B. TECH PROJECT REPORT
On
Performance Comparison of Two Algorithms for Partial Fingerprint Matching

BY

Chinmay Anand

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

NOVEMBER 2019

Performance Comparison of Two Algorithms for Partial Fingerprint Matching

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

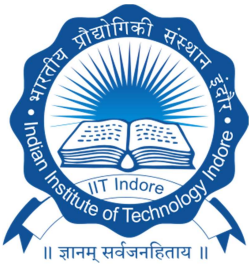
COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Chinmay Anand (160001018)

Guided by:

Dr. Somnath Dey,
Associate Professor,
Discipline of Computer Science and Engineering,
IIT Indore



INDIAN INSTITUTE OF TECHNOLOGY INDORE
November 2019

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **"*Performance Comparison of Two Algorithms for Partial Fingerprint Matching*"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'COMPUTER SCIENCE AND ENGINEERING' completed under the supervision of **Dr. Somnath Dey, Associate Professor, Discipline of Computer Science and Engineering,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Signed:

Chinmay Anand

CERTIFICATE

This is to certify that the B.Tech Project entitled, "*Performance Comparison of Two Algorithms for Partial Fingerprint Matching*" and submitted by Chinmay Anand in partial fulfillment of the requirements of B.Tech Project embodies the work done by him under my supervision.

Dr. Somnath Dey,
Associate Professor,
Discipline of Computer Science and Engineering,
IIT Indore

PREFACE

This report on *"Performance Comparison of Two Algorithms for Partial Fingerprint Matching"* is prepared under the guidance of Dr. Somnath Dey, Associate Professor, Discipline of Computer Science and Engineering, IIT Indore.

Through this report, we have tried to provide a detailed description of a partial fingerprint matching algorithm based on minutiae and ridge shape features. We have implemented this algorithm in Python and C++ and performed an in-depth comparison against the results obtained from a standard tool (bozorth3) which uses minutiae as the only feature. We have tried to explain every step of our project in a simple and lucid manner.

Chinmay Anand
B.Tech IV Year,
Discipline of Computer Science and Engineering,
IIT Indore

Acknowledgements

I wish to express our heartfelt gratitude to **Dr. Somnath Dey** for his sincere guidance and kind support in completing this project.

I am extremely thankful to **Mr. Mahesh Joshi** for his cooperation and guidance throughout the project.

I also thank my project partners as well as my friends who contributed to the making of this project.

Chinmay Anand
B.Tech. IV Year
Discipline of Computer Science and Engineering
IIT Indore

Abstract

The sensing area in modern mobile devices like smartphones, tablets etc. is very small for capturing the full fingerprint image. The partial fingerprints captured by these sensors give unreliable results when used with full fingerprint matching algorithms using minutiae as the only feature. So, in order to make a robust partial fingerprint recognition system, matching algorithm for partial fingerprints should consider other features along with minutiae.

In this project, we implemented a research paper (Lee et al. [1]) which uses additional features (here referred to as ridge shape features (RSFs)) along with minutiae for partial fingerprint matching and performed an in-depth performance comparison with the standard algorithm called bozorth3 implemented by NBIS (NIST Biometric Image Software) which uses minutiae as the only feature.

We used mindtct (a tool by NBIS) for extracting minutiae features and implemented the algorithm (Lee et al. [1]) in C++ and Python for extracting RSFs and final matching score. We created partial fingerprints from the standard dataset FVC2004 and used various parameters like cumulative genuine matches, false acceptance rate (FAR), false rejection rate (FRR), number of master prints and ROC curve for in-depth performance comparison.

Though this algorithm outperforms bozorth3 in most cases, the result obtained was not as good as claimed by the paper (Lee et al. [1]).

Contributions

The major contributions made by me are as follows:

1. Implementation of RSF extractor
    - Calculation of ridge width at each ridge point
    - Estimation of concave and convex ridge segments
2. Implementation of partial fingerprint matching algorithm based on minutiae and RSF features in Python and C++.
    - Algorithm to calculate RSF matching score for the overlapped region
    - Fusion of minutiae matching and RSF matching scores
3. Performance analysis
    - FAR and FRR
    - Master Fingerprints

I would also like to appreciate the contributions made by partner. The major contributions made by them include:

1. Implementation of partial fingerprint matching algorithm based on minutiae and RSF features in Python and C++ (remaining parts).
2. Image enhancement of partial fingerprint images
3. Performance analysis
    - cumulative genuine matches
    - ROC curve
4. Partial fingerprint generation from databases of FVC2004.
5. Generation of xyt files for above partial fingerprints using *mindtct*.
6. Calculation of matching score between every pair of partial fingerprints using *bozorth3*.

Table of Contents

Sl. No.	Topic	Page No.
1	Candidate's Declaration……..	i
2	Certificate……..	i
3	Preface……...	iii
4	Acknowledgements……...	v
5	Abstract…….	vii
6	Contributions……	ix
7	List of Figures……...	x
8	List of Tables…….	xi
9	List of Abbreviations……	0
10	Chapter 1: Introduction……...	1
	1.1 General Fingerprint Matching Algorithm……...	2
	1.2 Partial Fingerprints and Need for Additional Features….	3
	1.3 Objectives……	4
11	Chapter 2: Literature Review…….	5
12	Chapter 3: Partial Fingerprint Matching……..	9
	3.1 Image Enhancement……...	9
	3.2 Feature Extraction…….	11
	3.2.1 Minutiae……...	11
	3.2.2 Ridge Shape Features……..	12
	3.3 Matching Algorithm Using Minutiae and RSF……...	15
	3.4 Matching Algorithm Using Minutiae Only (bozorth3)…..	22
13	Chapter 4: Performance Analysis……...	23
	4.1 Dataset…….	23
	4.2 Performance Parameters and Protocol……	24
	4.3 Results……	27
14	Chapter 5: Conclusion…….	33
15	References…….	35

List of Figures

Sl. No	Description	Page No.
1	Figure 1.1 Fingerprint Authentication System……...	1
2	Figure 1.2 General Fingerprint Matching Algorithm…….	2
3	Figure 3.1 Steps Involved in Fingerprint Enhancement…….	9
4	Figure 3.2 Minutiae……	11
5	Figure 3.3 Concave and Convex Ridge Segments…….	12
6	Figure 3.4 Ridge Width…….	13
7	Figure 3.5 Comparison of Ridge Width with Convex and Concave Segments……..	14
8	Figure 3.6 Matching Algorithm Flowchart……	15
9	Figure 3.7 Local Structure of a Minutiae……...	16
10	Figure 4.1 Sample Testing Database……..	26
11	Figure 4.2 Cumulative Percentage (a)DB1_A (b)DB2_A….	28
12	Figure 4.3 Master Fingerprints (a)DB1_A (b)DB2_A……...	29
13	Figure 4.4 FAR and FRR vs Threshold (a) DB1_A (bozorth3) (b)DB1_A (rsfAlgo)……..	31
14	Figure 4.5 FAR and FRR vs Threshold (b) DB2_A (bozorth3) (b)DB2_A (rsfAlgo)……..	31
15	Figure 4.6 ROC Curve (a) DB1_A (b) DB2_A…….	32

List of Tables

Sl. No.	Description	Page No.
1	Table 4.1 Cumulative Percentage DB1_A……	28
2	Table 4.2 Cumulative Percentage DB2_A……	28
3	Table 4.3 Master Fingerprints DB1_A……..	30
4	Table 4.4 Master Fingerprints DB2_A……..	30
5	Table 4.5 EER and Threshold (T_o)……	32

List of Abbreviations

- **RSF**- **R**idge **S**hape **F**eatures
- **FVC**- **F**ingerprint **V**erification **C**ompetition
- **FAR**- **F**alse **A**cceptance **R**ate
- **FRR**- **F**alse **R**ejection **R**ate
- **BFS**- **B**readth **F**irst **S**earch
- **CBFS**- **C**oupled **B**readth **F**irst **S**earch
- **NIST**- **N**ational **I**nstitute of **S**tandards and **T**echnology
- **NBIS**- **NIST B**iometric **I**mage **S**oftware
- **MCS**- **M**atching **C**ertainty **S**core
- **ERR**- **E**qual **E**rror **R**ate

CHAPTER 1

Introduction

Fingerprint identification system is one of the oldest and the most reliable biometric recognition system which takes a fingerprint image from an individual, extract different features from it, and then compares them against extracted features from fingerprints already present in the database as shown in Fig. 1.1. It is one of the most widely used and well-known biometrics. Some of the reasons for their extensive use are uniqueness, consistency over time, inherent ease in acquisition, numerous sources (ten fingers per subject) available for collection.

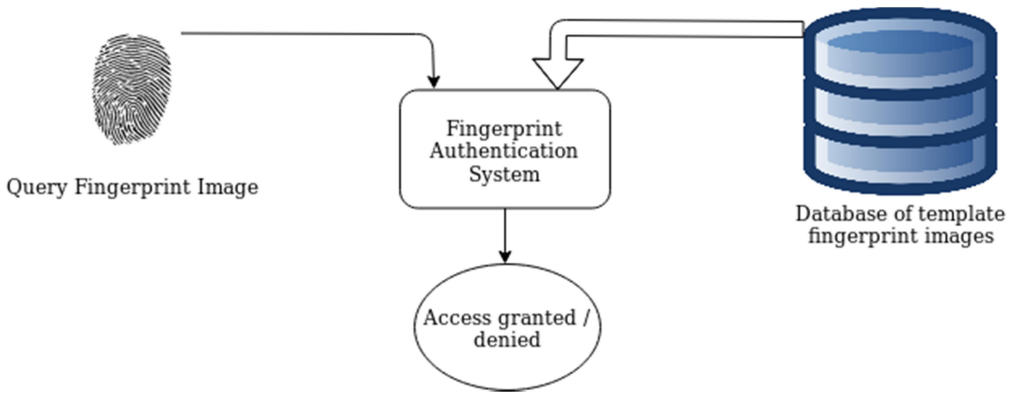


Fig. 1.1 Fingerprint Authentication System

In the beginning, It was mainly used for criminal identification purposes by law enforcement agencies. But in the last few years, it has been implemented in almost every field where a person's identification is important such as banking, college attendance, airport security, mobiles, laptops, etc. Modern mobile devices like smartphones, tablets, laptops have started to adopt miniaturized fingerprint scanners which capture only small partial fingerprint images.

Since the number of features to be extracted depends on the area of fingerprint scanned, the algorithms used for full fingerprint matching does not perform well with partial fingerprints. Generally, full fingerprint matching algorithm uses minutiae as the only feature for matching

and for a successful match, it needs more than 12 matched minutiae pairs. However, partial fingerprint images scanned by miniaturized fingerprint scanners contain less than the required number of minutiae to make a robust decision. In order to make a robust partial fingerprint identification system, other features should be considered along with minutiae.

1.1 General Fingerprint Matching Algorithm

Fingerprint matching algorithm is the soul of any fingerprint identification system. The general steps of every fingerprint matching algorithm are very similar up to some extent as shown in Fig. 1.2. It takes two images (Template and Query) as input, applies image enhancement on them, extracts features, then finally applies a matching algorithm on extracted features to determine the extent of similarity between both images. Fingerprint matching algorithms may differ in how each step has been implemented, i.e. they may differ in how and what features are being extracted, how the images are being enhanced, etc.

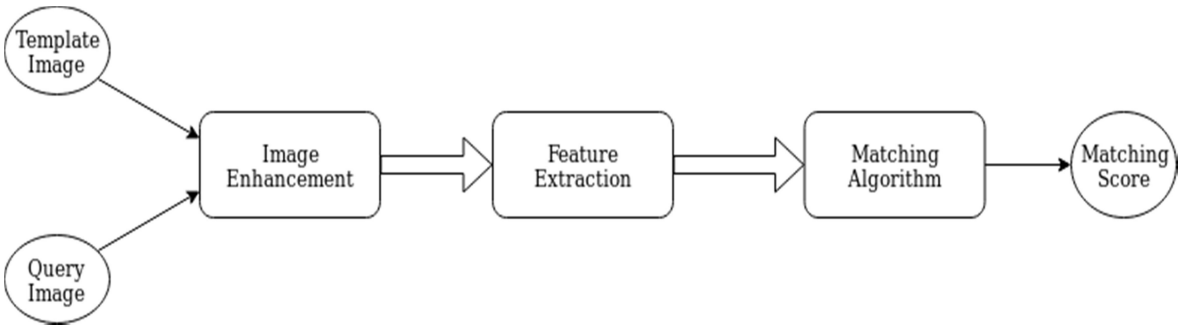


Fig. 1.2 General Fingerprint Matching Algorithm

The working of a fingerprint matching algorithm is divided into following major steps :

1. **Image Enhancement**: The images obtained from fingerprint scanners are generally not fit for feature extraction. They are processed to remove noise, improve image contrast and perform some minor reconstructions.

2. **Feature Extraction**: A fingerprint image is constituted of a set of ridgelines which generally run parallel, sometimes terminate and bifurcate. These ridge terminations and bifurcations are referred to as *minutiae*. The minutiae points obtained from both enhanced template and query image are passed to the matching algorithm for matching score calculation.

3. **Matching Algorithm**: The features obtained from template and query image are passed as input to an algorithm which generally uses the position and orientation of these points for finding the matching pairs in order to determine the extent of similarity between both images.

1.2 Partial Fingerprints and Need for Additional Features

Most modern mobile devices like smartphones, tablets and laptops have started to adopt a miniaturized fingerprint scanner, which captures small partial fingerprint images. The algorithm used for full fingerprint matching generally relies on minutiae as the only feature and needs more than 12 matched minutiae pairs for a successful match. The partial fingerprint matching algorithm needs to consider additional features along with minutiae because the number of minutiae extracted from it is not enough to make a robust decision.
Since only minutiae points cannot be relied upon for partial fingerprint matching, additional features called ridge shape features (RSFs) along with minutiae points are used for establishing the similarity between two partial fingerprint images.

1.3 Objectives

This project aims to achieve the following objectives :

1. Implementation of RSF extractor (Lee et al. [1]) in Python

2. Implementation of feature matching algorithm (Lee et al. [1]) in Python and then in C++ for higher efficiency and speed.

3. Generation of partial fingerprints from full fingerprints in FVC2004 dataset and calculation of matching scores for every pair of partial fingerprints using both algorithms, minutiae-based (here *bozorth3*) as well as minutiae + RSFs based (Lee et al. [1]).

4. In-depth performance comparison between the two algorithms using parameters like cumulative genuine matches, FAR (False Acceptance Rate), FRR (False Rejection Rate), number of master prints and ROC Curve.

CHAPTER 2

Literature Review

The major portion of our project was devoted to the implementation of a partial fingerprint matching algorithm based on minutiae and ridge shape features proposed by Lee et al.[1]. The complete algorithm involves different critical steps like image enhancement, feature extraction, pattern matching algorithm, etc., and all steps are not discussed in detail in this paper. In order to have a robust implementation, we had to study various works already done in this field.

Hong et al. [2] proposed a very efficient and fast algorithm for fingerprint image enhancement in order to improve the overall accuracy of a fingerprint verification system. The clarity of a fingerprint image's ridges and valleys structures is improved based on the pre-estimated local ridge orientation and frequency. A fingerprint image is normalized using it's local mean and variance followed by calculation of local orientation and the local frequency at each ridge point. Gabor filter tuned with local ridge orientation and ridge frequency is applied at each ridge point on the normalized image in order to improve its clarity.

Zhao et al. [3] proposed an algorithm for skeleton-based fingerprint minutiae extraction. It computes the binary image from the input grayscale fingerprint image using an adaptive threshold algorithm followed by computing skeletonized image. Unlike most other algorithms, it uses the fingerprint valley instead of ridge for computing binarized and thinned images. Finally, minutiae points are computed from thinned image using the Rutovitz Crossing Number. It removes all the noises like spurious islands, spurs and bridges the image during binarization process only which allows detection of maximum number of minutiae points.

Lee et al. [1] proposed a partial fingerprint matching algorithm which uses new features called ridge shape features (RSFs) along with the conventional minutiae features. The ridge segments which can be categorized into concave or convex segments are considered as ridge shape features. It finally proposes a matching algorithm incorporating RSFs along with minutiae in two stages i.e., minutiae matching and RSFs matching stages. In the minutiae

matching stage, each minutiae is redefined with its neighboring minutiae and RSFs called as its local structure and minutiae matching pairs are finally determined by comparing their local structures. Overlapped region with highest similarity between template and query images is further determined using matched minutiae pairs. RSF matching is further applied on this overlapped region in order to improve the overall matching accuracy. The scores obtained from both matching stages are combined to produce a final matching score which indicates the extent of similarity between template and query images.

Chikkerur et al. [4] proposed a fingerprint matching algorithm based on graph matching principles. It defines a new representation called k-plet to encode the local structure of each minutiae. It also defines a CBFS (Coupled Breadth-First Search) algorithm to determine an overlapping region between query and template regions. CBFS algorithm is a dual graph traversal algorithm which is used to collect all the local adjacent matches, and these collected matched minutiae pairs further define the overlapped region. In order to sort out the ambiguities in matching minutiae pairs, it uses dynamic programming based optimizing approach.

Tico et al. [6] proposed a fingerprint matching algorithm which uses an unconventional way to describe a minutiae. It uses orientation-based minutiae descriptor to represent the local structure of a minutiae. In order to define local structure of a central minutiae m_c, it takes k concentric circles of different radii and then for each neighborhood minutiae which falls under any of these k concentric circles, relative orientation is estimated, and these neighborhood minutiae further define local structure of the central minutiae. The orientation-based descriptor for each minutiae pair is further used to estimate the similarity between template and query images.

Zhang et al. [5] proposed an algorithm for high resolution partial fingerprint alignment using pore-valley descriptors. It utilizes the fact that even partial fingerprints contain enough pores to be considered as reliable features. At first, it extracts pores using a difference of Gaussian filtering approach and then defines a pore-valley descriptor for each pore feature using its local orientation and location. It further proposed a matching algorithm based on pore-valley descriptor in order to estimate the similarity between template and query images.

Cappelli et al. [10] discusses performance evaluation for fingerprint verification systems in detail. It starts with in-depth description of FVC2004 database for full fingerprints and then further goes on to discuss testing protocols and important terms like genuine match, imposter match, false matching rate, false non-matching rate, etc. It also discusses about the performance evaluation method i.e., how the accuracy of a fingerprint verification system can be determined using different graph plots like ROC Curve, error rate vs threshold, cumulative matching curve (CMC), etc.

Kenneth et al. [9] provides a detailed description about NIST Biometric Image Software (NBIS). It describes every detail about NBIS from installation process to algorithmic overview of each tool provided by it. It discusses some of the tools in great detail like mindtct (used for minutiae extraction), pcasys (fingerprint pattern classifier), imgtools (general purpose image utilities). It also provides an overview about bozorth3 (a fingerprint matching tool).

Algorithms and techniques described in these research papers helped us in some way in completing our project. Jain et al. [7] and Sharma et al. [8] provided us with great introduction to biometric and fingerprint verification systems respectively. Our aim was to implement partial fingerprint matching algorithm proposed by Lee et al. [1]. We had to refer to other research papers in order to implement different steps of the algorithm proposed by Lee et al. [1]. We implemented image enhancement algorithm proposed by Hong et al. [2] to generate an enhanced image and implemented thinning algorithm proposed by Zhao et al. [3] to generate skeleton image from the enhanced image. The skeleton image is further used to extract RSFs. We referred Kenneth et al. [9] for the details about mindtct and bozorth3. We used mindtct to extract minutiae features and bozorth3 as base fingerprint matching algorithm against which we performed an in-depth performance comparison of our implemented algorithm (Lee et al. [1]). We implemented CBFS algorithm proposed by Chikkerur et al. [4] in order to estimate overlapped regions between template and query images (it is an intermediate step in our implemented algorithm). We referred Cappelli et al. [10] for various protocols and parameters used in performance analysis of our implemented algorithm.

CHAPTER 3

Partial Fingerprint Matching

This chapter documents algorithmic overview of each step involved in partial fingerprint matching.

3.1 Image Enhancement

Feature extraction is a very critical step in any fingerprint identification system. However, the reliability and performance of a feature extraction algorithm depend heavily on the quality of the input fingerprint images. In order to make a robust fingerprint identification system, it becomes inevitable to incorporate image enhancement algorithm which ensures reliable feature extraction.

Due to variations in impression conditions, skin conditions, acquisition devices (fingerprint scanners), etc., acquired fingerprint images are in general of poor quality. An ideal fingerprint image consists of ridges and valleys, which are alternate and flow in a locally constant direction. Thus, the goal of any image enhancement algorithm is to make a fingerprint image as ideal as possible.

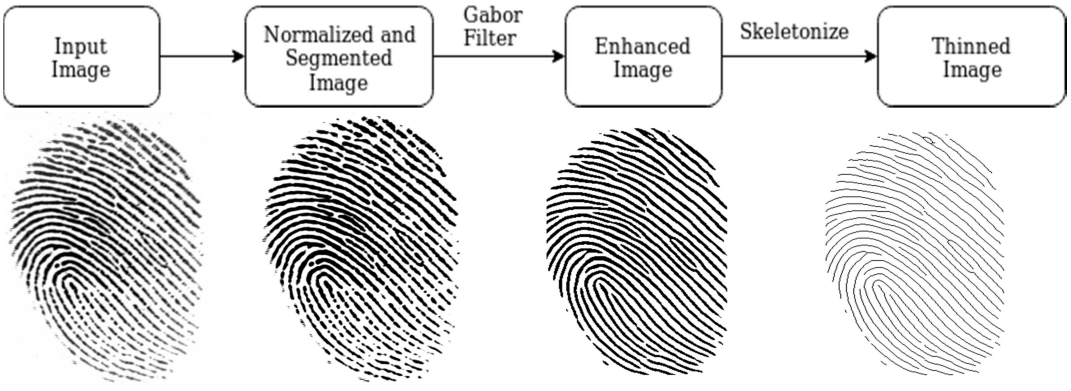


Fig. 3.1. Steps involved in Image enhancement algorithm and their outputs **(a)** Input Image **(b)** Normalized Image **(c)** Enhanced Image **(d)** Thinned/Skeletonized Image

In general, an image enhancement algorithm takes a fingerprint image as input, applies several preprocessing steps on it and finally produces an enhanced image with improved clarity of the ridge structures.

We have tried to keep the details about our image enhancement algorithm to as minimum as possible. For detailed description, refer to Hong et al. [2] as we have followed this for our implementation.

Steps :

- **Input**
  It takes a grayscale image (size: 150x150) as input.

- **Normalization**
  The foreground of input grayscale fingerprint image is segmented using the mean and variance of local pixel blocks so that it has prespecified mean and variance.

- **Local Orientation and Frequency Image Estimation**
  The local orientation and frequency image represent intrinsic properties of the fingerprint images and define invariant coordinates for ridges and valleys in a local neighborhood. These are then fed to Gabor filter which filters the normalized image in order to produce the enhanced image.

- **Filtering**
  Gabor filter which is tuned to local ridge orientation and ridge frequency is applied to the normalized image to produce the enhanced fingerprint image (refer to **Fig. 3.1 (c)**).

- **Skeletonization**
  Ridges in the enhanced image are generally thick, i.e. many curved lines together form a ridge. The aim of skeletonization is to preserve the general structure and connectivity of each ridge but at the same time reduce the thickness of each ridge to one, i.e. now

only one curved line defines a ridge and this curved line is fully connected too (refer to **Fig. 3.1(d)**).

3.2 Feature Extraction

The fingerprint matching algorithm relies completely on the features extracted from a fingerprint image. Thus, the number of features extracted heavily affects the accuracy and performance of a fingerprint identification system. In general, full fingerprint image has more than 40 minutiae points, but partial fingerprint image has very less number of minutiae (< 15). This is the reason why fingerprint identification system which deals with full fingerprint image considers minutiae as the only feature and still gives reliable performance.

In order to ensure robust partial fingerprint identification system, other features must be considered if the matching algorithm relies completely on extracted features. This algorithm considers RSFs (ridge shape features) along with minutiae.

3.2.1 Minutiae

Minutiae are specific points on ridges of a fingerprint image. Generally, this includes points such as ridge bifurcation or ridge ending as shown in Fig. 3.2.

Types of Minutiae :

- Ridge bifurcation
- Ridge ending

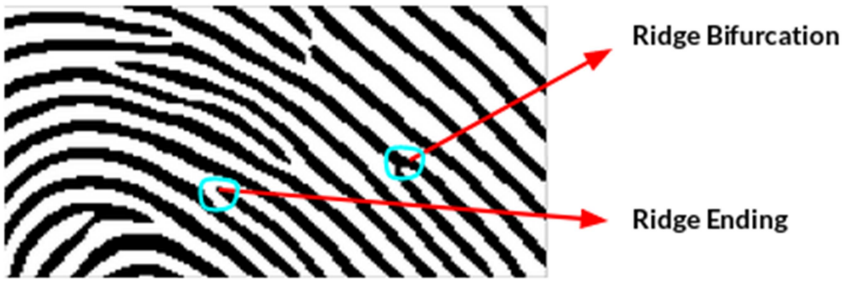


Fig. 3.2 Minutiae

We have used ***mindtct*** (a tool by NBIS aka NIST Biometric Image Software) to detect minutiae points. Any fingerprint image which yields less than ten minutiae is ignored.
Each minutiae point is described by its location (X-Y coordinates), orientation and degree of confidence. Here, the degree of confidence refers to how reliable the extracted minutiae point is.

3.2.2 Ridge Shape Features

When analyzing a fingerprint image, it can be seen that there are some segments of a ridge which are significantly wider or thinner as compared to their neighbors, these segments can be considered as convex and concave respectively as shown in Fig. 3.3. Each ridge point is determined to be either convex or concave in order to extract ridge shape features (RSF). These ridge features are actually continuous convex or concave ridge points and identifiable even if the images are of low resolution. We have tried to explain the process of extracting these features in detail.

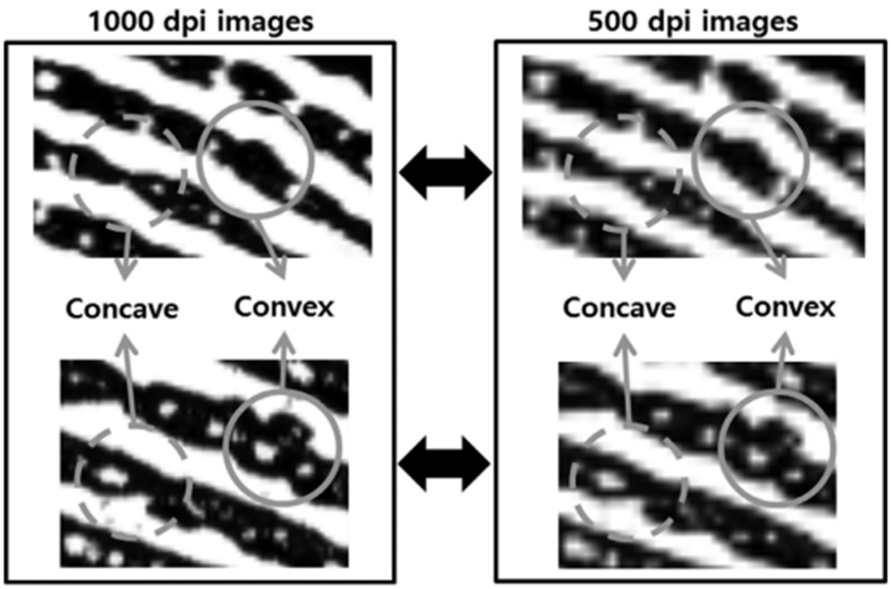


Fig. 3.3 convex and concave ridge segments (Lee et al. [1])

Extraction of RSFs :

1. **Ridge Width Calculation**
    - It needs enhanced image, thinned image and ridge orientation image as inputs. These images are calculated during image enhancement process.
    - For a particular ridge point R_p,
        - determine the direction perpendicular to its local orientation
        - travel in the upward direction of this perpendicular until a valley point (white pixel) V_1 is encountered
        - travel in the downward direction of this perpendicular until a valley point V_2 is encountered
        - Its ridge width r_d is the Euclidean distance between V_1 and V_2.

    Fig. 3.4 shows how ridge width is determined.

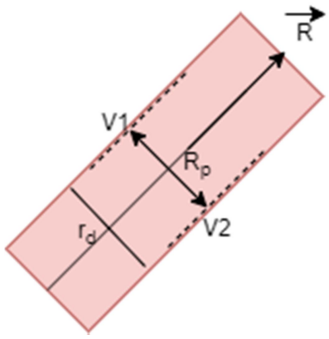


Fig.3.4 Ridge Width

2. **Estimation of Type for Each Ridge Point (Convex or Concave)**

    After obtaining the ridge width at each point of the thinned ridgeline, each ridge point is determined whether it belongs to concave or convex segment by comparing its width with that of its neighboring ridge points (refer to Fig. 3.5 (a)). Let x be a ridge point on the thinned (skeletonized) image. The ridge width at point x is compared with the ridge width of N $(here,\ N = 10)$ neighboring points on both sides, as shown in Fig 3.5. Each of these neighboring points

    $W_i\ , i = \{1,2,3,..,n\}$ is categorized into one of the following categories.

$$S(i) = \begin{cases} n, & W_i \leq W_x - T_w \quad (narrower) \\ s, & W_x - T_w < W_i < W_x + T_w \quad (similar) \\ w, & W_x + T_w \leq W_i \quad (wider) \end{cases}$$

Where,

W_x is ridge with a point x.

W_i is ridge width a neighboring point.

T_w is threshold for comparing ridge widths.

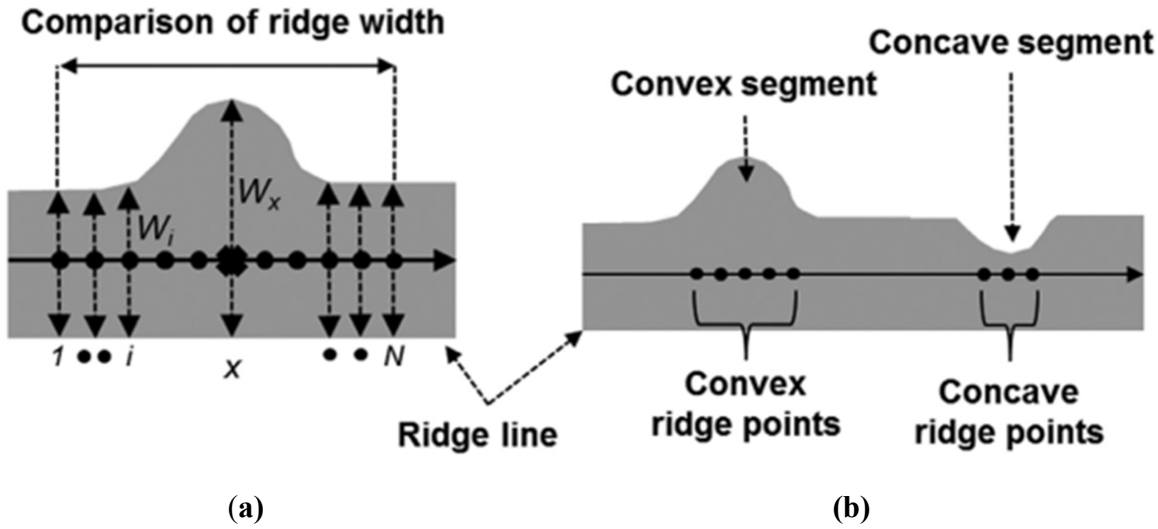


Fig 3.5 (a) Comparison of ridge width **(b)** convex and concave ridge segments (Lee et al. [1])

$S(i)$ is assigned as n, w or s to each of the neighboring points based on the values W_i, W_x and T_w

This is used to classify a ridge point as convex or concave. If more than half of N neighbors are of type w, then point x is classified as concave and if more than half of N neighbours are of type n, then point x is classified as convex.

$T(x)$ which denotes the type of ridge point is determined as follows :

$$T(x) = \begin{cases} concave, & \sum_{i=1}^{N}[S(i) = w] > \frac{N}{2} \\ convex, & \sum_{i=1}^{N}[S(i) = n] > \frac{N}{2} \end{cases}$$

If we get a consecutive set of points which are either concave or convex, then they constitute a ridge segment which is our RSF (or ridge shape feature) as can be seen in Fig. 3.5 (b). We have considered that at least two consecutive points should belong to either concave or convex to consider the segment as RSF.

An RSF r_k is represented by a central ridge point in the RSF segment as follows :

$$r_k = (x_k, y_k, \theta_k, t_k)$$

here, (x_k, y_k) represents the position of the central ridge point of the segment, θ_k represents the ridge orientation at that position t_k denotes the type of RSF (concave or convex).

3.3 Matching Algorithm Using Minutiae and RSF

Fig. 3.6 Matching algorithm flowchart

This matching algorithm (refer to Fig. 3.6) incorporates RSFs along with minutiae to improve the accuracy of partial fingerprint matching. Using the local structure of each minutia, the algorithm first finds matched minutiae pairs. The linear transformation and the overlapped region are further determined using the top few matched pairs. Subsequently, the RSF

matching is performed in the overlapped region. Finally, the matching score is calculated using both minutiae-matching score and rsf-matching score.

3.3.1 Local Structure Estimation

In case of full fingerprints, the local structure of minutiae is typically determined using a fixed number of its nearest minutiae. Since the number of minutiae in a typical partial fingerprint image is not enough to determine its local structure, this algorithm uses RSFs along with minutiae.

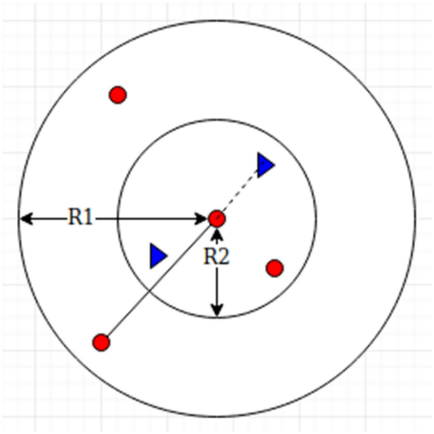


Fig. 3.7. Local structure of a minutia (a) Red circles are minutiae (b) Blue triangles are RSFs

Given a particular minutia, the sequential steps to determine its local structure are:

- Consider this particular minutia as the center (denoted by m_c) of two concentric circles of radius R_1 and R_2 ($R_1 > R_2$) as shown in Fig. 3.7. We considered $R_1 = 60$ and $R_2 = 40$.
- Each minutia and RSF which lies inside the circle of radius R_1 and R_2 respectively is considered as its neighbor.
- Each neighbor (denoted by n_k) is represented by $(\delta_{c,k}, \theta_{c,k}, \varphi_{c,k}, \tau_k)$

where,

$\delta_{c,k}$ is the Euclidean distance between n_k and m_c

$\theta_{c,k}$ is the orientation difference between n_k and m_c

$\varphi_{c,k}$ is the directional difference between n_k and m_c

τ_k is the type of n_k (0: minutia, 1: convex RSF, -1: concave RSF)

- Set of all N neighbors of m_c represented as $(\delta_{c,k}, \theta_{c,k}, \varphi_{c,k}, \tau_k)$ where $1 \leq k \leq N$, defines its local structure.

  It can be represented as

$$L(m_c) = \{(\delta_{c,k}, \theta_{c,k}, \varphi_{c,k}, \tau_k) \quad | \quad 1 \leq k \leq N\}$$

3.3.2 Optimal Similarity Score Calculation

The goal of the matching algorithm for a minutiae pair is to align their local structures in such a way that matching minutiae pairs in their local structure produces maximum similarity score.

We have used dynamic programming to get all the matching pairs and the maximum similarity score by comparing their local structures. The maximum similarity score here is called the optimal similarity score.

The similarity score of minutiae pair (m^T, m^Q) can be defined as :

$$S_{lm}(m^T, m^Q) = \frac{S_m(m^T, m^Q) + S_r(m^T, m^Q)}{2}$$

where,

$$S_k(m^T, m^Q) = \frac{2\sum MCS(n_i^T, n_j^Q)}{N^T + N^Q}, \qquad k = \{m, r\}$$

Here, $S_k(m^T, m^Q)$ is the similarity and $MCS(n_i^T, n_j^Q)$ is the matching certainty score, as explained below:

- **Matching Certainty Score**

  The matching certainty score between n_i^T and n_j^Q (neighbors of m^T and m^Q respectively) depends directly on the difference of their representation, i.e. topological properties with respect to the central minutiae.

  It is calculated as follows:

$$MCS(n_i^T, n_j^Q) = \begin{cases} \dfrac{T - |n_i^T - n_j^Q|}{T}, & if\ |n_i^T - n_j^Q| < T \\ 0, & otherwise \end{cases}$$

where,

T is the threshold for matching certainty.

- **Similarity**

  The similarity is always calculated between any minutiae pair (m^T, m^Q) by matching neighboring minutiae or neighboring RSF denoted by S_m and S_m respectively.

3.3.3 Minutiae Matching Rate and Overlapped Region

After calculating optimal similarity scores for each minutiae pair (m^T, m^Q), top N (in our experiment, $N = 7$) matched pairs are selected as the initially matched minutiae pairs which are further used to determine N overlapped regions. Each region has its own score called minutiae matching rate (mmr) as given by :

$$mmr = \frac{2\sum_{u=1}^{M_m} S_{lm}(m_u^T, m_u^Q)}{(N_{m,o}^T + N_{m,o}^Q)}$$

where,

$N_{m,o}^T$ and $N_{m,o}^Q$ are the total number of minutiae within the overlapped region of template and query image.

M_m is the number of matched minutiae in the overlapped region.

$S_{lm}(m_u^T, m_u^Q)$ is the similarity of matched minutiae pair (m_u^T, m_u^Q).

We implemented central breadth-search algorithm (Chikkerur et al. [4]) in order to determine the overlapped region. In this algorithm, we start with one matched minutiae pair and incrementally find other matching minutiae pairs. These matching pairs finally decide the overlapped region between template and query images.

The matched minutiae pairs within the overlapped region are further used to calculate **linear transformation** for each region.

3.3.4 RSF Matching

The overlapped region with highest mmr is further used for RSF matching. For each RSF point R^T in the template image, we take its projection on the query image using the linear transformation. It is very unlikely that $R^{T'}$ would match exactly with any of the RSF point R^Q on the query image. In order to determine the corresponding projection of R^T in the query image, we take all neighboring RSF R^Q of $R^{T'}$ within the distance of R_c ($R_c = 20$ in our experiment) and find similarity between all RSF pairs (R^T, R^Q) just like we did with minutiae.

Let M_m be the set of all matched minutiae pairs within the overlapped region. The local structure of each minutiae R^Q or R^T is calculated as follows :

- Choose K nearest minutiae of R^T among the matched minutiae pairs in the template region.

$K = min(5, |M_m|)$

- Represent the local structure of R^T as a set of K nearest minutiae in the relative form, i.e., relative distance, radial angle and the orientation difference.
- Now, In order to determine the local structure of R^Q, take corresponding matched minutiae of K nearest neighbor of R^T in the query image and represent its local structure just like R^T.

After calculating local structure for RSF pair (R^T, R^Q), calculate its similarity using the following formula:

$$S_{lr}(R^T, R^Q) = \frac{\sum_{j=1}^{k} MCS(m_j^T, m_j^Q)}{k}$$

where,

k is the number of neighbours as explained above.

(m_j^T, m_j^Q) is a matched minutia pair.

$MCS(m_j^T, m_j^Q)$ is the Matching Certainty Score for minutiae pair (m_j^T, m_j^Q). This is similar to the one explained for minutiae.

Let's denote R_m as the set of matched minutiae pairs and initially it is empty. Now, apply the greedy approach to find all the matched RSF pairs.
Steps to find matching RSF pairs:

- Sort all the RSF pairs based on its similarity in descending order.
- .Now, for each RSF pair (R^T, R^Q), check if any of R^T or R^Q is in the R_m.
    - If yes, then ignore it and process the next RSF pair in the sorted list
    - else, put this pair in R_m and process the next RSF pair in the sorted list

3.3.5 Fusion of Matching Scores

Now, we have the overlapped region with highest minutiae matching rate denoted by mmr, a set of matched minutiae and RSF pairs in this overlapped region dented by M_m and R_m respectively. In order to find the final matching score which will determine the extent of similarity between the template and query image, we combine S_m (minutiae-matching score) and S_{rsf} (RSF-matching score) using a weighing factor λ.

Minutiae-matching score (S_m) can be calculated as :

$$S_m = mmr \cdot \frac{2M_m}{\left(N_{m,t}^T + N_{m,t}^Q\right)}$$

where,

$N_{m,t}^T$ and $N_{m,t}^Q$ are the total number of minutiae in template and query image, respectively.

The RSF matching score (S_{rsf}) is similarly calculated as :

$$S_{rsf} = \frac{2\sum_{v=1}^{M_r} S_{lr}(r_v^T, r_v^Q)}{N_{r,o}^T + N_{r,o}^Q} \cdot \frac{2M_r}{N_{r,t}^T + N_{r,t}^Q}$$

where,

$S_{lr}(r_v^T, r_v^Q)$ is the similarity score of matched RSF pair (r_v^T, r_v^Q).

M_r is the number of matched RSF pairs.

$N_{r,o}^T$ and $N_{r,o}^Q$ are the total number of RSFs in the overlapped region of template and query images.

$N_{r,t}^T$ and $N_{r,t}^Q$ are the total number of RSFs in template and query images.

Final score S_{total} can be calculated as :

$$S_{total} = \lambda \cdot S_m + (1 - \lambda) \cdot S_{rsf}$$

where,

λ is the weighing factor and $0 <= \lambda <= 1$.

λ can be calculated as follows :

$$\lambda = \frac{N_{m,t}^T + N_{m,t}^Q}{\left(N_{m,t}^T + N_{m,t}^Q\right) + \alpha_{rsf}(N_{r,t}^T + N_{r,t}^Q)}$$

where,

α_{rsf} is the weight of the RSF which assigns relative importance of RSF to the minutiae feature.

In our experiment, $\alpha_{rsf} = 0.2$.

3.4 Matching Algorithm Using Minutiae Only (bozorth3)

We have used ***bozorth3*** (a tool by NBIS) which only minutiae as the feature to calculate matching score between query and template images. It is one of the standard tools used in the full fingerprint identification system. Since we have implemented this algorithm, we won't discuss it in detail in this report.

We have used ***bozorth3*** client provided with NBIS package to calculate the score between each pair of partial fingerprint images in each database used for performance analysis (refer to NBIS documentation for how to use ***bozorth3)***.

CHAPTER 4

Performance Analysis

This section contains an in-depth performance analysis of two algorithms for partial fingerprint matching. We have already discussed the minutiae + RSF based algorithm in detail in chapter 3. As another one of the two algorithms, we have used *bozorth3,* a tool for fingerprint matching by NIST Biometric Image Software (NBIS), which uses minutiae as the only feature. We have created results for the same dataset using both of these algorithms and tried to analyze using various parameters.

For the remaining part of this chapter, we will refer the minutiae + RSF based algorithm as ***rsfAlgo*** and the other algorithm as ***bozorth3.***

4.1 Dataset

The databases used for performance analysis of both algorithms are taken from FVC2004. It has four databases namely DB1, DB2, DB3 and DB4. Each one has two sets A and B like DB1_A and DB1_B. Set A of each database contains fingerprints of 100 users with each user having eight samples of the same fingerprint resulting in 800 (100x8) fingerprint images. Set B has fingerprints for only ten users with each user having eight samples of the same fingerprint.

We have used databases DB1 and DB2 to create results using both of the algorithms. Since all these databases contain full fingerprint images, we had to create partial fingerprint images from these full images as discussed below.

The partial fingerprints which are used for testing both algorithms are taken to be of (150 X 150) pixels in size. We take each sample of every subject and generate 150 X 150 partial fingerprints from it in the following way:

1. Take a 150 X 150 square image from the top left corner of the full fingerprint image. Iterate row-wise as well as column-wise taking jumps of 75 pixels. Include all of these images as the partial fingerprints for testing.

2. The partial fingerprints generated are checked for the number of minutiae points using ***mindtct***. If the number of minutiae points in a partial fingerprint comes out to be less than 10, then that partial fingerprint is not considered and rejected from the database.

All the partial fingerprints generated from each sample of all subjects with greater than or equal to 10 minutiae points are used for testing purposes.

4.2 Performance Parameters and Protocols

This section explains all the parameters and protocols used for performance analysis further in section 3 of this chapter.

4.2.1 Performance Parameters

Genuine Score: The score obtained by comparing two samples of the same fingerprint is considered as a genuine score.

Imposter Score: When a score is not genuine, it is considered to be an imposter score.

FAR (False Acceptance Rate): False Acceptance Rate shows the likelihood that a biometric identification system will accept an access attempt by an unauthorized user.

OR

$$FAR = \frac{Imposter\ matches\ accepted}{Total\ number\ of\ imposter\ matches}$$

FRR (False Rejection Rate): False rejection rate shows the likelihood that a genuine user is rejected access by a biometric identification system.

OR

$$FRR = \frac{Genuine\ matches\ rejected}{Total\ number\ of\ genuine\ matches}$$

EER (Equal Error Rate): EER is the point where False Acceptance Rate is equal to False Rejection Rate.

4.2.2 Protocol

FVC protocol has been used for calculation of FAR (False Acceptance Rate) and FRR (False Rejection Rate). The categorization of genuine and imposter is done in the following way when using FVC protocol.

One partial fingerprint of a sample is compared with all partial fingerprints of some other sample of the same subject. Maximum of the comparison scores is taken for comparison with the threshold to decide whether it's a match or not. This constitutes a single genuine match.

One partial fingerprint of a sample of some subject when compared with a partial fingerprint of the same sample of some other subject, then it constitutes a single imposter match.

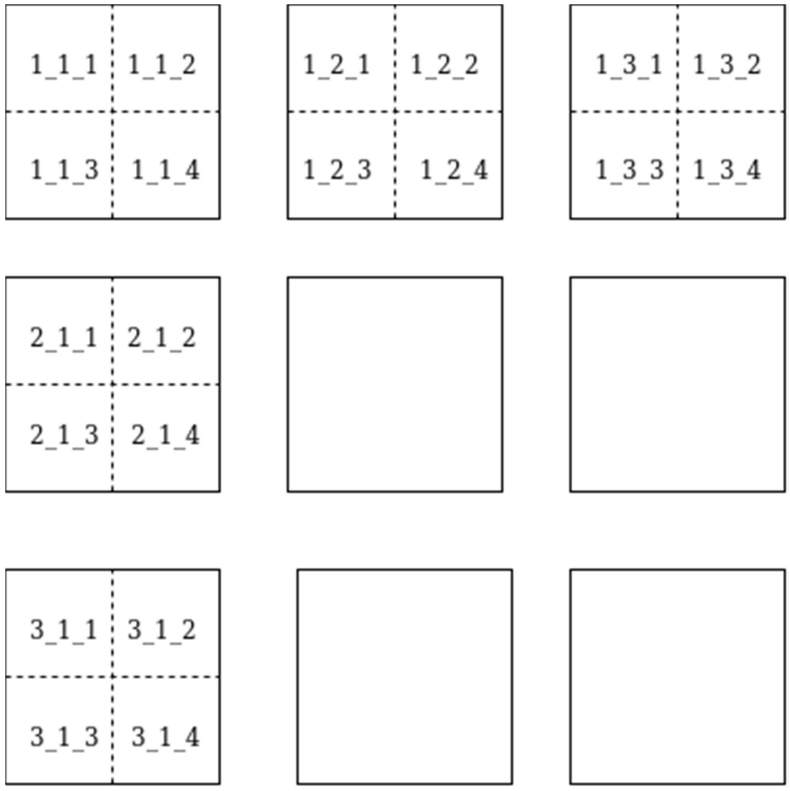


Fig. 4.1 Sample Testing Database

The above Fig. 4.1 shows a sample database in which each partial fingerprint is represented in **subjectId_sampleId_partialfingerprintId** format.

Let's consider genuine and imposter matches for 1_1_1. 1_1_1 will act as our probe fingerprint image, which will be compared with fingerprint images of the same subject as well as with the fingerprint images of other subjects and same sampleId, these constitutes the gallery fingerprint images. 1_1_1 is compared with 1_2_1, 1_2_2, 1_2_3 and 1_2_4, the maximum score out of these four is taken for comparison with threshold and this constitutes a single genuine. So we will have two genuine matches for 1_1_1, one with sample 2 and another with sample 3 of the same subject.

1_1_1 will be compared with all partial fingerprints of the same sample but different subjects. 1_1_1 will be compared with x_1_y, where 2<=x<=3 and 1<=y<=4. So there will be eight imposter matches for 1_1_1.

The genuine and imposter matches are then used for calculation of FAR and FRR for performance comparison.

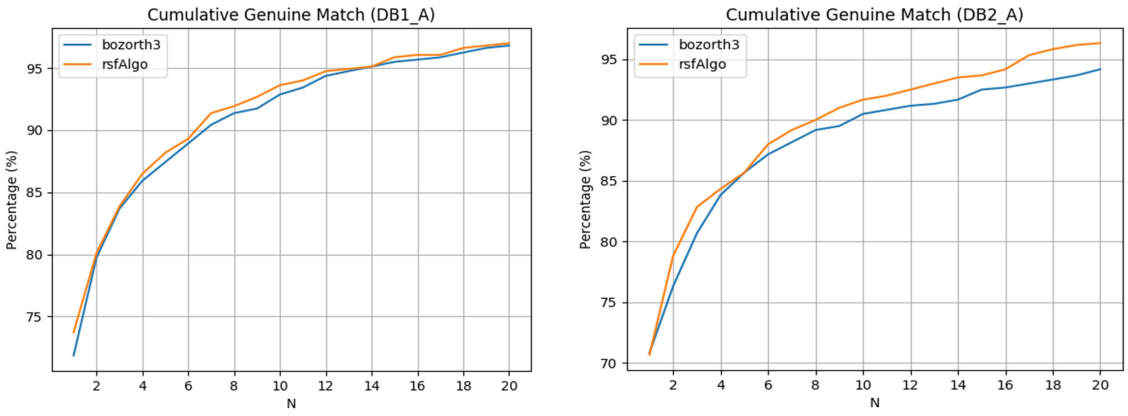
4.3 Results

4.3.1 Cumulative Genuine Matches

A fingerprint is considered to have a cumulative genuine match for a rank N, if we take its top N matches and at least X (here, X=1) of these N matches are genuine.

We have plotted the graph for N vs total percentage of partial fingerprints (provided it follows FVC protocol) which have cumulative genuine matches if we consider top N matches for each fingerprint. We have taken N from 1 to 20.
Since an algorithm is considered better if it has more number of cumulative genuine matches for a rank N, the **rsfAlgo** outperforms **bozorth3** in this regard but not to the same extent as expected.

As we can see from the graphs below (Fig. 4.2), **rsfAlgo** has higher value for cumulative percentage of genuine match for almost all ranks as compared to **bozorth3** for both databases. Both algorithms show greater changes in around smaller ranks, then the graphs flatten out after rank 10.



(a) **(b)**

Fig 4.2 Cumulative percentage **(a)** DB1_A **(b)** DB2_A

Tabular Representation

DB1_A

N	1	2	3	4	5	6	7	8	9	10
bozorth 3	71.86	79.74	83.68	85.93	87.43	88.93	90.43	91.37	91.74	92.87
rsfAlgo	73.73	80.11	83.86	86.49	88.18	89.31	91.37	91.93	92.68	93.62

Table 4.1 Cumulative percentage DB1_A

DB2_A

N	1	2	3	4	5	6	7	8	9	10
bozorth 3	70.83	76.33	80.67	83.83	85.67	87.17	88.17	89.17	89.50	90.50
rsfAlgo	70.67	78.83	82.83	84.33	85.67	88.00	89.17	90.00	91.00	91.67

Table 4.2 Cumulative percentage DB2_A

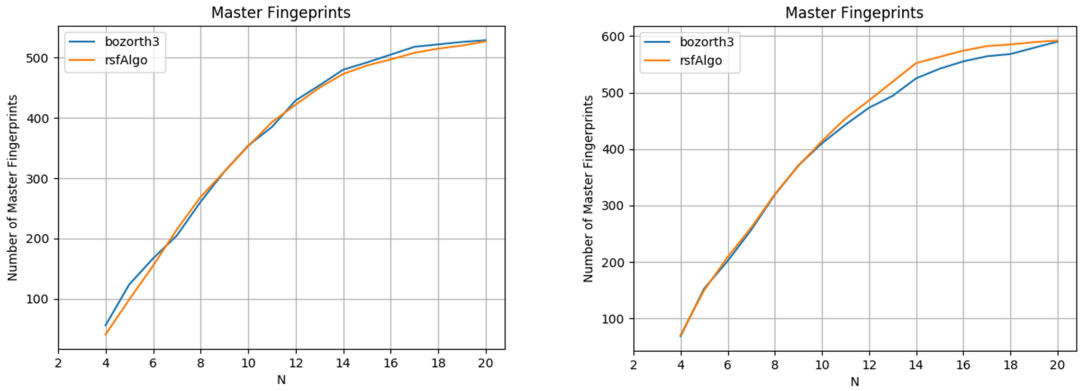
4.3.2 Master Fingerprints

A fingerprint is considered to be a master fingerprint for a rank N, if we take its top N matches and at least X (here, X=4) of these N matches are imposter provided all of these X imposter matches belong to different user's fingerprint.

One of the main goals of any partial fingerprint matching algorithm is to limit imposter matches to a minimum, and this parameter indicates how well it has achieved the goal. An algorithm is considered to be better if it has lesser number of master fingerprints for a rank N. The **rsfAlgo** outperforms **bozorth3** in this regard too.

We have plotted the graph for N vs the total number of master fingerprints found if we consider top N matches for each fingerprint (provided it follows FVC protocol). We have taken N from X to X+10, i.e. 4 to 10.

As can be seen from the graphs below (Fig. 4.3), both algorithms perform almost equally well, as the number of master fingerprints found are almost the same for different ranks in both databases. The number of master fingerprints increases almost linearly up to rank 12 for both algorithms, then the graph flattens out.



(a) **(b)**

Fig. 4.3 Master Fingerprints **(a)** DB1_A **(b)** DB2_A

Tabular representation

N	4	5	6	7	8	9	10	11	12	13
bozorth 3	56	124	167	205	261	311	354	385	429	454
rsfAlgo	41	99	155	215	269	311	353	393	422	450

Table 4.3 Master Fingerprints (DB1_A)

N	4	5	6	7	8	9	10	11	12	13
bozorth 3	69	153	202	257	319	371	410	443	473	494
rsfAlgo	71	150	209	261	320	370	414	454	486	519

Table 4.4 Master Fingerprints (DB2_A)

4.3.3 FAR and FRR vs Threshold

A fingerprint identification system always accepts or rejects a user based on the predetermined threshold ($\mathbf{T_o}$). It accepts only if the matching score is greater than or equal to $\mathbf{T_o}$.

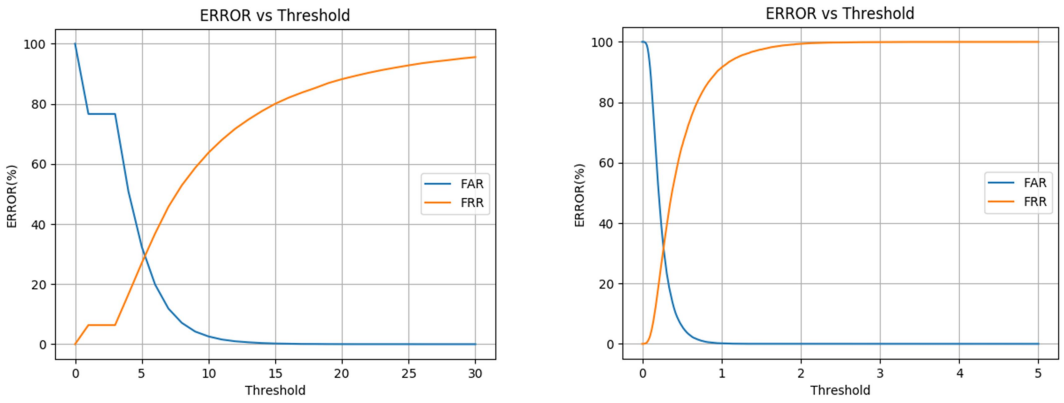
One of the main goals of a fingerprint identification system is to minimize the false acceptance rate (FAR) and false rejection rate (FRR). The point where FAR and FRR are equal is called equal error rate (EER). The overall accuracy of a fingerprint system is determined by EER.

In order to determine $\mathbf{T_o}$, we plot FAR and FRR vs threshold on the same graph. The threshold for which FAR and FRR are equal is called $\mathbf{T_o}$.

As can be seen from the graphs below (Fig. 4.4 and Fig. 4.5), with increasing threshold the FAR (False Acceptance Rate) decreases and FRR (False Rejection Rate) increases. The desired

result is to have minimum percentage for both FAR and FRR, and the threshold value where both these values are minimum (point where FAR and FRR curves intersect in graphs below) is used by any fingerprint authentication system.

The graphs obtained for rsfAlgo are smoother as compared to those obtained from bozorth3 as the score values obtained using bozorth3 are integer and because of that we used only integer values threshold, but with decimal score values of rsfAlgo fine tuning of threshold values was possible. Both algorithms perform almost similarly in both databases.



(a) **(b)**

Fig 4.4 FAR and FRR vs Threshold **(a)** DB1_A (bozorth3) **(b)** DB1_A (rsfAlgo)

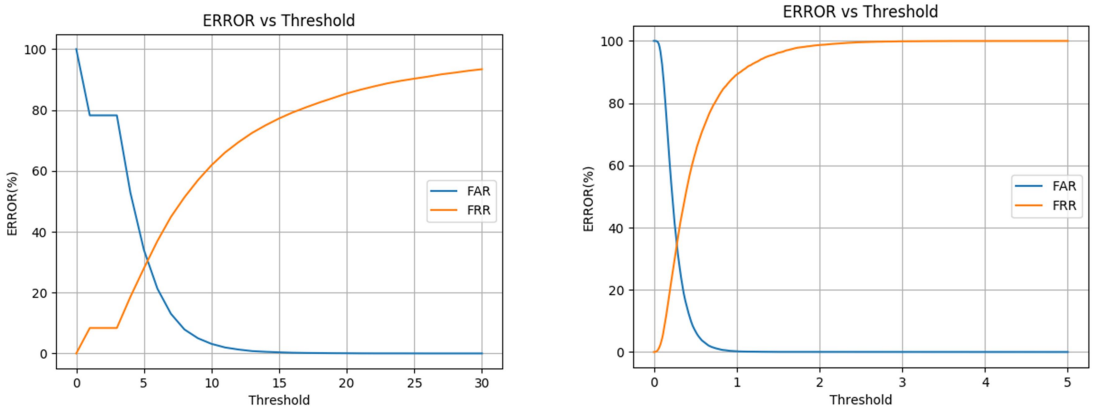


Fig 4.5 FAR and FRR vs Threshold **(a)** DB2_A (bozorth3) **(b)** DB2_A (rsfAlgo)

	DB1_A(bozorth3)	DB1_A(rsfAlgo)	DB2_A(bozorth3)	DB2_A(rsfAlgo)
EER	0.29	0.31	0.31	0.34
Threshold (T_o)	5.2	0.29	5.4	0.28

Table 4.5 EER and Threshold (T_o)

4.3.4 ROC Curve

Plot of FAR (accepted imposter attempts) on the x-axis and FRR (rejected genuine attempts) on the y-axis as a parametric function of threshold t is called ROC(t) (Receiver Operating Characteristic). EER (Equal Error Rate) can easily be found using ROC curve. Since EER is the point where FAR and FRR are equal, we can get this point just by plotting y=x line. ROC curve here indicates that bozorth3 has better accuracy than rsfAlgo i.e. bozorth3 has lower FAR and FRR than rsfAlgo.

The graphs below (Fig. 4.6) can be used to calculate the EER (Equal Error Rate) – point where FAR is equal to FRR. bozorth3 performs better as compared to rsfAlgo with less value of EER (point where y=x line intersects the curves) for both databases.

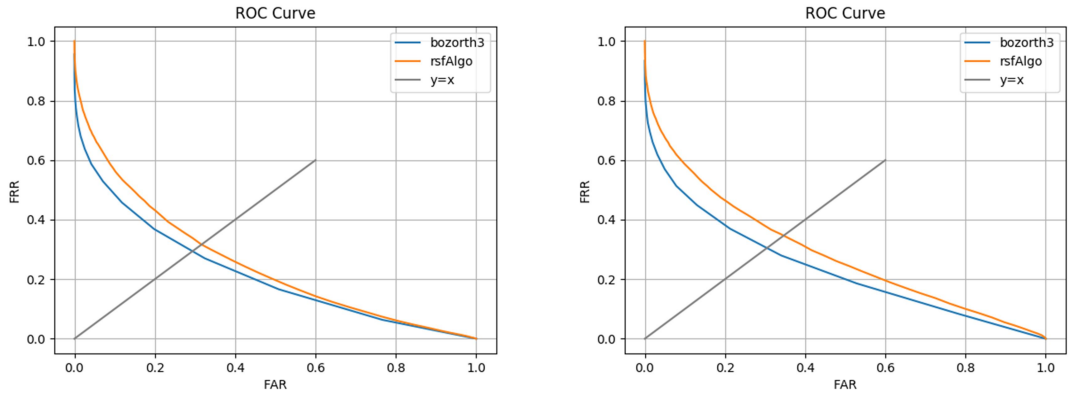


Fig 4.6 ROC Curve **(a)** DB1_A **(b)** DB2_A

CHAPTER 5
Conclusion

This project work includes the robust implementation of a partial fingerprint matching algorithm based on minutiae and ridge shape features (Lee et al. [1]) followed by a detailed performance comparison of this algorithm with bozorth3 which uses minutiae as the only feature. We used FVC2004 full fingerprint database to generate partial fingerprints of size 150x150, which were further used for performance analysis.

The algorithm based on RSF and minutiae features (rsfAlgo) gives an error rate of ~32% where as bozorth3 which uses only minutiae features gives an error rate of ~30%. Bozorth3 clearly outperforms rsfAlgo based on accuracy but rsfAlgo produces better result if we consider other parameters like minimization of master fingerprints, cumulative genuine matches, etc. The score distribution in both of these algorithms differ greatly. In case of bozorth3, we have encountered scores in the range of 0 to 200, most of the scores (>99%) falling under 50. In case of rsfAlgo, we have encountered scores in a very compact range i.e., 0 to 20 and most of them falling under 2.

Our implementation of rsfAlgo definitely does stand up to the result claimed by Lee et al. [1]. The difference in performance may be due to how partial fingerprints were generated for result analysis.

We believe that ridge shape features may be very promising in order to make a robust partial fingerprint identification system as it increases the number of extracted features immensely. The importance to RSF in Lee et al. [1] was very low in comparison with minutiae (importance ratio = 1:4). So, we believe that increasing its importance in the feature matching algorithm may increase the overall accuracy of the partial fingerprint verification systems.

References

[1] Lee, W., Cho, S., Choi, H., & Kim, J. (2017). Partial fingerprint matching using minutiae and ridge shape features for small fingerprint scanners. *Expert Systems with Applications*, *87*, 183-198.

[2] Hong, L., Wan, Y., & Jain, A. (1998). Fingerprint image enhancement: algorithm and performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 777-789.

[3] Zhao, F., & Tang, X. (2007). Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction. *Pattern Recognition*, *40*(4), 1270-1281.

[4] Chikkerur, S., Cartwright, A. N., & Govindaraju, V. (2006, January). K-plet and coupled BFS: a graph based fingerprint representation and matching algorithm. In *International Conference on Biometrics* (pp. 309-315). Springer, Berlin, Heidelberg.

[5] Zhao, Q., Zhang, D., Zhang, L., & Luo, N. (2010). High resolution partial fingerprint alignment using pore–valley descriptors. *Pattern Recognition*, *43*(3), 1050-1061.

[6] Tico, M., & Kuosmanen, P. (2003). Fingerprint matching using an orientation-based minutia descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(8), 1009-1014.

[7] Jain, A. K., Ross, A., & Prabhakar, S. (2004). An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, *14*(1).

[8] Sharma, R., Mishra, N., & Yadav, S. K. (2013). Fingerprint recognition systems and techniques: A survey. *International Journal of Scientific & Engineering Research*, *4*(6), 1670.

[9] Ko, Kenneth. *User's guide to nist biometric image software (nbis)*. No. NIST Interagency/Internal Report (NISTIR)-7392. 2007.

[10] Cappelli, R., Maio, D., Maltoni, D., Wayman, J. L., & Jain, A. K. (2005). Performance evaluation of fingerprint verification systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(1), 3-18.