints already existing in database.

Verification Verification means to verify that a person really is the person who he or she claims to be.

1.3 Fingerprints

Fingerprints matching is required for both de-duplication and verification. De-duplication involves 1:N and verification involves 1:1 matching. Fingerprint matching is been studied topic over last three decades and there are many fingerprints matching algorithms available in the literature [2]. But all the matching algorithms have been studied for small databases. Also the fastest matching algorithm takes tens of milliseconds to match one pair of fingerprints which is very poor when we consider database like UID. Identification process can be speeded up by reducing the number of comparisons that are required to be performed. Sometimes, information about race, age, sex and other data related to the individual are available and the portion of the database to be searched can be significantly reduced, this can be done in UID however we are not using any such reduction instead we are using information intrinsic to fingerprints such as global features of fingerprints. And use local features for final exhaustive matching. Ridges, valleys, orientation and singularity points are some global features and minutia points are the local features present in fingerprint as shown in figure 1.1 on a sample fingerprint.

Singularity Points Ridges of fingerprint images often run smoothly in parallel but exhibit one or more regions where they show distinctive shapes such as high curvature known as core point and frequent ridge terminations known as delta point.

Minutia points The minutiae points are the details of the ridge-valley structures namely terminals and bifurcations.

1.4 Organization of this thesis

The rest of this report is organized as follows: Section 2 has outlines of the related work. In section 3 working of whole implemented system is explained. In Section 4 we

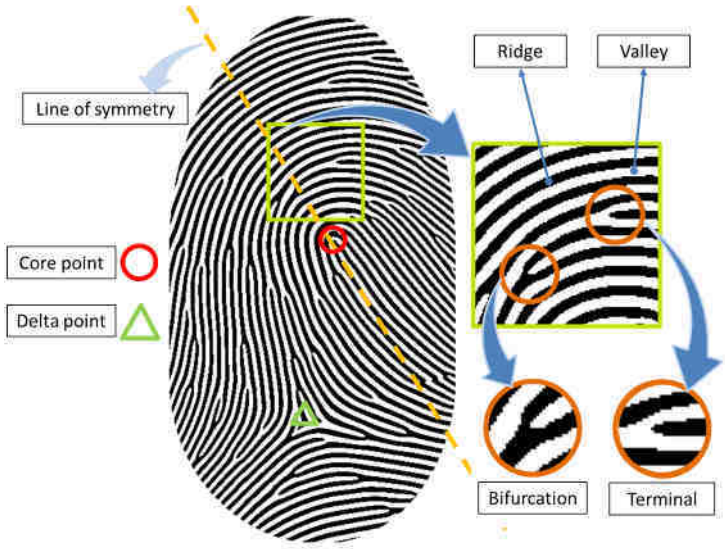


Figure 1.1: Finger prints image indicating its local and global features

have summarized feature extraction procedure and accuracy of each feature. Section 5 contains description and results of various continuous classification techniques we have tried. Section 6 explains minutia matching algorithm and its results on different noise levels detail. Section 6 concludes the work and give some directions about how this work can be extended for bigger database and future work.

Chapter 2

Background and related work

2.1 Classification

Fingerprint classification refers to the problem of assigning a fingerprint to a class in a consistent and reliable way. [2]

2.1.1 Exclusive Classification

Fingerprints in the database are classified into different classes and they are stored in partial databases per class. One of the well known exclusive classification known as henry classification is shown in Figure 2.1. In Henry classification the classes are divided on the basis of singularity points. The query fingerprint is also classified, and is only matched to the fingerprints in that part of the database that contains fingerprints of the corresponding class of query fingerprint. This will in effect decrease search space from whole database to only a single class. Performance of exclusive classification strongly depends on the number of classes and on the distribution of fingerprints. Distribution of fingerprints according to henry classification is shown in Figure 2.2. Unfortunately, the number of classes is often small, the fingerprints are non-uniformly distributed. In the most famous classification schemes approximately 90 % of fingerprints belong to only three classes and there are many "ambiguous" fingerprints whose exclusive membership cannot be reliably stated even by human expert.

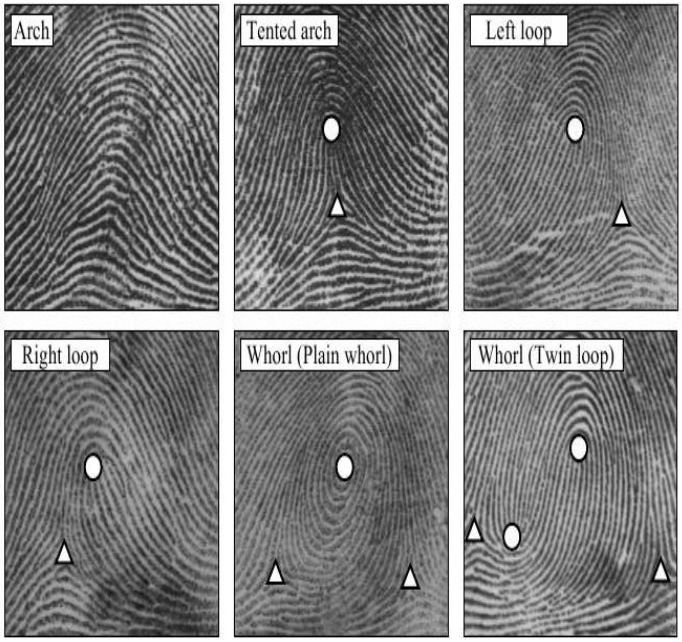


Figure 2.1: Henry classification of fingerprint images based on position of singularity points

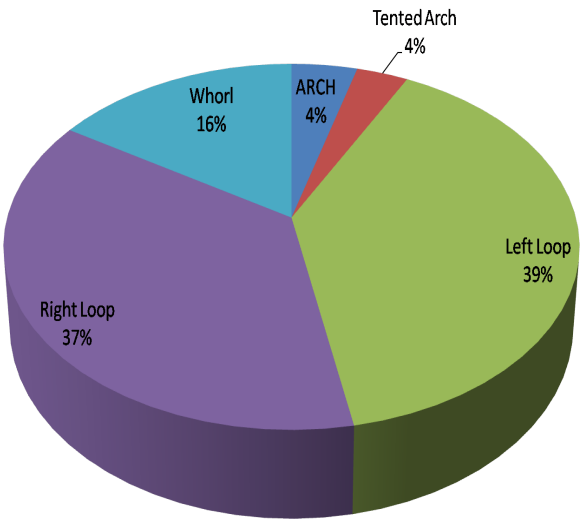


Figure 2.2: Distribution according to henry classification

2.1.2 Continuous Classification

In continuous classification [3], fingerprints are not partitioned into non overlapping classes, but each fingerprint is classified uniquely and independently summarizing its main features. Each finger print is and so can be represented as data-point in spatial feature space. These feature vectors are created through a similarity-preserving transformation, so that similar fingerprints are mapped into close points in the multidimensional space. The continuous features obtained are used for indexing fingerprints through spatial data structures and for retrieving fingerprints by means of spatial queries. By means of special queries all the points in the neighborhood of given radius around query point are retrieved, this set of retrieved points is called nearest neighbor set. Continuous classification approach also allows the problem of exclusive membership of ambiguous fingerprints to be avoided and the system efficiency and accuracy to be balanced by adjusting the size of the nearest neighborhood set.

2.2 Retrieval Strategy

Choosing an indexing technique alone is usually not sufficient: a retrieval strategy should also be defined according to the application requirements such as the desired accuracy and efficiency.
In continuous classification technique following retrieval strategies can be used:

Fixed radius: given a prefixed tolerance ρ, the fingerprints considered are those whose corresponding vectors are inside the hypersphere with radius centered at the point associated with the input fingerprint, the search may be halted as soon as a match is found, or when the whole portion of the database enclosed by the hypersphere has been explored.

Incremental search: fingerprints are visited according to the distance between their associated vectors and the input point, the search continues until a match is found and

in the worst case, it is extended to the whole database.

2.3 Combination Of Features

It is concluded in [4] that continuous classification based on combination of multiple features, are able to search a database more effectively both in terms of time and accuracy. There are two major ways for combining:

Implementing combination In the first approach the classification are fused in some fashion during the classification phase. In the second approach the classification of one classifier is selected according to some criterion. Fusing methods aim at providing the classification by combining the outputs of several classifiers. The base member's classification are combined using weights that are assigned to each member. The member's weight indicates its effect on the final classification.

In second method the premise is that there is a competent authority that nominates the best classifier for a given instance. The output of the selected classifier is referred to as the output of the ensemble as a whole. Very often the input space is partitioned into K competence sub-spaces which can have any shape or size. Then for each sub-space we nominate one classifier to be the predictor.

2.4 Matching

Fingerprint matching algorithm compares two given fingerprints and returns degree of similarity. Matching fingerprint images is a very difficult problem, mainly due to the large variability in different impressions of the same finger due to displacement, distortion, rotation, pressure and skin conditions etc. In literature, very few matching algorithms operate directly on gray scale fingerprint images, most of them require an intermediate fingerprint representation. Minutiae based matching [12] is the most well-known and most widely used method for fingerprint matching, where minutia's are the intermediate representation of fingerprint. Minutiae refer to the bifurcation or termination points of

ridges on the finger surface. They are mainly utilized in fingerprint matching since their distribution on the fingerprint provides a unique signature for an individual.

Performance matrices for matching The accuracy of matching algorithm is defined by two errors: FAR (false acceptance rate) and FRR (false rejection rate).

FAR

It measures the probability that the fingerprint of an impostor is accepted by the system.

FRR

It is the probability that the fingerprint of an authorized person is rejected.

Chapter 3

System as a whole

According to our problem requirements we are using continuous classification for reducing the search space and fixed radius retrieval strategy. Also we are combining two features for better continuous classification performance, the way of combination is explained later. To start with firstly we need a fingerprint database to work with. Almost all the previous work on fingerprints is been mostly tested on NIST database [5] . It contains only 2000 fingerprints , which is not at all good enough for us therefore synthetically generated fingerprint database of size 10 million fingerprints is used [6]. Secondly for the tests of false accept and false reject we need new images for the former and different impressions of the images present in database for the latter. From now on we will refer, both new fingerprints and different impressions of fingerprints present in database as query fingerprint and the context will make it clear.

For generating different impressions of same fingerprint we have used noise model described in [17]. Preferred noise model has two degrees of freedom namely translation and rotation limits, using those we have defined twelve levels of noise. Values of translation and rotation limits for all noise levels are shown in Table 3.1. Noise Levels upto 4 are termed as easy , 5-7 as moderate and 9-12 as difficult noise levels. We start our experiments assuming fingerprints already present in database with their features and each finger present in database is unique.

For our experiments we have used system having 16 GB of main memory and Hardisk

speed of 7200 RPM. We have setted our time bound referring to the current scenario [20]. The maximum time for deduplication of one fingerprint can be 3 seconds and maximum time for verification can be 1 sec.

Noise Level	Translation limits in pixels		Rotation limits in degrees	
	Min	Max	Min	Max
1	-2	1	-2	1
3	-3	3	-3	3
4	-5	4	-5	4
6	-6	6	-6	6
7	-8	7	-8	7
9	-9	9	-9	9
10	-11	10	-11	10
12	-12	12	-12	12

Table 3.1: Parameter values for different noise level's

As deduplication and verification are two totally different functions we have decided to build separate module for each.

3.1 Deduplication Module

Input to the deduplication module is fingerprint which is claimed to be a new fingerprint, and output of the module says whether the fingerprint is new or already present in database, as the result fingerprints is inserted into database in former case and rejected for the latter. Process diagram is shown in Figure 3.1 and steps involved in deduplication are as follows:

1: Extract features

2: Near neighbor search

3: Combining Features

4: Minutia Match

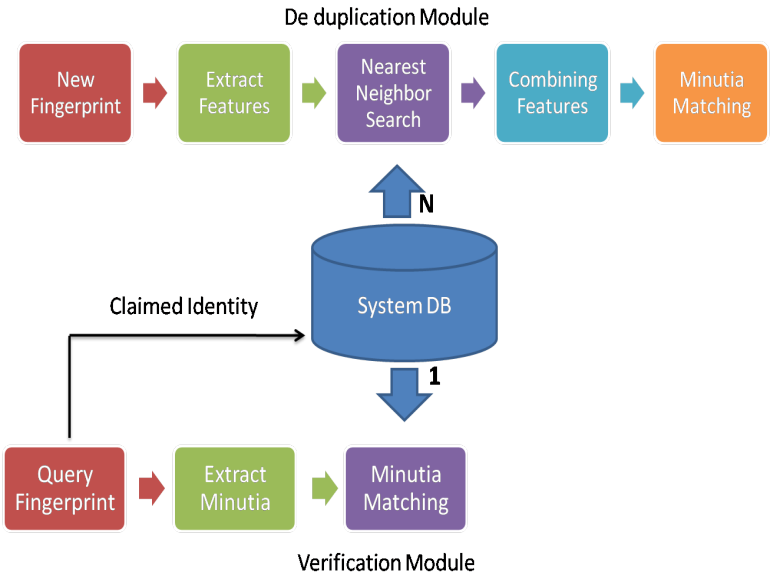


Figure 3.1: Working of Deduplication and verification module

De duplication starts with extraction of directional and fingercode feature of new fingerprint, since feature extraction is very fast and is done once for each fingerprint we are assuming it negligible and also the extraction procedure of fingercode and directional field is independent of each other they are extracted parallel y. Section 4 explains feature extraction process in detail. Basically these features are numerical vector of dimension 127 and 192 respectively.

Second stage is nearest neighbor search which is one of two major time taking part of the pipeline. Using these two numerical vectors nearest neighbors search is applied in their respective tables, Since the database size is very big, efficient methods for nearest neighbor search have to be used, we have used B-tree and Locality Sensitive Hashing [14] for our work. PostgreSQL has inbuilt B-tree index which we have directly used and LSH is package provided by Alexandr Andoni and Piotr Indyk which uses projection of high dimensional data to lover dimensional space to speed up the search, B-tree gives results accurately that is if the point is in the neighbor hood it will be reported with probability 1, and LSH gives reports results with some success probability which is user

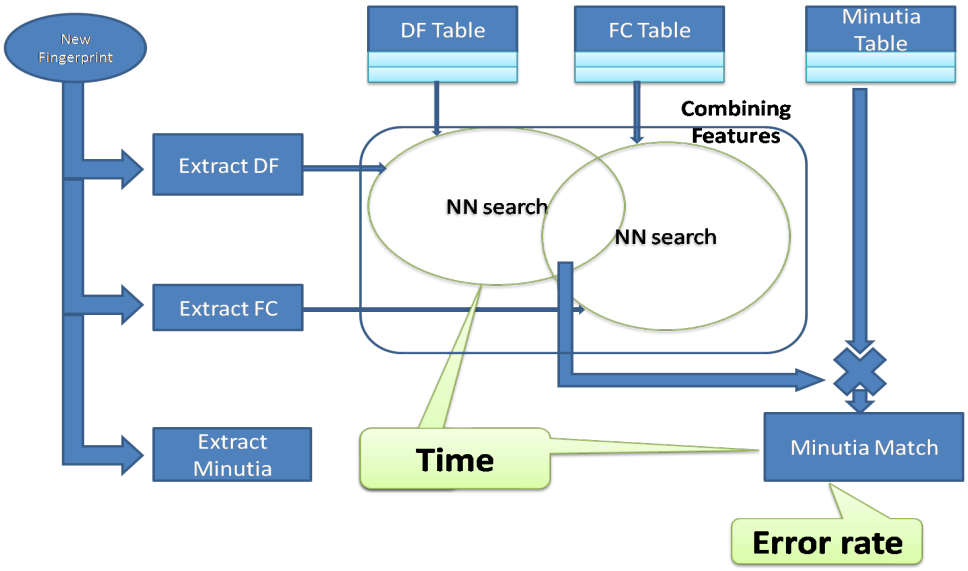


Figure 3.2: Detailed Work Flow Of Deduplication procedure

defined. Detailed description, results and comparison of these schemes are described in section 5.

Nearest neighbor obtained is such that it assures zero error, that is if the fingerprint is present in database it should be in its nearest neighbor set and if the fingerprint is not present in database we can say 'no' only by looking at nearest neighbor set. Since both the near neighbor are with zero error our reduced search space or the final nearest neighbor set can be form fingercode or directional or the intersection of both nearest neighbor set, which is decided by to the trade off between the time taken to reach zero error and the cardinality of nearest neighbor set. Final step is the exhaustive minutia match (detailed explanation in section 6) which matches query fingerprint with each fingerprint in reduced set. Detailed process flow of deduplication is explained in Figure 3.2 . Since feature extraction for fingerprint images is very fast, nearest neighbor search and minutia match are the two time consuming stages which we have to optimize. Since no matching algorithm can assure hundred percent accuracy the whole deduplication process will have some error which is quantified in minutia match stage in terms of false accept rate and false reject rate.

3.2 Verification Module

Verification as defined before means to verify that a person really is the person who he or she claims to be. Prerequisite for verification is that the person should already be enrolled in the system and should have a identity number. We are using minutia match (explained in section 6) for verification purpose. First step will be to check whether the identity number given exists in the database and then extract minutia feature of the given fingerprint and then retrieve minutia feature of given identity number form database and apply minutia matching using both which will output yes or no. Basically its a 1:1 matching and our minutia match algorithm is fast enough to match the required time bound. Figure 3.1 show the flow of verification module.

Chapter 4

FEATURE EXTRACTION

Feature of a fingerprint is the one which represent fingerprint as a numerical vector in order to facilitate classification and matching. There can be multiple features extracted from fingerprints and we have worked on two of those.

1: Directional Field

2: Fingercode

4.1 Directional Field

Directional field (DF) describes the coarse structure, or basic shape of a fingerprint, it is defined as the orientation of the ridge-valley structures and is the most straightforward feature that describes the shape of a fingerprint. Directional field feature on a sample fingerprint is shown in Figure 4.1 .

Extraction of directional field is done using averaging squared gradient which is taken from [7]. Algorithm 1 shows the steps used by us for extraction of DF.

Using DF directly as features, resulted in very poor results because DF is very sensitive to noise. DF changes abruptly with scaling of image because the block size is fixed also it feature value changes if the mask of image changes. To avoid this we have used **directional field with Multi space KL transform(MKL)** proposed by R. Cappelli, D. Maio, and D. Maltoni in [9]. MKL is the multispace generalization of KL transform.

Algorithm 1 Directional Field Extraction

1: Image is divided into the blocks of size WxW
2: **for** Each block **do**
3: Calculate gradient value g_x and g_y along X and Y direction for each pixel of the block using Sobel masks.
4: For each block Average squared gradient θ is calculated by
$tan2\theta = \frac{2\Sigma\Sigma(g_x.g_y)}{\Sigma\Sigma(g_x^2 - g_y^2)}$
5: Directional field for the block is DF $= \frac{\theta}{2} + \frac{\Pi}{2}$,
Since Directional field is perpendicular to the Average Gradient.
6: **end for**

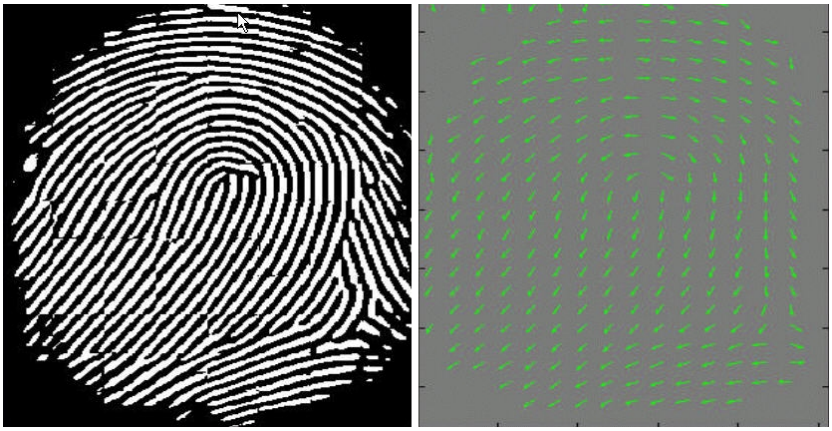


Figure 4.1: Directional Field feature for a sample fingerprint

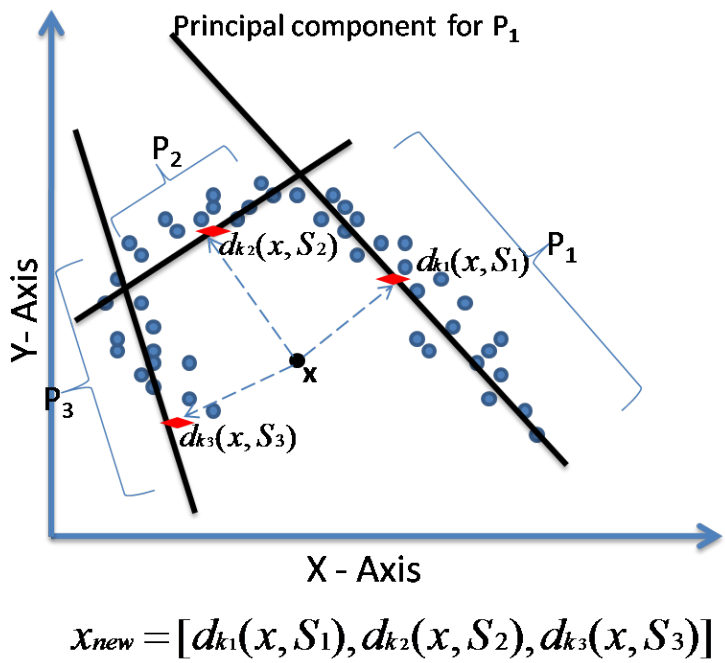


Figure 4.2: Feature vector x in 2-d is converted to x_{new} using MKL

Intuitively in MKL several views of same data are exploited in order to better represent and distinguish the patterns. Figure 4.2 shows an example of MKL on 2-dimensional data having three classes.

Say P is our training set having m fingerprints. After classification P is partitioned into three subsets P_1, P_2, P_3. Given a set of scalars $K = \{K_1, K_2, K_3\}$. MKL transform here is defined by set of subspaces $S = \{S_1, S_2, S_3, \}$, where for each class $c \in \{1, 2, 3\}$, S_c is the $k_c - dimentional$ subspace obtained by calculating the KL transform on the corresponding training subset P_c.

Say x is a directional field vector and $d_{k_c}(x, S_c)$ is the projection of x on subspace S_c. Then our final vector $x_{new} = [d_{k_1}(x, S_1), d_{k_2}(x, S_2), d_{k_3}(x, S_3)]$

DF AS A FEATURE : In our implementation we have taken block size for DF extraction as 16. And window of 16x16 blocks around the core is taken and ridge orientation is each blocks results in 256 dimensional DF vector. For MKL training data of 3000 fingerprints partitioned in five subsets according to Henry classification to form five partitions namely P_A, P_L, P_R, P_W, P_T. And values of $K = \{k_A = 19, k_L = 27, k_R = 27, k_W = 29, k_T = 25\}$ are taken from [8]. Giving DF with mkl to be a 127 dimensional vector.

4.2 Finger Code

This method filters the fingerprint image by a bank of Gabor filters [2] which are tuned to different orientations. Next, the fixed-feature vector, which is called fingerCode, is computed as the standard deviation in a number of sectors, which is indicative of the overall ridge activity in a certain orientation in that sector.

Extraction of Finger Code

Finger Code extraction is taken form [10]. Four main steps involved here are:

1: Determine a reference point(core point in our case) and region of interest for the fingerprint image.
2: Tessellate the region of interest around the reference point.
3: Filter the region of interest in eight different directions using a bank of Gabor filters.
4: Compute the average absolute deviation from the mean(AAD) of gray values in individual sectors in filtered images to define the feature vector or the Finger Code Figure 4.3 and 4.4.

FC as a feature Our fingercode feature is a 192 dimensional feature produced by 24 sectors x with 8 different Gabor filters.

Figure 4.3: Region of interest for fingercode extraction

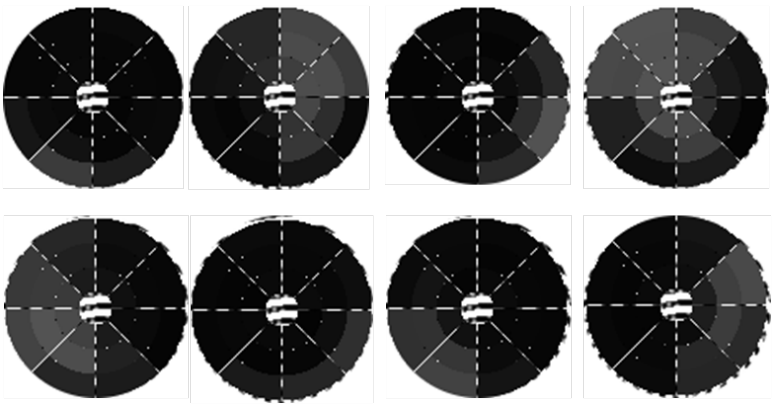


Figure 4.4: Fingercode feature vector

4.3 Core Point Extraction

Core point is a very important global feature of fingerprint image extraction of which helps to make image translation independent. Both directional field and fingercode uses core point extraction. Most of the algorithms present in literature uses orientation image for core point detection, we have used poincare index based method proposed in [11] in our work.

Algorithm 2 Core point extraction by poincare index

1: Divide image into blocks of size WxW.
2: Calculate ridge orientation in every block.
3: For each block $[i, j]$ calculate poincare index say $PI_{[i,j]}$. Poincare index is computed by algebraically summing the orientation differences between the adjacent elements of $[i, j]$. Value of poincare index is discretized such that it only $\in \{0, \pm 180^o, \pm 360^o\}$

$$PI_{[i,j]} = \begin{cases} 0^o, & [i,j]\text{is block of any type} \\ 360^o, & [i,j]\text{is block of Whorl type} \\ -180^o, & [i,j]\text{is block of delta type} \\ 180^o, & [i,j]\text{is block of core type} \end{cases}$$

Figure 4.5 shows blocks containing different singularity points and their poincare index computed using 8 neighboring blocks. This method gives accuracy of around 93% which is very poor, but the big point is their is no core extraction algorithm which gives accuracy that near 100%. By the nature of core extraction it is tailor made machine learning problem but since no big datasets are available for training of learning algorithms, this have never been tried. We are using some of such techniques. Till now we have applied it for short databases training sizes up to size 10000 and got accuracy up to 50%. Currently we are using core points known to us while generation, for DF and FC extraction.

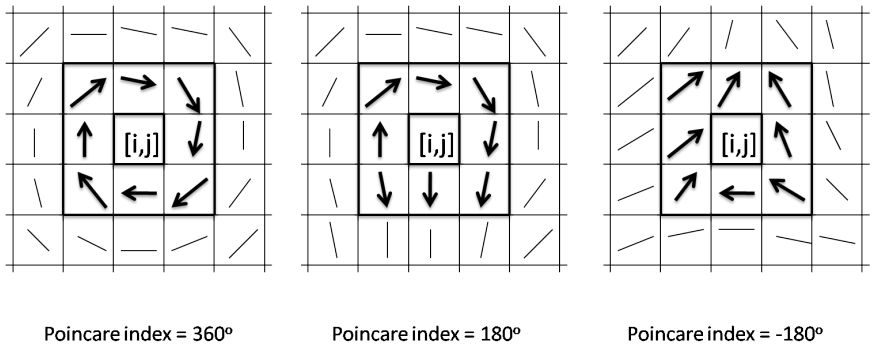


Figure 4.5: Poincare computation in 8-neighborhood for whorl, core, and delta respectively

4.4 Feature accuracy

Accuracy of feature depends upon its robustness with respect to different noise levels. Feature is said to be efficient when it remains same for different impressions of same fingerprint and different for two different fingerprints, In other words points are mapped near by for different impressions of same fingerprint and are far apart for impressions different fingerprints. We have defined two accuracy matrices for continuous classification.

Continuous classification error is defined as ratio of 'number of query fingerprints whose original fingerprint is not found in its nearest neighbor set' and 'Total number of query fingerprints'.

Penetration percentage is defined as ratio of size of nearest neighbor set' and 'Total size of database'.

Error Vs Penetration graph for FC and DF_MKL are shown in Figure 4.6 and 4.7 respectively. With FC feature we have found that for noise level 1, 3 and 4 query fingerprint and original fingerprint are very close to each other in many case the first nearest neighbor of query fingerprint is the original instance of that fingerprint in database and for noise level 6 , 7 and further FC performs very badly. Where as DF_MKL is not effected much by noise levels for noise level 1, 3 and 4 results are poor than FC but for

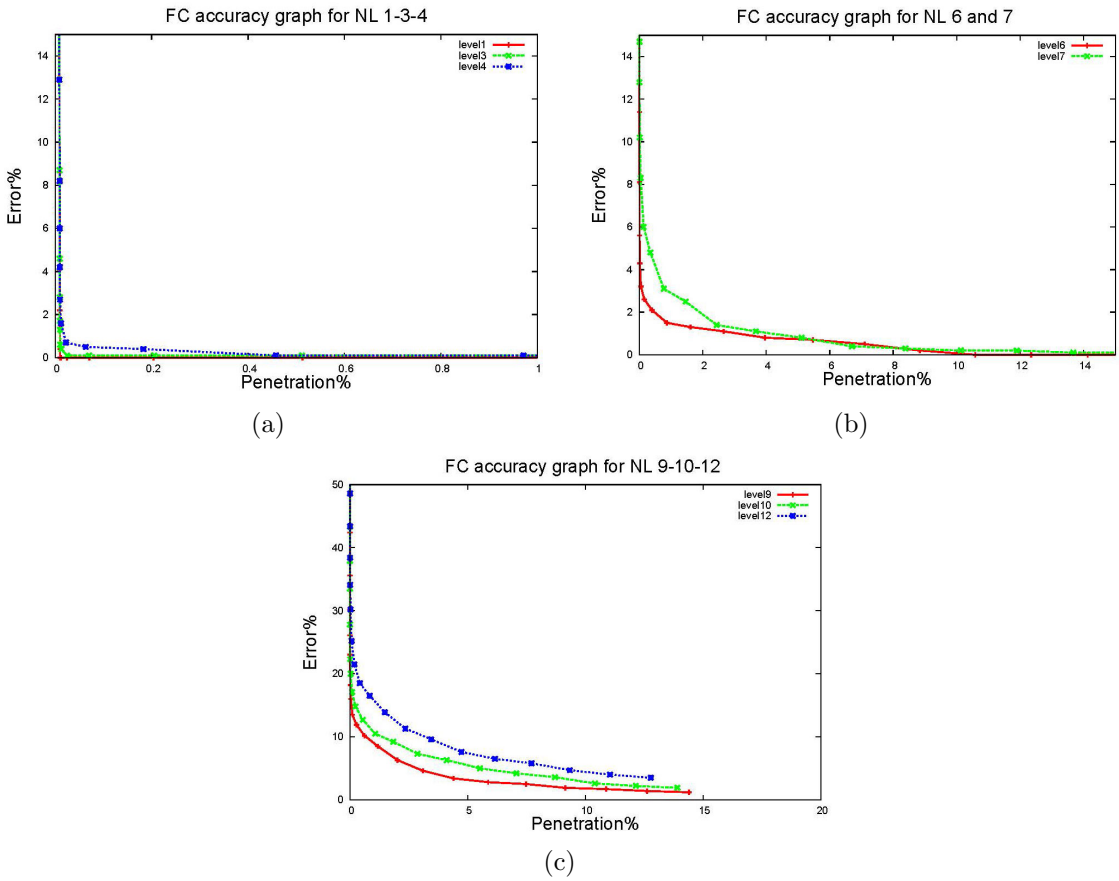
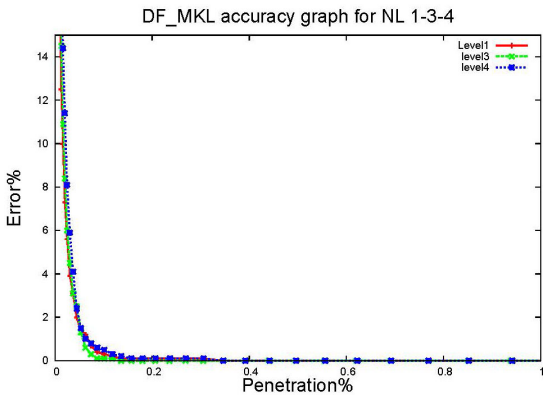
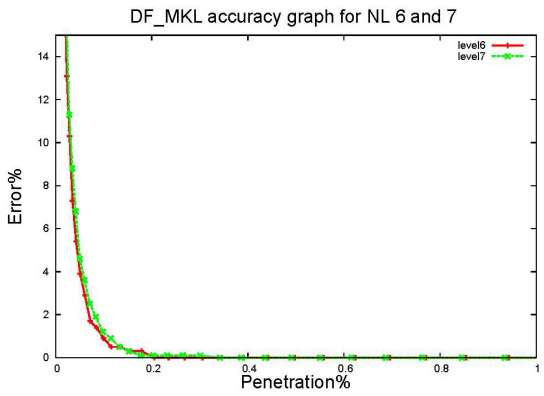


Figure 4.6: Error Vs Penetration graph for FC feature

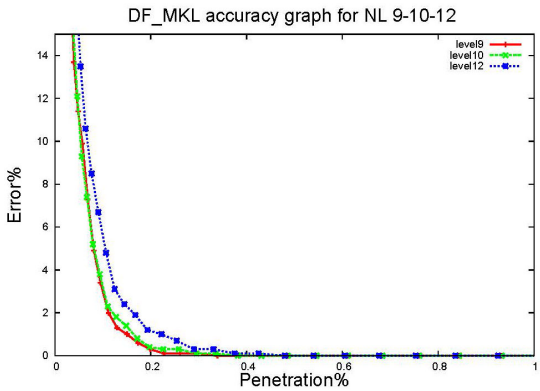
noise level 6 , 7 zero error is achieved at 0.2. penetration. Graphs in figure 4.6 and 4.7 clearly shows that FC is good for noise levels upto 4 and DF_MKL further noise levels using this information **combination of features** is done by electing on the basis of noise levels, if the query fingerprint has noise level up to level 4 FC feature is used for nearest neighbor search and for bigger noise levels DF_MKL is used.



(a)



(b)



(c)

Figure 4.7: Error Vs Penetration graph for DF_MKL feature

Chapter 5

INDEXING

5.1 Btree Indexing

Btree indexing is implemented using multicolumn btree index available in postgerSQL. This method is completely ported into postgersql. Final database schema is shown in Figure 5.1. We have analyzed results for nearest neighbor search on DF and FC with databases of size 1, 2 million and 4 million and results are shown in Figure 5.2 and 5.3. For FC with 4 million data time goes up to 84 sec (approx) per NN search.

5.2 LSH indexing

E2LSH package performs two main steps, preprocessing step and query processing step. Firstly it reads the whole data once and preprocess. Preprocessing mainly includes projecting data on lower dimension space of dimension k and calculating key values for

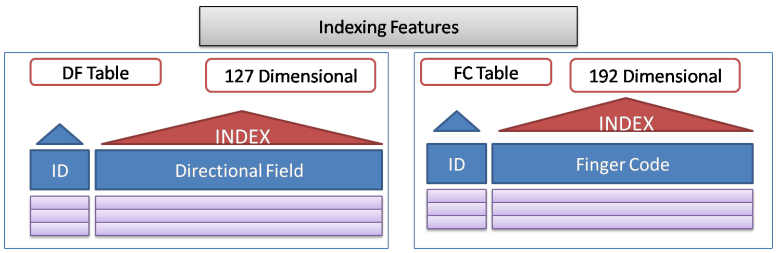


Figure 5.1: Database schema for B tree indexing

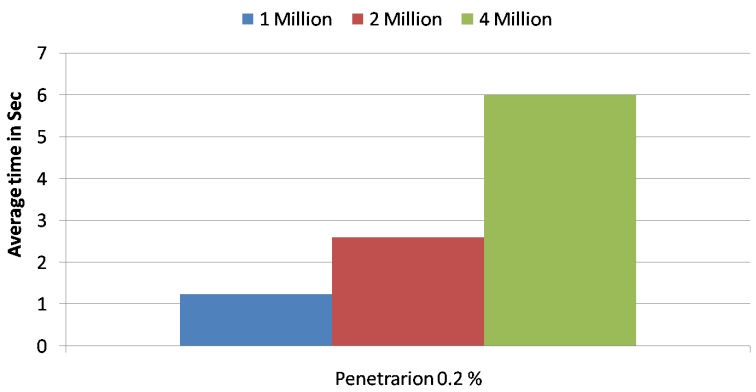


Figure 5.2: Average time taken for nearest neighbor search having penetration of 0.2% using B-tree indexing

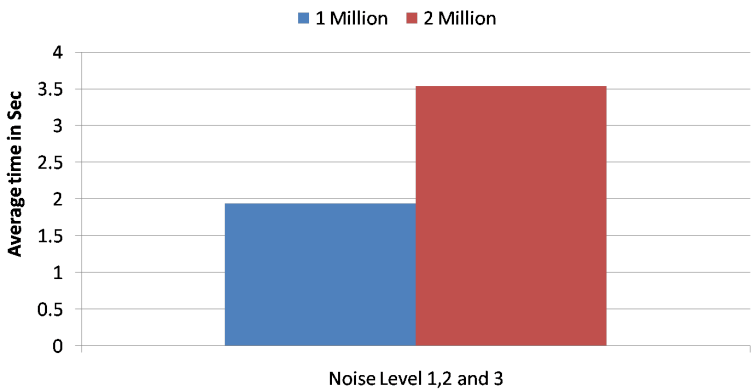


Figure 5.3: Average time taken for Nearest neighbor search using FC feature for NL 1,3 and 4 using B-tree indexing

lower dimensional data and building hash tables on those key values. E2LSH does this procedure many times say L and maintains L different hash table to increase the success probability. Various optimization steps are also performed so as to know the correct value of K and L. that also comes under preprocessing step.

During query processing step it retrieves fingerprints from the bucket where query fingerprint is mapped and checks for points that have distances less than given radius. This is also done for L different table and the final nearest neighbor set is the union of all the nearest neighbor sets. In our work we have separated preprocessing step and query processing step of E2LSH, we have only given the average time taken for nearest neighbor search for query fingerprint, time taken for preprocessing is neglected because preprocessing is done only ones.

All the structures created during preprocessing step are memory resident But if we have dedicated systems with large main memory this can be easily used. Another problem with E2LSH is that it is not scalable, One solution for it is to keep database in chunks of 1 million and build E2LSH structures for every chunk and apply nearest neighbor search on each separately and finally union the results to get required nearest neighbor set. E2LSH package also has a limit of 1048576(2^{32}) data points, same solution as discussed for scalability problem works here, for average query time computation we have added the time for all the chunks. Time results for nearest neighbor search using DF_MKL feature for zero error are shown in Figure . Input to E2LSH is the success probability, radius to be searched and data. Value of radius is so that the *penetration*% is 0.2%. And success probability value used is very high (0.999999) approximately equal to one. Time taken by E2LSH is shown in Figure 5.4

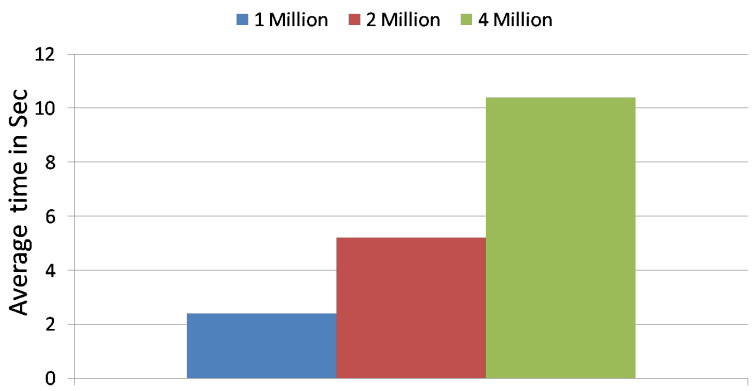


Figure 5.4: Average time taken for Nearest neighbor search using DF_MKL feature for penetration 0.2% using E2LSH

Chapter 6

MATCHING

6.1 Minutia Extraction

In minutiae-based matching, finger prints are represented by a feature vector whose elements are the fingerprint minutiae. Each minutia may be described by a number of attributes, including its location in the fingerprint image, orientation, type (e.g. ridge ending or ridge bifurcation), a weight based on the quality of the fingerprint image in the neighborhood of the minutia, and so on. Most common minutiae matching algorithms consider each minutia as a triplet m = {x,y,θ} that indicates the x,y minutia location coordinates and the minutia angle θ. Extracted minutia's on sample fingerprint is shown in Figure 6.1

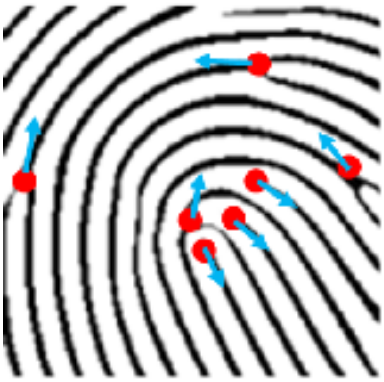


Figure 6.1: Extracted Minutia's On Sample Image

Template minutia file

$minutia1_x$ $minutia1_y$ $minutia1_\theta$

$minutia2_x$ $minutia2_y$ $minutia2_\theta$

....

....

$minutiaN_x$ $minutiaN_y$ $minutiaN_\theta$

$ridgepoint1_x$ $ridgepoint1_y$ 1

....

$ridgepointM_x$ $ridgepointM_y$ 1

$ridgepoint1_x$ $ridgepoint1_y$ 2

....

$ridgepointM_x$ $ridgepointM_y$ 2

....

....

$ridgepoint1_x$ $ridgepoint1_y$ N

....

$ridgepointM_x$ $ridgepointM_y$ N

6.2 Minutiae Matching

A fingerprint matching algorithm exhaustively match two given fingerprints and returns degree of similarity. This is done in two stages. First is alignment stage in which first find the minutia points which has maximum similarity from each of the image, Now translate and rotate template fingerprint so that the the maximum matched minutiae collide each other in all three dimensions. After getting two set of transformed minutia points, comes

match stage which count the matched minutia pairs by assuming two minutia having nearly the same position and direction are identical.

Given two minutia templates with number of minutia's $minu_1$ and $minu_2$ respectively.

Algorithm 3 Minutia Matching

```
for k1 = 1 to minu1 do
    TR_k1 = TransformRidge(k1)
    for k2 = 1 to minu2 do
        TR_k2 = TransformRidge(k1)
        if |TR_k1 − TR_k2| ≥ AT then
            TMs_k1 = TranformAllMinutia(k1)
            TMs_k2 = TranformAllMinutia(k2)
            CompareMinutias(TMs_k1, TMs_k2)
        end if
    end for
end for
```

$TransfromRidge(k)$ returns ridge corresponding to minutia k after translating it to origin and aligning its ridge orientation with $XAxis$ so that the ridges can be matched on same grounds.

$TranformAllMinutia(k)$ returns all minutia points translated and rotated so that minutia k is at origin and its ridge orientation aligns with $XAxis$.

Algorithm 4 CompareMinutias

```
for i = 1 to minu1 do
    for j = 1 to minu2 do
        if |TMs_k1i − TMs_k2j| ≥ MT then
            matched = matched + 1
        end if
    end for
end for
matchScore = matched/minu1 matchScore
```

Performance Average time taken by minutia match algorithm is 0.02 sec and false accept and false reject error's are shown in Figure 6.2 . For noise level one FRR and FAR at knee point is around 1 percent and the error increases as the noise level increases and goes up to 9-10 percent for noise level 6 and further.

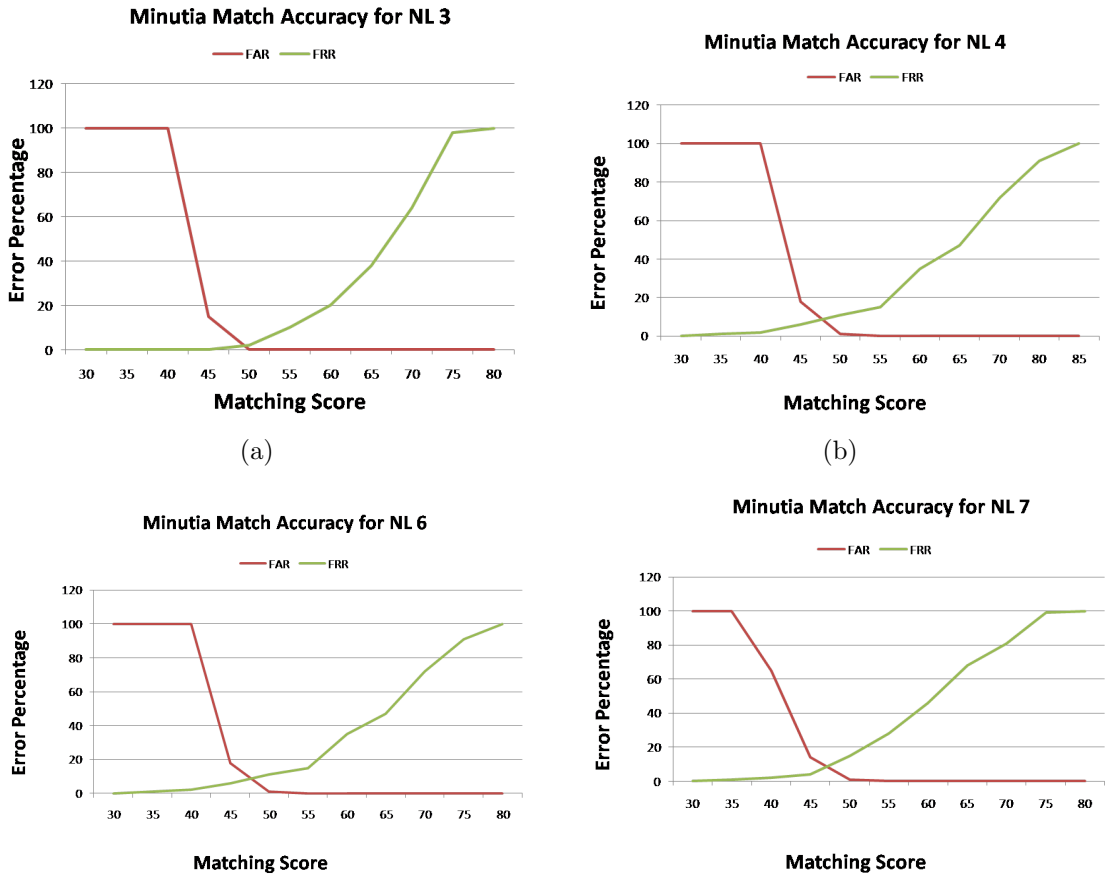


Figure 6.2: FAR and FRR graphs for Minutia match algorithm on different Noise levels

6.3 Implementation in postgreSQL

We have implemented minutia match algorithm inside postgreSQL using "create function" functionality in postgreSQL for creating user defined function. The command used to create function in PostgreSQL is shown below:

```
CREATE FUNCTION minutiaMatch(minutia1 , minutia2) RETURNS boolean AS
' DIRECTORY/filename ',minutiaMatch' 'LANGUAGE C STRICT;
```

By adding minutia match module to postgreSQL we have completely ported our system on postgreSQL.

Chapter 7

Conclusion And Future Work

Average Time taken for deduplication of one fingerprint on database of sizes 1, 2 and 4 million is shown is Table 7.1. Since we have not used any parallelism in nearest neighbor search the time given in table 7.1 is by using only one core of the machine. But minutia match execution is a separate process for each match there is done parallely and the time given is for all eight core of machine is used.

Upto Noise Level 4 and database size of 1 million required time bounds and error bounds are achieved FC works better for lower noise where as DF_MKL is good for higher noise levels. Near neighbor search is the most time consuming part for noise level 1-4 where as Minutia match is bottleneck for noise level 6 - 7. For identification system of UID scale 1000 dedicated machines, running in parallel of our specification are required.

Both Finger code and Directional field depends on core point and detection of core point is not accurate. Since we have a large fingerprint data with known core points we can apply some learning algorithms for efficient core point detection. We can also use other deterministic indexing techniques like pyramid tree indexing and compare the results with LSH and B tree indexing. More indexing features such as minutia triplet for better performance.

Database Size	Noise Level 1-4		Noise Level 6-7	
	Classification Time	Matching Time	Classification Time	Matching Time
1 Million	1.94 sec	0.02 sec	1.2 sec	5 sec
2 Million	2.5 sec	0.02 sec	2.4 sec	10 sec
4 Million	out of bound's	out of bound's	6.0 sec	20 sec

Table 7.1: Time results for deduplication per query fingerprint by our system over different databases size and noise levels

References

[1] "Biometrics Design Standards For UID Applications", *Prepared by: UIDAI Committee on Biometrics*, December 2009

[2] Davide Maltoni and Dario Maio and Anil K. Jain and Salil Prabhakar, "Handbook Of FIngerPrint Recognition", *Springer, second edition, 2009*

[3] Alessandra Lumini and Dario Maio and Davide Maltoni, "Continuous versus exclusive classification for fingerprint retrieval", *Pattern Recognition Letters Volume 18, Issue 10, Pages 1027-1034* , 1997

[4] Johan De Boer and Asker M Bazen and Sabih H Gerez, "Indexing Fingerprint Databases Based on Multiple Features", *In ProRISC the 12th Annual Workshop on Circuits, Systems and Signal Processing Workshop* , 2001

[5] C. I. Watson and C. L. Wilson "Special Database 4, Fingerprint Database", *"U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology",* , 1992

[6] Afzalul Haque Ansari "Synthetic Fingerprint Generation", Master's Thesis, Indian Inst. of Science, June 2011.

[7] Asker M Bazen and Sabih H Gerez, "Systematic Methods for the Computation of the Directional Fields and Singular Points of Fingerprints", *IEEE Transactions on Pattern Analysis and Machine Intelligence Volume 24 Issue 7* ,2002

[8] Raffaele Cappelli and Dario Maio and Davide Maltoni "Fingerprint Classification Based On Multi-space kl". *IEEE Transactions on Pattern Analysis and Machine Intelligence Volume 23 Issue 9* ,2001

[9] Raffaele Cappelli and Dario Maio and Davide Maltoni "Multispace KL for Pattern Representation and Classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence Volume 23 Issue 9* ,2001

[10] Anil K Jain and Salil Prabhakar and Student Member and Lin Hong, "A Multichannel Approach to Fingerprint Classification", *IEEE Transactions on Pattern Analysis and Machine Intelligence Volume 21 Issue 4*,1999

[11] Jin Bo and Tang Hua Ping and Xu Ming Lan, "Fingerprint singular point detection algorithm by Poincare index", *WTOS, 7:14531462* ,2008.

[12] Wuzhili "Fingerprint recognition", Masters thesis, Hong Kong Baptist University, April 2002.

[13] Alexandr Andoni and Piotr Indyk "E2LSH 0.1 User Manual", June, 2005

[14] Alexandr Andoni and Piotr Indyk "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions", *FOCS '2006 Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*

[15] B. Sherlock and D. Monro, "A Model for Interpreting Fingerprint Topology", *Pattern Recognition Volume 26, Issue 7, Pages 1047-1055* ,1993

[16] Pedro R. Vizcaya and Lester A. Gerhardt, "A Nonlinear Orientation Model for Global Description of Fingerprints", *Pattern Recognition Volume 29, Issue 7, Pages 1221-1231* , 1996

[17] "R. Cappelli and A. Erol and D. Maio and D. Maltoni", "Synthetic Fingerprint-image Generation", *Proceeding ICPR '2000 Proceedings of the International Conference on Pattern Recognition - Volume 3*

[18] *http://www.fbi.gov/about-us/cjis/fingerprints_biometrics/iafis/iafis*

[19] *http://www.postgresql.org/docs/8.3/static/release-8-3-6.html*

[20] *http://en.wikipedia.org/wiki/Fingerprint Verification Competition*

[21] *http://biolab.csr.unibo.it/home.asp*

[22] *http://uidai.gov.in/*

Appendix A

Correlation between features

Intuitively correlation between DF and FC can be explained by looking at their extraction procedure, DF and FC captures direction of ridges and texture information around the core respectively. In the first step of FC extraction image is filtered by eight Gabor filters which are oriented in eight different directions and what Gabor filter does is it smoothens the ridges that are oriented along its orientation and distorts others. Therefore this filtering step to capture some information of ridge orientation with it.

To analyze correlation empirically between features we have taken some random set of query points and applied nearest neighbor search on them and retrieved nearest neighbor set of sufficiently large size for both FC and DF_MKL. Then sorted the the retrieved set with respect to distances from query points. And the tried to find out intersection between these sets by taking first 100 fingers then 100 to 200 and so on we call it bins. What we have found is the size of intersection set obtained is more in first bins Figure A.1 shows the size of intersection set for different bins.

Secondly we have found that for very big nearest neighbor set almost all the fingerprints retrieved are same as shown in Figure A.2 (Y axis show the percentage of fingerprints that are common and Y axis shows the size of nearest neighbor set) for large values at X axis percentage of common fingerprints is almost 100.

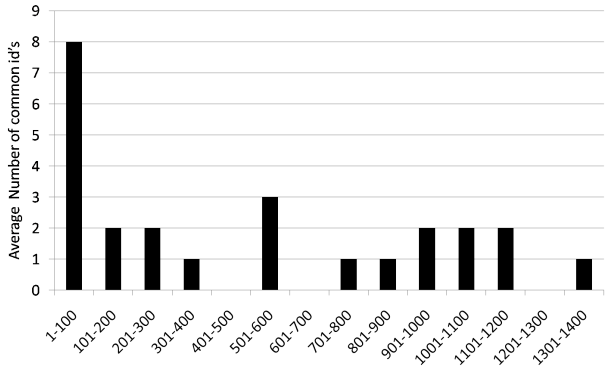


Figure A.1: Correlation between FC and DF for different bins of nearest neighbor set

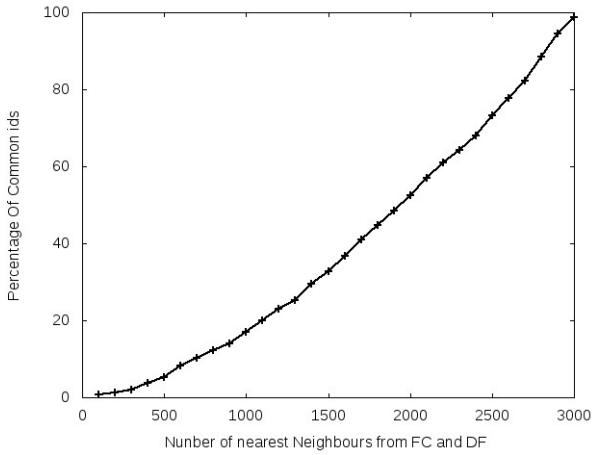


Figure A.2: Correlation between FC and DF for increasing size of nearest neighbor set

Appendix B

Preprocessing of fingerprint image

Histogram Equalization

Histogram equalization is to expand the pixel value distribution of an image so as to increase the perceptional information. The histogram after the histogram equalization occupies all the range from 0 to 255 and the visualization effect is enhanced as shown in Figure B.1

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. Binarization is to transform the 8-bit Gray fingerprint image to a 1-bit image as shown in Figure B.2.

Figure B.1: Histogram Enhancement.(Left) original image, (Right)Enhanced image

Figure B.2: Fingerprint image after binarization.(Left) Enhanced gray image, (Right)Binarized image

Fingerprint Image segmentation

The image area without effective ridges and furrows is first discarded since it only holds background information because region are confusing with those spurious minutia that are generated when the ridges are out of the sensor.

Appendix C

Feasibility of UID database on postgres

To check the feasibility of postgres for 1.1 billion fingerprints we have taken 20 distinct images generated by Sfinge and replicated them 50 million times. As the images generated by Sfinge have size approx 80 KB each. Keeping space constraint in mind, we required to store each fingerprint in almost 1 KB space. So we processed the fingerprint images in a way that they contain about 85 percent of the image area and almost all important features of fingerprints.

Schema used :

Info (id integer, image bytea)

It took 3 days to insert 1.1 billion images.

Index