Mining Rare Features in Fingerprints Using Core Points and Triplet-based Features

Indira Munagani

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Computer Engineering

Michael S. Hsiao, Chair

A. Lynn Abbott

Sandeep K. Shukla

December 9, 2013
Blacksburg, Virginia

Keywords: Fingerprints, Rare Features,
Rarity, Latent, Core Points, Triplets, GPU

Mining Rare Features in Fingerprints Using Core Points and Triplet-based Features

Indira Munagani

A fingerprint matching algorithm with a novel set of matching parameters based on core points and triangular descriptors is proposed to discover rarity in fingerprints. The algorithm uses a mathematical and statistical approach to discover rare features in fingerprints which provides scientific validation for both ten-print and latent fingerprint evidence. A feature is considered rare if it is statistically uncommon; that is, the rare feature should be unique among N (N>100) randomly sampled prints. A rare feature in a fingerprint has higher discriminatory power when it is identified in a print (latent or otherwise). In the case of latent fingerprint matching, the enhanced discriminatory power from the rare features can help in delivering a confident court judgment. In addition to mining the rare features, a parallel algorithm for fingerprint matching on GPUs is also proposed to reduce the run-time of fingerprint matching on larger databases. Results show that 1) matching algorithm is useful in eliminating false matches. 2) each of the 30 fingerprints randomly selected to mine rare features have a small set of highly distinctive statistically rare features some of whose occurrence is one in 1000 fingerprints. 3) the parallel algorithm implemented on GPUs for larger databases is around 40 times faster than the sequential algorithm.

Acknowledgments

I would like to extend my deep sense of gratitude to my advisor Dr. Michael Hsiao for giving me this opportunity to work with him. I thank him for motivating me and for being optimistic about our approach when we faced many challenges during the course of this work. I thank him for his patient guidance and his confidence in me through these 2 years. It was a continuous learning process working with him. I would like to thank Dr. Lynn Abbott for his continued interest in this work and for his valuable time in reviewing my thesis. I would also like to thank Dr. Sandeep Shukla for serving on my committee and for his vital comments about my thesis work. I would also like to thank Dr. Edward Fox for his helpful discussions initially.

I am grateful to my research team Kelson Gent, Nathan Short and Sung Hee Park for their suggestions and help with my research. I would also like to thank my lab mates Sarvesh Prabhu, Supratik Misra, Avinash Desai, Sharad Bagri, Vineeth Acharya, Shuchi Pandit and Dilip Murali for building a fun environment in lab and for their constant support.

Special thanks to Manideep Chavali and his parents for their prayers and good thoughts. I would like to thank my brother Dhiraj Munagani for helping me in making the right decisions in my career and my parents, for their prayers, love and support all through my life.

Finally, to the almighty for giving me the courage and strength to face all the challenges in my life.

I would also like to acknowledge our funding source, National Institute of Justice (Grant: 2009-DN-BX-K229).

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Motivation

Fingerprint evidence has been challenged many times in the past because of lack of scientific backing of fingerprint identification process [1]. The process of matching latent fingerprints has never been scientifically tested. There is a need for scientific validation of latent fingerprint evidence because Automated Fingerprint Identification Systems (AFIS) and human examiners face great challenges from latent fingerprints as they have very few details and it is difficult to make a court judgment about identification or exclusion of the evidence. In the jury, a latent fingerprint with some kind of distinctiveness proves to be highly distinguishing and will provide significant confidence to the evidence.

Friction ridge patterns on human fingers are considered to be unique to every individual [2, 3, 4], but there has been no proven mathematical or statistical approach to quantify the rarity of these friction ridge patterns. In this work, the individuality of fingerprints is explored by finding statistically rare features with high discriminatory power that are obtained after comparisons with thousands of fingerprints. The goal to identify individuality in fingerprints is further motivated by the need expressed by National Institute of Justice (NIJ) in [5, 6] and by the National Academy of Sciences (NAS) in [7] for validating the science behind fingerprint identification methods.

The existing fingerprint triplet-based matching techniques take a significant amount of time when they are run sequentially on a Central Processing Unit (CPU) [8, 9]. When a large database of fingerprints is considered, the matching can take up to several days. The use of multi-triangular features which are formed by combining multiple triples in a fingerprint showed that certain distinctive features can be extracted from fingerprints [9]. Each fingerprint has millions of multi-triangular features and these further increases the total run-time because, as the number of the multi-triangular features increases in a fingerprint, the number of computations and

comparisons required increases and hence the run-time of the algorithm increases. There is a need to reduce the run-time and GPGPUs (General Purpose computing on Graphics Processing Units) provide the ideal platform for parallelizing triplet-based feature matching techniques and help in reducing the time taken. The feature matching technique [8] is easily parallelizable given that the same matching criteria are applied for all the features. This matching technique is similar to SIMD (Single Instruction Multiple Data) where the same matching technique is applied for all the data. Also, GPUs are suitable to applications that are computationally intensive and operate on large data sets. The thousands of triangular features and millions of multi-triangular features can be compared simultaneously on multiple threads of a GPU and all the computations related to matching can be performed on the GPU.

1.2 Contributions of this Thesis

Although the main focus of this thesis work is mining rare features in fingerprints, the work initially started with parallelizing an existing triplet-based feature matching algorithm and mining rare features by using the same GPU parallel algorithm. The tolerance levels of some of the matching parameters used for parallelization were stringent and it was later realized that these tolerance levels needed to be changed. The work related to fingerprint matching on GPUs is retained in this thesis because it shows methods and optimization techniques which can accelerate the algorithm. Though triangular features are used in a hierarchical fashion in this thesis (3-points, 6-points and 9-points), most of the triplet-based matching algorithms use only triangles (3-points) for fingerprint matching [10, 11, 12]. The GPU parallel algorithm technique proposed for 3-point comparisons can very well be used for fingerprint matching (using different tolerances) and equivalent speedups can be achieved. The method used for 6-point and 9-point comparisons on GPUs can also be adopted for future implementations. The contributions of this thesis are as follows:

- Parallelizing fingerprint triplet-based feature matching on GPUs.
- Detailed discussion of distortion in different impressions of the same finger.
- Development of a novel fingerprint triplet-based matching algorithm using core points for mining statistically rare features in a fingerprint to aid in latent fingerprint matching.

- Identification of some rare features in 30 fingerprints that are randomly selected from a database of 11,036 fingerprints.

1.3 Outline

Chapter 2 lays the background for this thesis. Triplet based feature matching techniques and previous works on rarity in fingerprints are discussed. Since this work involves usage of GPUs for fingerprint feature matching, NVIDIA GPU architecture and CUDA (the parallel programming and computing platform) are briefly discussed.

Chapter 3 discusses the GPU implementation of triplet based feature matching algorithms on two different databases. Some optimizations and usage of shared memory which enhanced the GPU speedup are also discussed.

Chapter 4 discusses the matching algorithm using core points for finding statistically rare features in fingerprints. The different kinds of distortion in fingerprints are also discussed.

Chapter 5 concludes the thesis work and proposes some future work.

Chapter 2: Background

2.1 Fingerprint Representation

Fingerprints are characterized and defined by their unique ridges and valleys (see Figure 2.1). Ridge details in a fingerprint are described in a hierarchical fashion – Level 1: Overall global ridge flow pattern, Level 2: Minutia points, Level 3: Pores, Local shape of ridges. Fingerprint ridges usually form distinctive shapes in certain regions called singular regions or singularities broadly classified into loop, whorl and delta. Minutiae are the smaller details of the fingerprints where a ridge comes to an end, called ridge ending or termination, or a ridge divides into two, called ridge bifurcation (see Figure 2.2). The minutiae are usually represented by x- and y-coordinates and the angle made by the tangent to the ridge with the horizontal axis as shown in Figure 2.3. Minutiae are the most commonly used features in automated fingerprint matching. At Level 3, ridge attributes such as shape, width, edge contour, sweat pores, incipient ridges, scars and creases can be extracted. Though level 3 features are highly distinctive, their reliable detection requires high resolution scanners.

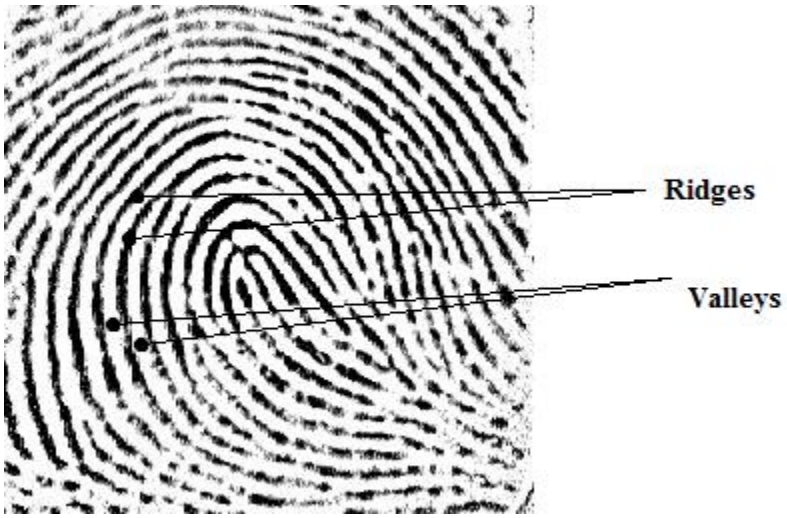

Figure 2.1 Ridges and valleys in a fingerprint

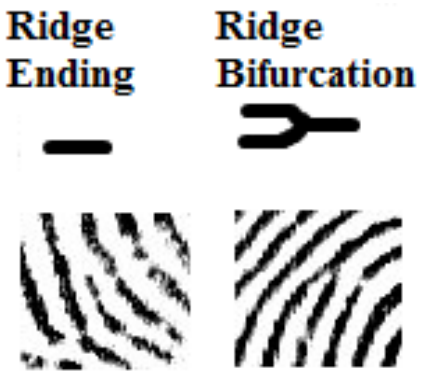

Figure 2.2 Minutiae ridge ending and ridge bifurcation

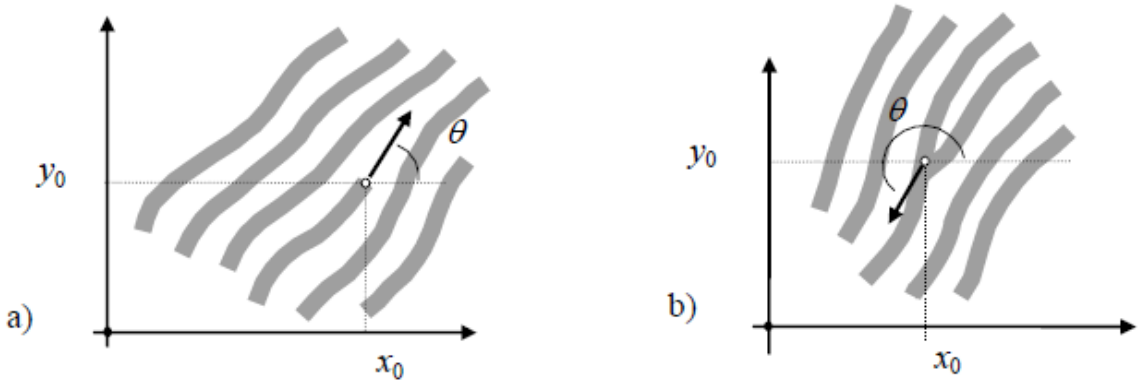

Figure 2.3 a) a ridge ending minutia: [x_0, y_0] are the minutia coordinates; θ is the angle that the minutia tangent forms with the horizontal axis; b) a bifurcation minutia: θ is now defined by means of the ridge ending minutia corresponding to the original bifurcation that exists in the negative image. [13] (*from D. Maltoni, D. Maio, A. K. Jain and S. Prabhakar, Handbook of Fingerprint Recognition (2nd Edition), Springer, 2009*) (*Used under fair use*)

2.2 Latent Fingerprints

Latent fingerprints are the impressions which are accidentally left on any surface and they may not be obvious to the naked eye, but are extracted through more refined techniques. They are usually partial fingerprints with limited details (fewer minutiae) and pose a great challenge for fingerprint matching. One such latent fingerprint is shown in Figure 2.4. These prints are deposited because of the sweat produced from the fingers that adheres to the friction ridges of the

finger and when the finger is placed on any surface such as glass, table and wall, an impression is left behind. To expose these latent impressions, fingerprint technicians use powder dusting, ninhydrin spraying, iodine fuming and silver nitrate soaking [14]. Better procedures [15] have been developed based on chemical reagents and systemic approaches to expose and lift the latent fingerprints when they are deposited on wet surfaces and untreated wood. [13].

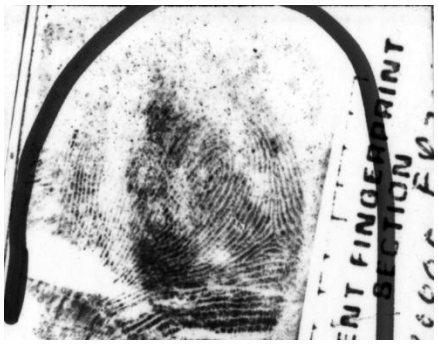

Figure 2.4 Latent fingerprint

2.3 Fingerprint Classification

As the number of enrolled users increases in a fingerprint identification system, more comparisons are required to identify a fingerprint match. Hence, the time required for performing these comparisons also increases. Decreasing the number of comparisons required can reduce the time taken. In order to achieve this, the fingerprints in the database are partitioned into subsets. Typically fingerprints can be divided into five classes based on global ridge patterns –right loop, left loop, whorl, arch and tented arch [13]. These five classes of fingerprints are shown in Figure 2.5. Some parts of the ridge patterns of a fingerprint form semicircles or closed loops known as core points. Some parts form a triangular pattern called as a delta point. The core regions are shown by blue-colored dots and delta regions are shown by orange-colored dots in Figure 2.5. The fingerprints with a right loop or a left loop have a single core point and a single delta point. A whorl has two core points and two delta points. A tented arch has a single delta point. An arch has no core points and no delta points.

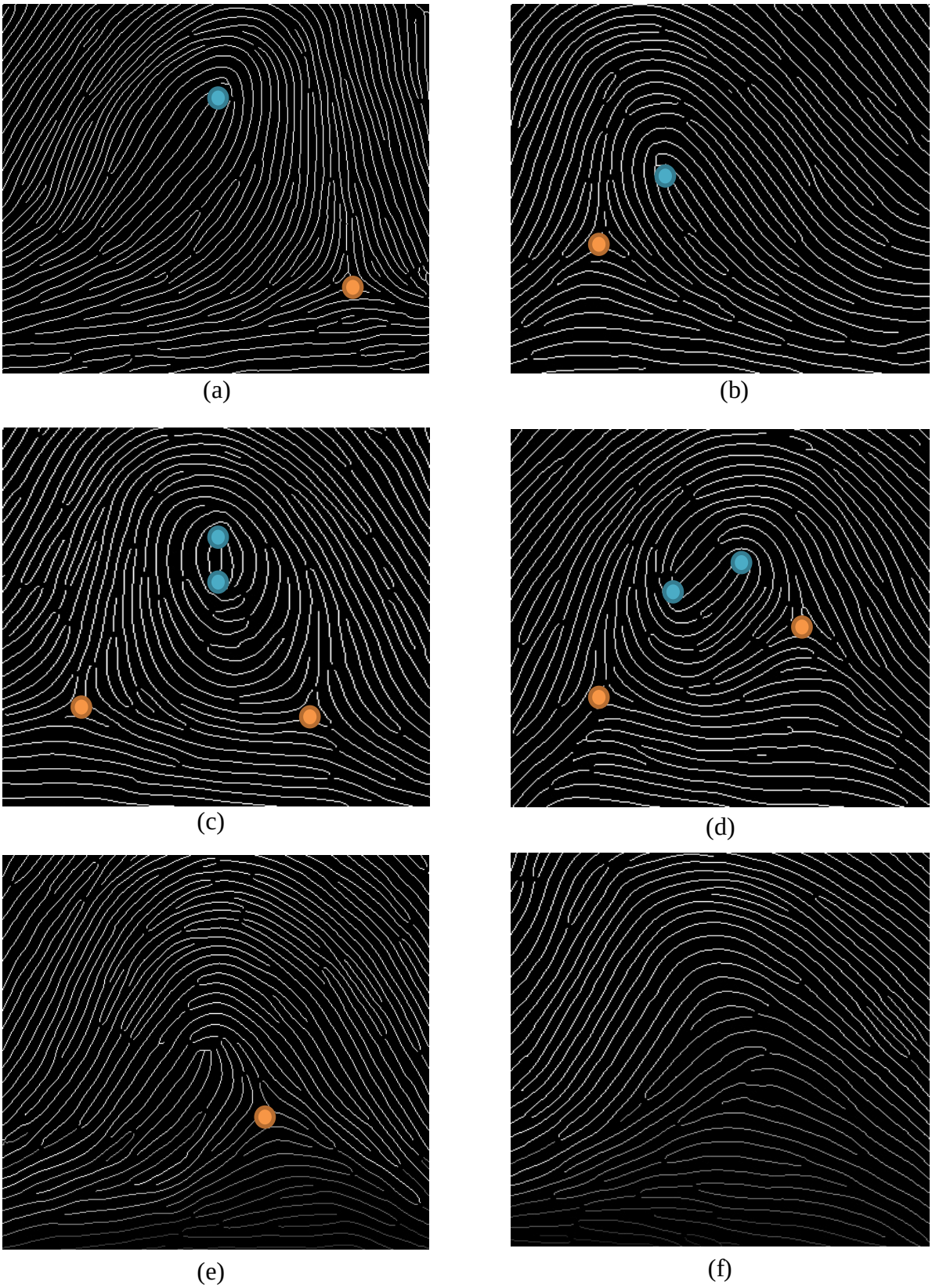

Figure 2.5 Classification of fingerprints (a) right loop (b) left loop (c) whorl (d) two loops (e) tented arch (f) arch.

The distribution of the five classes of fingerprints is non-uniform in a typical database. Wilson et al. considered more than 222 million prints and found that the proportions of fingerprints in arch, tented arch, left loop, right loop and whorl is 3.7%, 2.9%, 33.8%, 31.7% and 27.9%, respectively [16]. In the FBI database that is used for this work, the percentage of tented arch and arch together, left loop and right loop together and whorl are 5.3%, 75.3% and 19.4%, respectively. Hence, by classifying the fingerprints into these classes, the number of fingerprints to be used for identification purposes drastically decreases. Singular points, when reliably detected, also help in reducing the fingerprint verification time [17]. In this thesis work, singular points are used for fingerprint matching.

2.4 Previous Work in Triplet-Based Fingerprint Matching

Fingerprint matching can be classified into correlation-based matching, minutiae-based matching and non-minutiae feature-based matching. In correlation-based techniques, the two fingerprint images to be compared are aligned over one another and the amount of correlation is calculated. The correlation is computed locally over certain regions of interest in the fingerprint to overcome non-linear distortion.

Non-minutiae based feature matching is mostly used in the cases where the area of the fingerprint is small with 4-5 minutiae in it. In these cases, minutiae based matching cannot be used. For non-minutiae based feature matching, the most commonly used features are number, type and position of core and delta points, global and local texture information, geometrical attributes and spatial relationship of ridge lines (length and curvature of ridge lines is compared) and level 3 features. These techniques are often used in addition to minutiae based techniques to improve accuracy.

Minutia based matching is well known and used widely. Fingerprint is represented as a feature vector with minutiae as its elements. Each minutia is further described by many characteristics such as location, type, quality and orientation. Typically, minutiae are represented as $m = (x, y, \theta)$ where (x, y) is the location of minutia in the fingerprint and θ represents the direction of the minutia. A reference fingerprint r and a target fingerprint t can be represented as shown below.

$$\mathbf{r} = \{\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3 \dots \mathbf{m}_m\}, \qquad \mathbf{m}_i = \{x_i, y_i, \theta_i\}, \qquad i = 1 \dots m$$

$$\mathbf{t} = \{\mathbf{m}'_1, \mathbf{m}'_2, \mathbf{m}'_3 \dots \mathbf{m}'_n\}, \qquad \mathbf{m}'_j = \{x'_j, y'_j, \theta'_j\}, \qquad j = 1 \dots n$$

The reference and target fingerprints are considered a match when the spatial distance between the minutia points is smaller than a tolerance r_0 and the direction difference between them is smaller than a tolerance θ_0. The tolerances are considered to compensate for the feature extraction errors and the plastic distortion that cause the minutiae positions to change. The fingerprints need to be aligned to maximize the number of matching minutiae. This alignment requires displacement and rotation of the fingerprints. Scale should also be considered when the fingerprints are taken at different resolutions. Also distortion tolerant geometric transformations are considered while matching fingerprints.

Alignment is the most important step for minutia based matching. A few matching segments of pairs of minutiae between both the fingerprints are identified and the alignment parameters (displacement, rotation) are derived based on them. The fingerprints are aligned and the remaining minutiae in the fingerprints are compared within some tolerance levels. The best alignments are compared for consistency. The maximum number of mated pairs, fraction of mutually consistent alignments, minutiae direction and ridge counts determine the final score. This method is computationally very expensive since it works in an iterative fashion. Many algorithms have been proposed for fingerprint alignment that use singularity- based (core point and delta point) pre-alignment in which the singularities are superimposed and ridge based pre-alignment in which the length and orientation of ridges is compared.

To avoid the need to perform alignment, global and local minutiae matching algorithms have been proposed that use the global spatial relationships and local features which are translation and rotation invariant. One method using global minutiae matching involves using an intrinsic coordinate system (ICS) which runs along hypothetical axes defined according to the ridge orientation. Local minutiae matching involve use of local structures which are invariant to global transformations such as translation and rotation. However, by using only local minutiae for fingerprint matching, the global spatial relationships which have high distinctive power are eliminated and therefore reduce the information for discriminating fingerprints. Some of the earliest approaches of local minutiae matching involved use of local structures which are defined

by the number of minutiae falling inside some regions followed by some local structures defined by the relationship between a minutiae and its neighboring minutiae invariant to translation and rotation of fingerprints.

Jiang et al. used local structures that are formed by a central minutiae and its two nearest neighboring minutiae (see Figure 2.6) [18]; the feature vector v_i for the central minutia m_i whose closest minutiae are m_j and m_k is defined by the following.

$$v_i = \{d_{ij}, d_{ik}, \theta_{ij}, \theta_{ik}, \varphi_{ij}, \varphi_{ik}, n_{ij}, n_{ik}, t_i, t_j, t_k\}$$

d_{ij} is the distance between minutiae m_i and m_j, θ_{ij} is the direction difference between the angles θ_i and θ_j of minutiae m_i and m_j, φ_{ij} is the direction difference between angle θ_i of minutia m_i and the edge joining m_i and m_j, n_{ij} is the number of ridge counts between minutiae m_i and m_j, t_i is the type of minutia m_i.

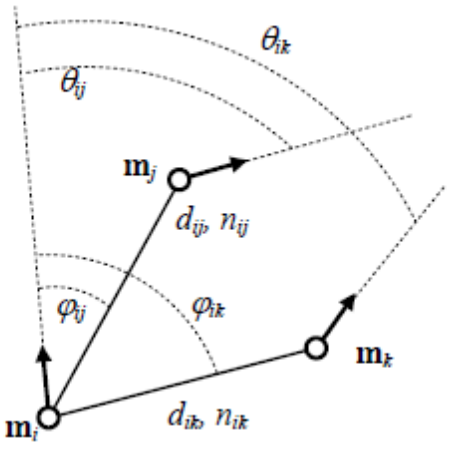

Figure 2.6 Features of local structures used by Jiang and Yau [13] (*from D. Maltoni, D. Maio, A. K. Jain and S. Prabhakar, Handbook of Fingerprint Recognition (2nd Edition), Springer, 2009*) (*Used under fair use*)

Ratha et al. used graph notation, a star in which all the minutiae around a central minutia whose distance is smaller than d_{max} are the vertices in the graph and the edges are formed by joining the central minutia with the remaining minutiae in the graph [19]. Each star in the input fingerprint is compared against every star in the template.

The matching of local structures invariant to translation and rotation forms the basis for most of the fingerprint matching techniques proposed after year 2000 [13]. Minutiae triangles or triplets have been first used by Germain et al. for fingerprint indexing [20]. Later, Bhanu et al. used minutia triangles for fingerprint indexing in which they chose angles, triangle handedness, triangle type, triangle direction and maximum side length as the parameters [21]. Kovacs-Vajna et al. and Parziale et al. also used minutiae triangles for fingerprint matching [12, 22]. Kovacs-Vajna et al. used only those triangles which are formed by Delaunay triangulation (minutiae points forming triangles are selected such that no point lies inside the circumference of any triangle) to cope with the non-linear deformations in fingerprint. By using Delaunay triangulation, triangles with smaller angles are skipped. They showed that distortion in local areas can be more easily controlled than global deformations. Jea et al. used minutia triangles for partial fingerprint matching [23, 24]. Their triangle selection and matching procedure is similar to that of Jiang and Yau [18]. However, minutia type and ridge counts are removed in their method because minutia type is difficult to distinguish between fingerprints of the same finger and ridge counts are not extracted by all the feature extracting tools.

Tan et al. and Ghazvini et al. used minutiae triangles for fingerprint verification using genetic algorithms [11, 25]. Tan et al. used a fitness function that checks the global consistency of minutiae followed by minutiae triangles to verify detailed matching. Chen et al., Zheng et al. and Feng et al. also used minutiae triangles for fingerprint matching [26, 27, 28] where they used different matching parameters to overcome the non-linear distortion in fingerprints. The most commonly used matching parameters in the above methods are spatial distances between minutiae, type, angles, direction, ridge counts between minutiae. Some methods have avoided use of minutiae type because it is unreliable and also spatial distances which are highly distorted. Medina-Perez et al. proposed an algorithm called M3g1 for triplet based matching which discards some of the unmatched triplets (which are typically matched by other algorithms) in the initial stage and used direction and side lengths as the basic matching criteria [10]. This thesis work is similar to the methods used by Hoyle et al. and Xu et al. in which minutiae triangles are used in a hierarchical fashion forming 2-triangle and 3-triangle combinations [9, 29].

2.5 Previous Work on Rarity in Fingerprints

Hoyle et al. mined for distinctive features fingerprints and they discovered a set of 10 rare features in a database of 93 fingerprints [9]. Rare features are the features whose occurrence is low in a database. Some of the parameters that they used in mining for rare features are intra-shared ridge segments, inter-shared ridge segment, intra ridge counts, total ridge counts, side lengths to ridge count ratio and vice-versa. Intra-shared ridge segments refer to minutiae in a triplet that share the same ridge and inter-shared ridge segment refers to minutiae in a 2-triangle or 3-triangle combination that share the same ridge. Hoyle et al. searched for rare features by fine-tuning their parameters such that a single feature is returned after every search. This thesis work is similar to the work of Hoyle et al. in that the methods used in this thesis also search for statistically rare features in a fingerprint by using single, two and three triangle combinations, however the search is performed using different matching parameters that use singular points and predefined search parameters are not used while mining for rare features. Also minutia type is not considered because of the ambiguity in ridge endings and bifurcations [13]. Identifying minutiae that share the same ridge segment also depends largely on minutiae type. When a ridge bifurcation becomes a ridge ending because of finger pressure variations, the minutiae that shared the same ridge (when the minutia was a bifurcation) before do not share the same ridge anymore. Hence, inter and intra shared ridge segments are not used in this work. The search for rare features in this work returns a set of unmatched triangles for each fingerprint after comparing with a larger database. By using a larger database, rare features that are truly statistically rare can be identified because a feature that is rare in 100 fingerprints may find a match in 1000 fingerprints. So, the confidence of rare features obtained by using larger databases is higher.

2.6 NVIDIA GPU

GPUs are used widely in applications that are computationally intensive because they can distribute the work among several cores. GPUs are well suited for applications that involve data

parallel computations where the arithmetic intensity is much higher than memory operations. The latency for memory operations is hidden on a GPU with arithmetic calculations.

GPUs have already been used in various biometric applications such as Iris Matching, Face recognition and fingerprint matching [30, 31, 32]. Fingerprint matching becomes challenging especially when large databases are considered because of the longer run-times. There are some existing techniques that use GPUs to accelerate fingerprint matching algorithms. Gutierrez et al. used GPUs for fingerprint matching on large databases [32]. They consider a 3-dimensional structure called cylinder associated with each minutia which stores the contributions of each of the neighboring minutiae based on their location and direction. They use only location and orientation of the minutiae as the matching criteria.

GPUs have always been used for graphics applications since their invention, but in the recent years, they are widely used for accelerating scientific applications marking the advent of GPGPUs, or General Purpose computation on GPU.

Many real-world applications can be parallelized on a GPU and can run significantly much faster than on a CPU [33]. GPUs are used in a wide variety of applications such as computation chemistry and biology, bioinformatics, defense and intelligence, computational finance, electronic design automation, computer aided design, computer vision, video processing, scientific computing to name a few [34].
A CPU consists of a fewer cores while a GPU consists of thousands of smaller and efficient cores designed for parallel computing. A combination of CPU and GPU are generally used where the serial portion of the application runs on the CPU and the parallel portion of the application runs on the GPU. The data is transferred between CPU and GPU.

Fingerprint matching is both data intensive and computation intensive application. In this thesis work, GPUs are used for fingerprint matching on larger databases by using triangular features. All the comparisons which involve computations are performed on the GPU.

2.7 CUDA

CUDA stands for “Compute Unified Device Architecture”. CUDA is a general purpose computing platform developed by NVIDIA for programming on GPUs. CUDA has extensions in C and C++ which can be used for parallel programming. CUDA can also be used with other high level languages such as FORTRAN. Figure 2.7 shows a code snippet of a saxpy program on CPU and GPU. The parallel code for this saxpy program is written in CUDA.

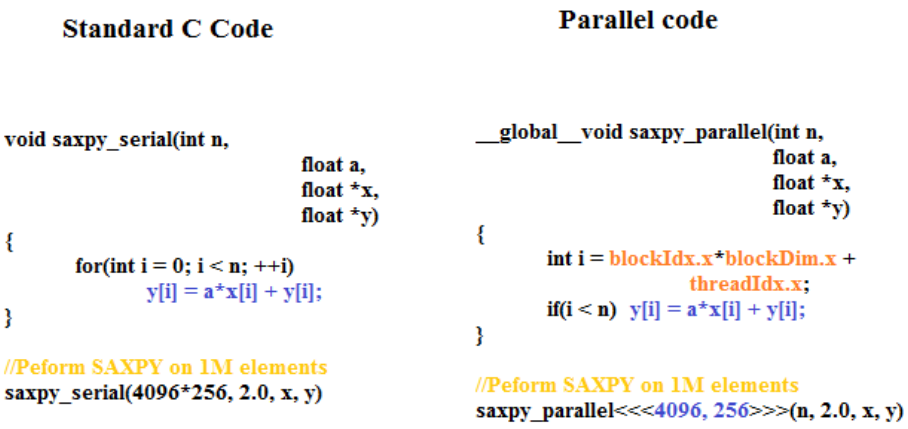

Figure 2.7 Comparison of code between CPU and GPU [33] (*from NVIDIA, "GPU Computing," [Online]. Available: www.nvidia.com*) (*Used under fair use*)

In this work, the GeForce GTX 285 device is used. The GeForce GTX 200 GPUs consist of 10 Thread Processing clusters (TPC). Each TPC consists of 3 Streaming Multiprocessor (SM) and each SM in turn is made up of 8 Streaming Processors or Thread Processors. So, the total number of cuda cores or processor cores is 10*3*8 = 240 as shown in Table 2.1 [35].

Table 2.1 Number of GPU processing cores (*from NVIDIA, "GeForce GTX 200 GPU Technical Brief," 2008*) (Used under fair use)

Chip	TPCs	SMs per TPC	SPs per SM	Total SPs
GeForce GTX 200 GPUs	10	3	8	240

In Figure 2.8, the architecture of GeForce GTX 200 GPUs [35] is shown. Each SM consists of shared memory and registers. All the SMs access the same local and global memory.

While programming in CUDA, terms like device, host and kernel are used extensively. A device is nothing but the GPU, CPU is the host and kernel is the function that runs on the device. The kernel consists of the parallel portion of the application. The threads run in blocks of a grid. A

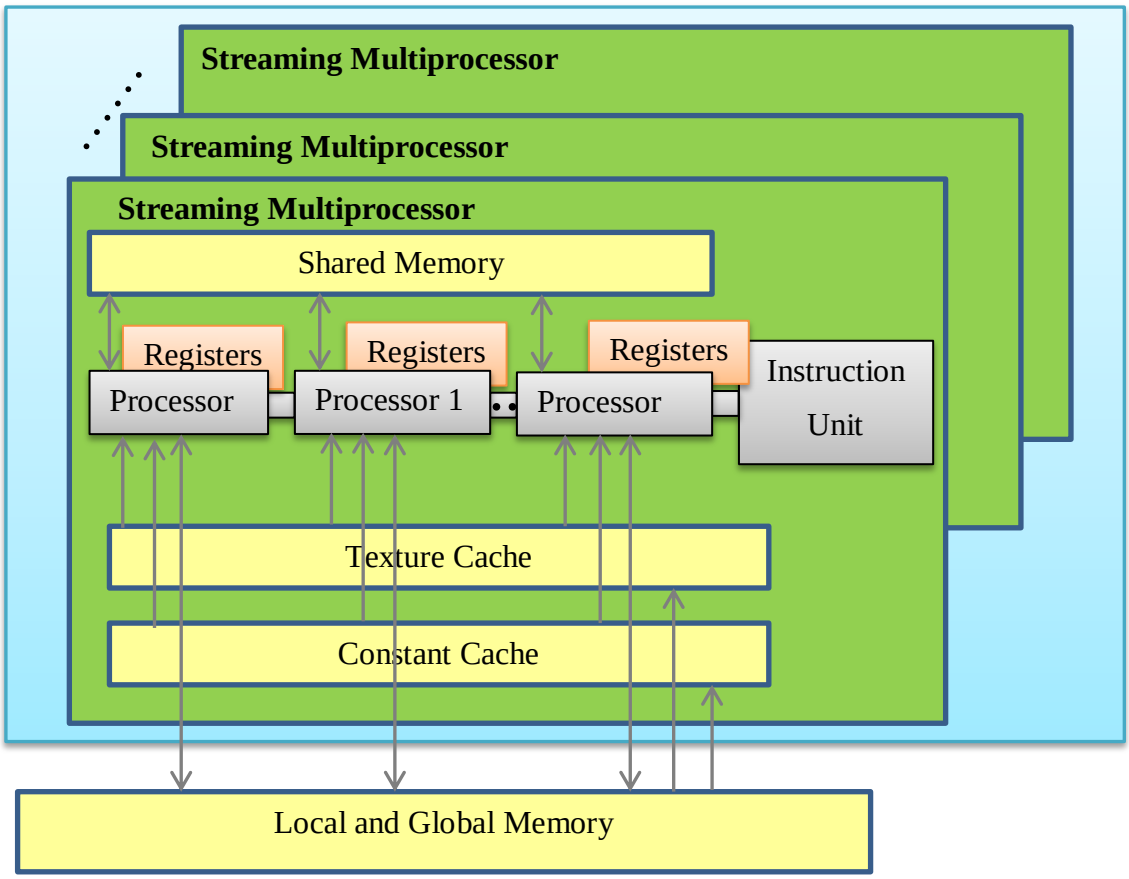

Figure 2.8 NVIDIA GT200 architecture [36] (*NVIDIA, CUDA Programming Guide*) (*Used under fair use*)

grid is a 3-dimensional structure though the third dimension is always 1. A grid consists of blocks distributed in a 2-dimensional manner. Each block consists of several threads. In Figure 2.9, a grid of 6 blocks with 2 rows and 3 columns is shown and Block (1, 1) consists of threads with 3 rows and 4 columns [36].

There is a limit on number of threads per block, number of blocks per grid. For GTX 285, Figure 2.10 shows the maximum number of threads and blocks. It also shows some more device

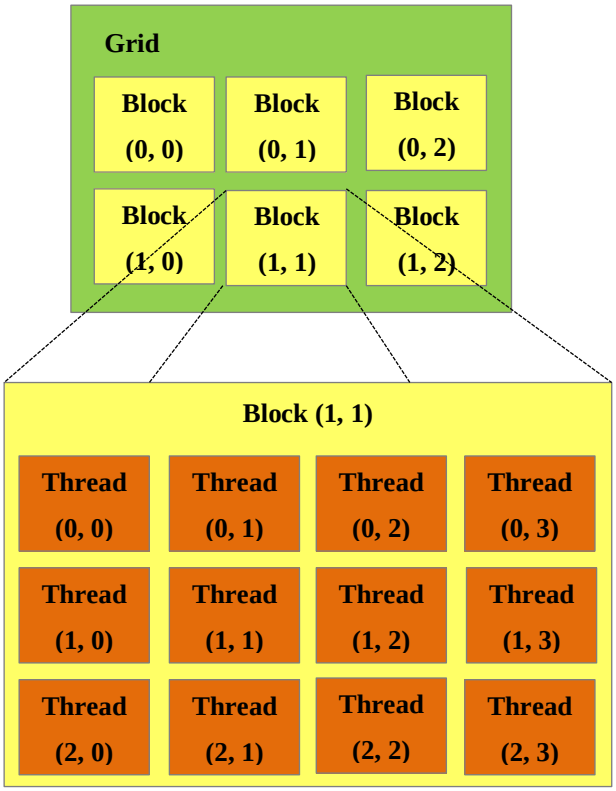

Figure 2.9 Grid of thread blocks (*NVIDIA, CUDA Programming Guide*) (*Used under fair use*)

properties which will be discussed subsequently.

When a grid is launched, the blocks are enumerated and distributed to the multiprocessors and threads in a block run concurrently on one multiprocessor. It is also possible for a multiprocessor to run more than one block at a time if there is enough number of hardware resources (shared memory and registers) on it to run multiple blocks. In a multiprocessor, threads are executed in groups of 32 parallel threads called warps. When a block is assigned to a multiprocessor, the threads in that block are divided into warps and these warps get scheduled by a warp scheduler.

Memory Hierarchy: Each thread has some local memory associated with it. Each block has 16KB of shared memory accessible to all the threads in that block. All the blocks have access to the same global memory. Memory accesses from global memory are much slower than those from shared memory. Threads in a block can co-operate with each other by using shared

memory. If all the threads in a block use the same data, this data can be transferred to shared memory. After the data is transferred, computations are performed in each thread. Some threads may finish copying data to shared memory faster than the other threads. To ensure that all the threads in a block finish copying data to shared memory, the function __syncthreads () is used.

```
Device name GeForce GTX 285
Compute capability 1 3
Clock rate 1476000
Total Global Memory 1073020928
Total Constant Memory 65536
Multiprocessor count 30
Shared memory per Block 16384
Registers per Block 16384
Warp Size 32
Max threads per block 512
Max thread Dimensions 512 512 64
Max Grid Dimensions 65535 65535 1
```

Figure 2.10 GTX 285 properties

2.8 Terminology

This section defines some of the terms used extensively in this thesis.

3-point feature or triplet or triangle combination: A 3 point feature or a triplet is formed by joining any 3 minutia in a fingerprint. A fingerprint containing 50 minutiae has ${}^{50}C_3 = 19600$ 3-point features.

6-point feature or two-triangle combination: A 6-point feature is formed by joining any 6 minutia in a fingerprint. It can also be formed by combining any two 3-point features, but all the 6 minutia points in the two 3-point features should be different. A fingerprint containing 50 minutiae has ${}^{50}C_6 = 15.89$ million 6-point features

9-point feature or three-triangle combination: A 9 point feature is formed by joining any 9 minutiae in a fingerprint. It can also be formed by combining three 3-point features. It can also be formed by joining a 6-point feature with a 3-point feature with all the minutia points different from each other. A fingerprint containing 50 minutiae has ${}^{50}C_9 = 2.50$ billion 9-point features.

Ridge count or Ridges crossed: The number of the ridges crossed between any two minutiae points is the ridge count. In Figure 2.11, an example of ridges crossed between two minutiae points is shown.

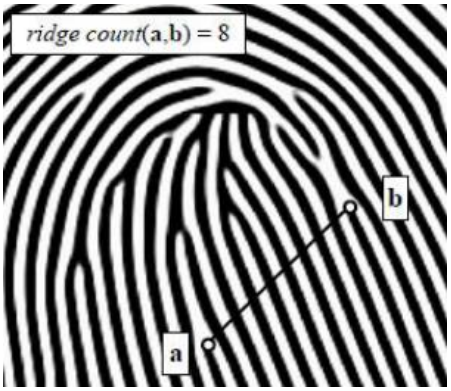

Figure 2.11 In this example, number of ridges crossed between a and b is 8. [13] (*from D. Maltoni, D. Maio, A. K. Jain and S. Prabhakar, Handbook of Fingerprint Recognition (2nd Edition), Springer, 2009*) (*Used under fair use*)

Chapter 3: Fingerprint Feature Matching Using GPUs

3.1 Need for GPUs

Each fingerprint has as many as thousands of 3-point features and millions of 6-point and 9-point features. On a CPU, every feature of the fingerprint is matched in a sequential way. The matching of millions of features takes significant amount of time. Also, when thousands of fingerprints are compared, the computational time increases further. On a GPU, because of multiple smaller and efficient cores, multiple threads can run in parallel and thereby multiple feature comparisons can be performed in parallel.

The rest of the chapter discusses GPU comparisons on two databases. The basic idea of implementation is similar for both the databases. For the second database, some optimizations and shared memory are used which reduced the time taken to a great extent.

3.2 Databases

The databases used in this work are FVC2000 DB1 (Fingerprint Verification Competition 2000 Database 1) [37] with 93 fingerprints, and an FBI database with 74,140 fingerprints. Fingerprint matching of FVC2000 DB1 database is discussed first, and later the FBI database fingerprint matching is discussed.

3.3 GPU Comparisons of FVC2000 DB1 Fingerprints

3.3.1 Input Files

The FVC2000 fingerprints were processed to get text files called minutia files with extension .min. The software used to extract the features is NIST Biometric Image Software (NBIS) developed by National Institute of Standards and Technology (NIST). These minutia files are

augmented by ridge ID numbers for each minutia by our internally developed tool (by Nathan Short, Virginia Tech). These ridge ID numbers are the IDs of the ridges forming the minutia points. These minutia files contain the coordinates of minutia points, the direction of the minutia point given by angle (theta), the quality of minutia, the type of minutia and the ridges of the minutia point. The type of minutia is either a ridge ending or a ridge bifurcation. For a ridge ending, the number of ridges is 1 whereas for a ridge bifurcation, the number of ridges is 3. The quality range for these minutia points is 0-4. A sample minutia file is shown in Figure 3.1.For example, the first row in Figure 3.1 shows a minutia point with x-coordinate 56, y-coordinate 136, angle of 0.1717 radians, quality of 4, minutiae type 1(ridge ending) and formed by a ridge whose ridge ID is 21. Some of the minutiae in this figure whose quality is <3 have a type 0. These minutiae are false minutiae and they are not considered. Minutiae with type 2 are ridge bifurcations. All the minutia points which have quality greater than or equal to 3 are considered.

X-coordinate	Y-coordinate	Angle	Quality	Type	Ridge ID 1	Ridge ID 2	Ridge ID 3
56	136	0.1717	4	1	21		
284	216	0.35564	2	0	22		
16	197	3.12586	2	0	22		
186	125	1.20723	4	1	23		
115	121	2.20939	4	1	25		
283	275	0.60209	2	0	24		
283	259	0.74768	2	0	26	23	25
259	270	0.86077	3	1	27		
46	134	2.2026	3	2	29	20	28

Figure 3.1 Sample minutia file

Also, other data that gives the number of ridges crossed between any two minutia points in a fingerprint is used. This data is extracted into ridge count files. Figure 3.2 shows an example of one such file. These files have the coordinates of the two minutia points and the ridge count between them. For instance, in Figure 3.2, (216, 36) and (111, 31) are coordinates of 2 minutia points in a fingerprint and the ridges crossed between them is 3.

```
216 36 111 31 3
216 36 219 31 0
216 36 211 56 2
216 36 210 42 2
216 36 75 68 9
216 36 213 52 1
216 36 242 75 2
216 36 77 87 10
216 36 247 76 3
216 36 204 97 6
216 36 36 130 13
216 36 71 83 11
216 36 72 92 9
216 36 29 101 11
216 36 31 147 17
216 36 56 136 16
216 36 186 125 9
```

Figure 3.2 Sample ridge count file

Comparisons on GPU: The fingerprint feature matching is performed in a step by step manner by comparing one fingerprint at a time with the reference fingerprint. In this work, 3 point, 6 point and 9 point comparisons are made on GPU. Firstly, 3 point comparisons are made on GPU, then the results from 3 point comparisons are further used for 6 point comparisons whose results are used for 9 point comparisons as shown in Figure 3.3.

3.3.2 Single Triangle (3-point) Comparisons on the GPU

A triple is formed by a combination of any 3 minutiae points and thousands of such triples exist in each fingerprint. On a CPU, the matching can be performed sequentially by comparing a single triangle from the reference fingerprint with all the triples from the target fingerprint. In contrast, in this implementation on GPU, in each thread, 32 such comparisons are made in parallel and the result is stored in a 32-bit integer value. Memory is allocated for reference and target triples on GPU and this data is transferred onto GPU using CUDA APIs. A two dimensional grid as illustrated in Figure 3.4 is launched to perform comparisons on GPU.

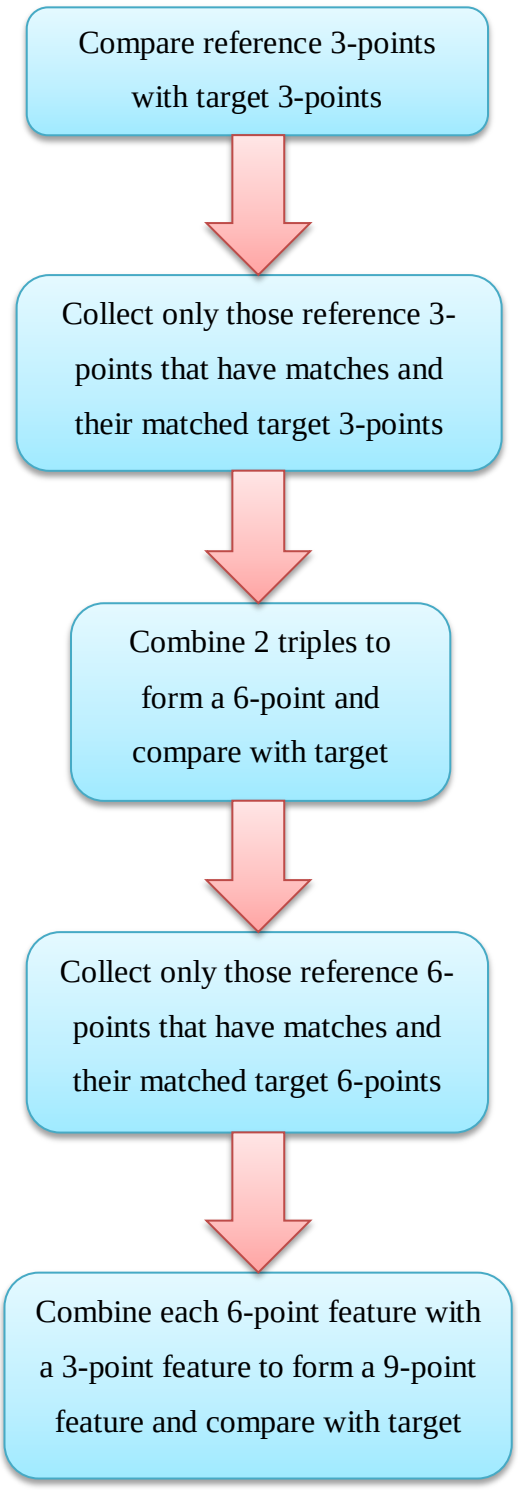

Figure 3.3 GPU comparisons flowchart

The number of blocks is calculated based on the number of reference and target triples. Each block consists of 256 threads. In each row of the grid, 256 triples of the reference fingerprint are

compared with all the triples from the target fingerprint. In the first row, the first 256 triples are compared with target triples and in the second row, the next 256 triples are compared. In the last row of the grid, the last 256 triples of the reference fingerprint are compared.

The number of rows is given by the following equation.

$$Number\ of\ rows = \frac{Number\ of\ reference\ triples}{Number\ of\ target\ triples}$$

In Figure 3.4, 5 rows of blocks are shown. In the first row, triples 0-255 from reference fingerprint are used for comparisons. In the second row, triples 256-511 are used and the last row i.e. the fifth row, triples 1024-1279 are used. In each thread of the block, a reference triple is compared.

In each block, 256 triples from the reference fingerprint are compared with 32 triples from the target fingerprint. In the first column, 256 reference triples are compared with 0-31 target triples. In the second column, 32-63 target triples are compared. The block structure is illustrated in Figure 3.5.

T0, T1, T2 … T255 are the threads in Block (0, 0). In thread T0, reference triple 0 is compared with 0-31 target triples. In thread T1, reference triple 1 is compared with 0-31 target triples. Similarly, for Block (1, 0) which is in the next row of Block (0, 0), in thread T0, reference triple 256 is compared with 0-31 target triples and in thread T1, reference triple 257 is compared with 0-31 target triples. In Block (0, 1) which is in the same row as Block (0, 0), in thread T0, reference triple 0 is compared with 32-63 target triples and in thread T1, reference triple 1 is compared with 32-63 target triples.

Matching Criteria: The side lengths of the triples, the sum of angles, the sum of number of ends and bifurcations, the sum of number of ends, the sum of number of bifurcations are compared between reference and target triples. Reference triples with largest distance > (150) and smallest distance < 10 are not considered in comparisons. Local structures have lesser distortion [22], hence only those triples whose side lengths are ≤150 are selected. Also the smallest distance between minutiae is selected to be >10 to eliminate false minutiae extraction because minutiae

are usually not that close to each other. The target triples which match the above criteria are considered to be matched.

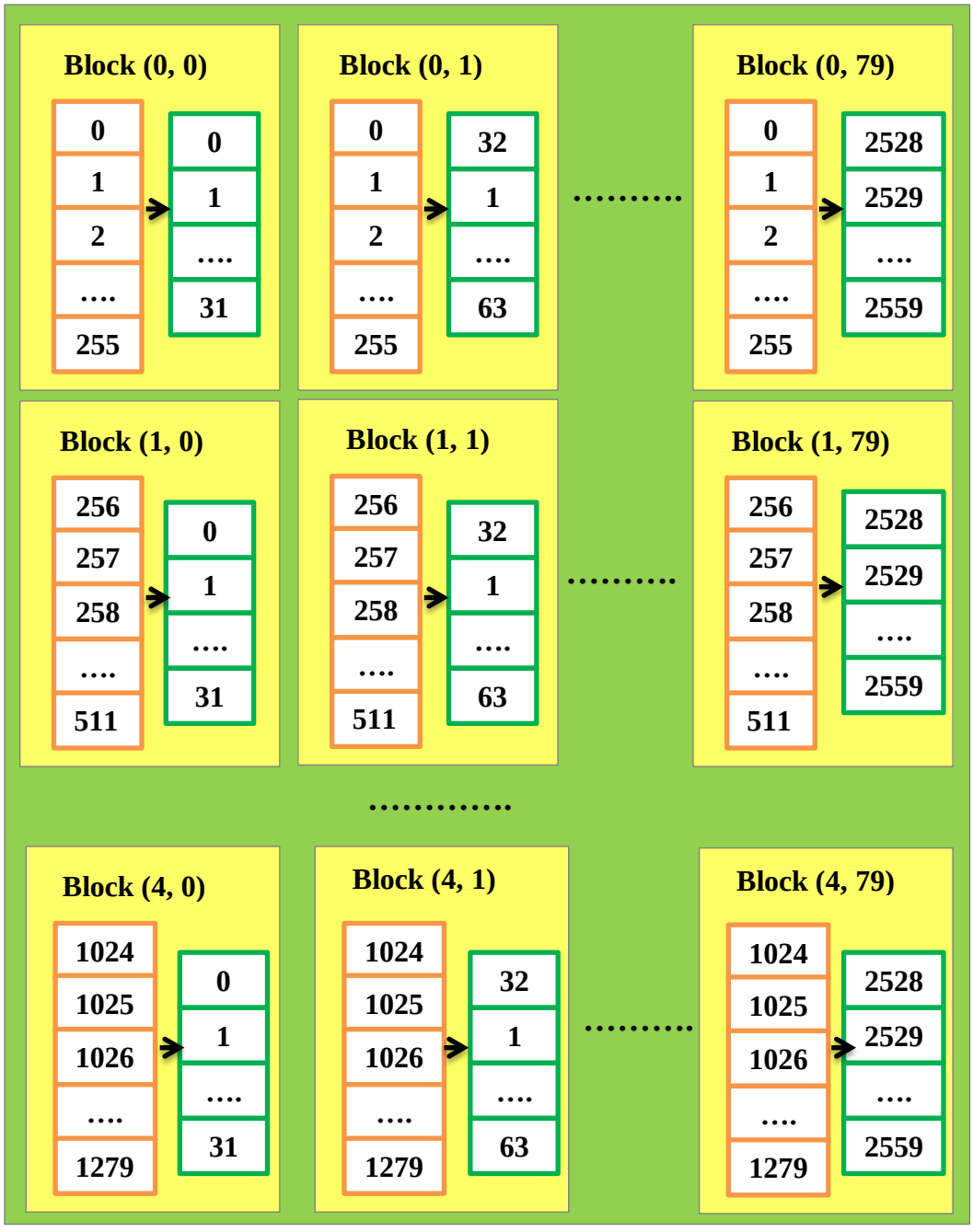

Figure 3.4 Grid structure for 3-point comparisons

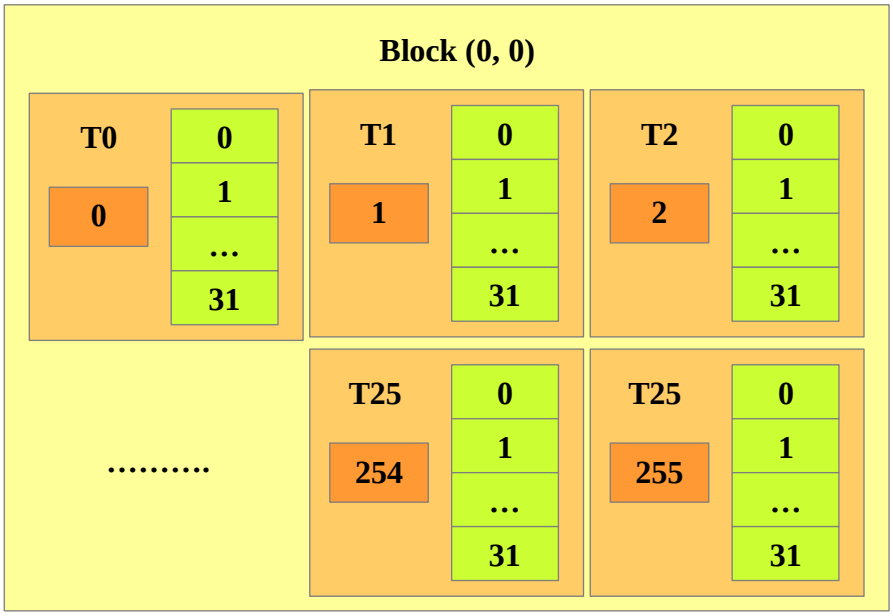

Figure 3.5 Block structure for 3-point comparisons

Consider a reference triple compared with 32 target triples in a thread. The results for these comparisons are stored in an array whose size is equal to the total number of threads in the grid given by the following equation.

$$Array\ size\ for\ matched\ triples = Number\ of\ blocks\ in\ a\ row * Number\ of\ blocks\ in\ a\ column * Number\ of\ threads\ per\ block$$

For the grid shown in Figure 3.4, the array size is 5 * 80 * 256 = 102400. The data transfer between CPU and GPU takes considerable time and the amount of data transferred should be reduced. Hence, in a block, though 32 target triples are compared, the result is stored in a single integer.

After all the threads finish comparisons, the results are transferred back to CPU. The number of triple matches which is the sum of matches of all the reference triples is calculated from these results. The results are stored in two arrays – one for the reference triples and the other for target triples. The elements in the array are stored in such a way that triples at the same index in the two arrays match. Matched reference and target arrays are shown in Figure 3.6. N+1 is the number of 3-point matches. R0, R1, R2, R3 … RN are reference triples and T0, T1, T3, T3 …

Reference [3-point matches] = {R0, R1, R2, R3, R4, R5, R6, R7…….RN}

Target [3-point matches] = {T0, T1, T2, T3, T4, T5, T6, T7…….TN}

Figure 3.6 Matched reference and target arrays

TN are target triples. Also, R0 matches T0, R1 matches T1, R2 matches T2, R3 matches T3, and R4 matches T4 and so on.

3.3.3 Two Triangle (6-point) Comparisons on the GPU

The matched reference and target arrays from 3-point comparisons are used for 6-point comparisons. Two 3-point features can be combined to form a 6-point feature. For example, in Figure 3.6, (R0, R1) together form a 6-point feature. Similarly, (R0, R2), (R0, R3) … (R0, RN), (R1, R2), (R1, R3) … (R1, RN) … RN-1, RN) form 6-point features. Though a 6-point feature is formed by combining two triples, the actual number of minutiae points in a 6-point feature can be less than 6 because two triples can have more than one minutiae point in common. In Figure

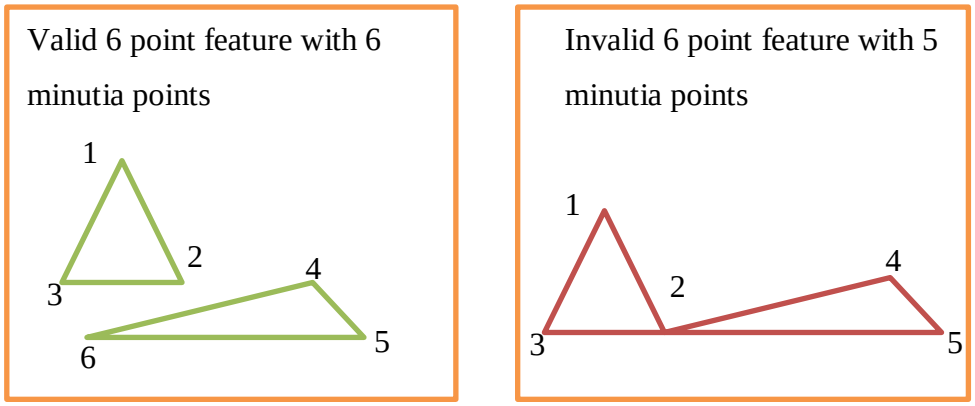

Figure 3.7 Valid and invalid 6-point features

3.7 valid and invalid 6-point features are shown. Similarly, 6-point features can be formed by combining elements of matched target array. Whenever a 6-point feature is formed from elements of reference matched array, a 6-point feature is also formed from the elements of the same indices of target matched array. For example, 6-point features (R0, R1) and (T0, T1) are formed together and compared. Some of the other comparisons are ((R0, R2) -> (T0, T2)), ((R0, R3) -> (T0, T3)), ((R1, RN) -> (T1, TN)) and so on.

The number of blocks is based on the number of 3-point matches. The main idea is to combine one index form the reference matched array with the remaining indices to form a 6-point feature and compare in one block. If *n* is the number of 3-point matches which is also the size of the reference matched array, then the number of blocks is also equal to *n*. These bocks are distributed into the 2-dimensional grid with number of rows equal to $\sqrt{n}$ and number of columns is equal to $\sqrt{n}$ or $\sqrt{n} + 1$. The grid structure is illustrated in Figure 3.8. In Block 0, all the combinations that can be formed with R0 and the remaining 3-point features R1, R2, R3 … RN are formed and compared with target 6-point features which are also formed in the exact same way as that of reference 6-point features.

The number of threads in a block is fixed at 256. The number of 6-point features compared in a thread is given by the following equation.

$$Number\ of\ 6point\ comparisons\ per\ thread\ = \frac{Number\ of\ 3point\ matches}{256} = \frac{n}{256}$$

Matching criteria: In a 6-point feature, the centroids of both the triples are calculated and the distance between the two centroids is calculated. The centroid distances of the reference and target fingerprints are compared to find a match. The reference 3-points involved in the 6-point feature already match with target 3-points. So, in a 6 point feature, on adding the matching criteria for 3-point comparisons, totally 6 sides, sum of angles of both the triples, types of minutiae, centroid distances are used for comparisons. Similar to 3-point comparisons, the results are saved in two reference arrays – one for each of the triples involved in forming the 6-point feature. Similar arrays are also formed for the target triples as shown Figure 3.9. The elements R10, R20 from the Reference arrays form a 6-point feature and they match with the 6-point feature formed by elements T10, T20 from the target arrays.

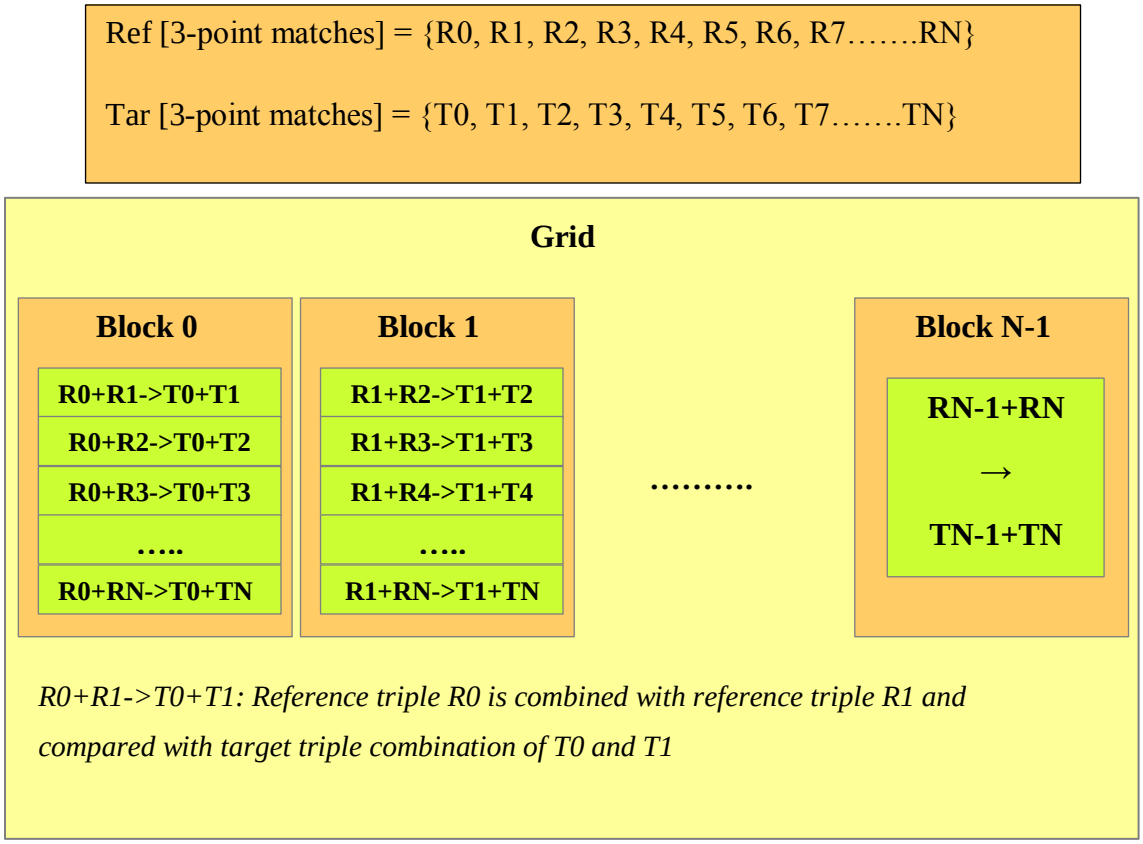

Figure 3.8 Grid Structure for 6-point comparisons

Reference 1 [3-point matches] = {R10, R11, R12, R13, R14, R15, R16, R17…….R1N}

Reference2 [3-point matches] = {R20, R21, R22, R23, R24, R25, R26, R27…….R2N}

Target1 [3-point matches] = {T10, T11, T12, T13, T14, T15, T16, T17…….T1N}

Target2 [3-point matches] = {T20, T21, T22, T23, T24, T25, T26, T27…….T2N}

Figure 3.9 6-point matches

3.3.4 Three Triangle (9-point) Comparisons on the GPU

The matched reference1, reference2, target1 and target2 arrays from 6-point comparisons and matched reference and target arrays from 3-point comparisons are used for 9-point comparisons. A 6-point feature can be combined with a 3-point feature to form a 9-point feature. In Figure 3.11 three arrays of reference triples and also three arrays of target triples are shown. (R10, R20, R30) together forms a 9-point feature. Similarly, (R10, R20, R31), (R10, R20, R32) … (R10, R20, R3N), (R11, R21, R31), (R11, R21, R32) … (R11, R21, R3N) … (R1N, R2N, R3N) form 9-point features. It should be noted that only elements of the same index from Reference1 and Reference2 arrays form a 6-point feature and it can be combined with any element of Reference3 array. Some invalid 9-point features may also be formed. Figure 3.10 shows valid and invalid 9-

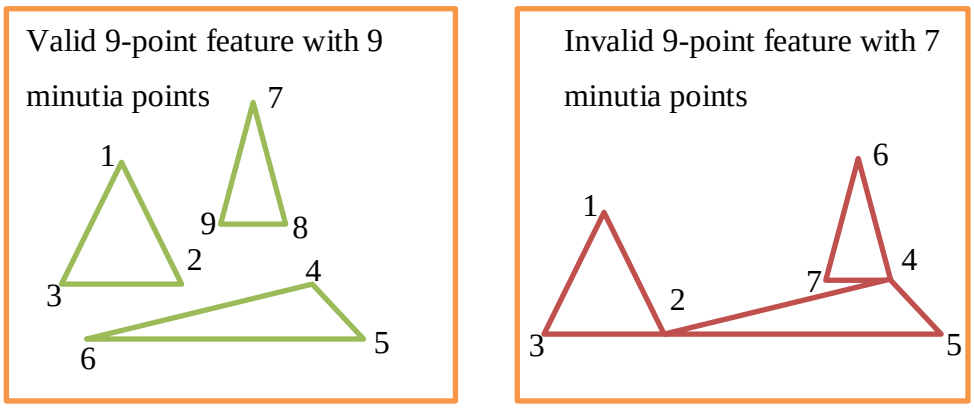

Figure 3.10 Valid and invalid 9-point features

point features. The invalid features are filtered out before comparisons. 9-point features are also formed by combining elements of matched target arrays. Whenever a 9-point feature is formed from elements of reference matched arrays, a 9-point feature is also formed from the elements of the same indices of target matched arrays. For example, 9-point features (R10, R20, R30) and (T10, T20, T30) are formed together and compared. Some of the other comparisons are ((R10, R20, R31) → (T10, T20, T31)), ((R10, R20, R32) → (T10, T20, T32)), ((R10, R20, RN0) → (T10, T20, TN0)) and so on.

Figure 3.11 Grid structure for 9-point comparisons

Similar to the grid structure of 6-point comparisons, the number of blocks in the grid of 9-point comparisons is based on the number of 6-point matches. If *n* is the number of 6-point matches, then the number of blocks is also equal to *n*. These bocks are distributed into the 2-dimensional grid with number of rows equal to $\sqrt{n}$ and number of columns is equal to $\sqrt{n}$ or $\sqrt{n}+1$. The grid structure is illustrated in Figure 3.11. In Block 0, all the combinations that can be formed with R10, R20 and the 3-point features from reference3 array R30, R31, R32… R3N are formed and compared with target 9-point features which are also formed in the exact same way as that of reference 9-point features.The number of threads in a block is fixed at 256. The number of 9-point features compared in a thread is given by the following equation.

$$Number\ of\ 9point\ comparisons\ per\ thread\ = \frac{Number\ of\ 6point\ matches}{256} = \frac{n}{256}$$

Matching criteria: In a 9-point feature, the centroids of the three triples are computed and the distance between the first triple's centroid and third triple's centroid, second triple's centroid and third triple's centroid are calculated. It should be noted that distance between first triple's centroid and second triple's centroid is already compared in 6-point comparisons. The centroid distances of the reference and target 9-point features are compared to find a match. The reference 3-points involved in the 9-point feature already match with target 3-points. So, in a 9 point feature, on adding the matching criteria for 3-point comparisons, totally 9 sides, sum of angles of the three triples, types of minutiae, centroid distances are used for comparison.

3.3.5 Results

The correctness of the GPU parallel algorithm is verified by comparing the 3-point and 6-point matches of CPU and GPU. The results are identical for both of them. Table 3.1 shows the time taken for 3-point and 6-point comparisons on CPU and GPU. The fifth and last columns show the speedup obtained for the GPU parallel algorithm.

The table shows results for 24 randomly selected fingerprints compared with a single reference fingerprint with 5456 triples to demonstrate the speedup obtained by using the parallel algorithm. Though only 24 fingerprints are selected, any number of fingerprints can be used for matching. For any target fingerprint compared with any reference fingerprint, similar speedups can be observed. The GPU and CPU times are shown in milliseconds (ms).

The fingerprints with fewer minutiae and therefore fewer triple combinations take less time even on a CPU which can be observed for fingerprints 3, 12, 21 which take 10 ms, 40 ms and 40 ms on a CPU respectively. On a CPU, the comparison is performed with multiple 'if' conditions. If an 'if' condition does not satisfy, then the rest of the comparison is skipped. But on a GPU, none of the comparisons are skipped because comparisons run independently on separate threads. So, the results from GPU are obtained only after all the threads finish comparisons and hence the time which was reduced on a CPU by skipping the comparisons cannot be reduced.

Table 3.1 Results of 3-point and 6-point comparisons

File	Number of triples	3 point CPU (ms)	3 point GPU (ms)	3 point speedup (ms)	6 point CPU (ms)	6 point GPU (ms)	6 point speedup (ms)
1	8436	190	60	3.17	4770	760	6.28
2	3654	80	20	4.00	350	50	7.00
3	680	10	10	1.00	70	10	7.00
4	10660	260	30	8.67	4580	710	6.45
5	5456	140	20	7.00	2550	360	7.08
6	20825	550	60	9.17	33310	5570	5.98
7	13244	280	30	9.33	22210	3830	5.80
8	1771	50	10	5.00	270	40	6.75
9	1771	50	10	5.00	140	20	7.00
10	4960	120	10	12.00	350	50	7.00
11	2300	60	10	6.00	580	80	7.25
12	1540	40	10	4.00	150	20	7.50
13	3276	90	10	9.00	650	90	7.22
14	5984	160	20	8.00	2870	450	6.38
15	11480	320	30	10.67	13740	2390	5.75
16	12341	310	30	10.33	9580	1550	6.18
17	19600	440	60	7.33	7880	1250	6.30
18	11480	310	30	10.33	3230	500	6.46
19	5984	140	10	14.00	1750	230	7.61
20	9139	240	20	12.00	3540	530	6.68
21	1540	40	10	4.00	70	10	7.00
22	18424	410	40	10.25	36330	6740	5.39
23	23426	570	60	9.50	40630	7750	5.24
24	16215	360	50	7.20	9190	1480	6.21
Average Speedup				7.79			6.56

The average speedup obtained for 3-point comparisons is 7.79 and for some of the fingerprints, the speedup is as high as 14 (Fingerprint 19 in Table 3.1). The average speedup obtained for 6-point comparisons is 6.56. For 6-point comparisons, the invalid 6-points are resolved only on GPU and since in each block, the valid 6-point combinations differ, there are multiple unused threads running on the GPU. If the invalid 6-points are resolved on CPU before passing them to the GPU, it would take much longer on CPU because there are millions of 6-point combinations that can be formed. Hence, though it takes longer, the 6-point features are resolved only on GPU.

It should be noted that shared memory was not used for this method. By moving the target triples into shared memory, the time for memory transfer can be reduced to a great extent. Also, the total time for GPU comparisons is the sum of times taken for data transfer between CPU and GPU and the actual computations on GPU. It is observed that the data transfer takes considerable amount of time because unnecessary data was being copied to GPU. The following section describes usage of shared memory and the optimizations that were used to reduce the data transfer time.

3.4 GPU Comparisons of FBI Database Fingerprints

The FBI database consists of 74,140 rolled fingerprint images from which there were 2,575 different subjects providing multiple impressions of 23,942 unique fingerprints. This database was provided by FBI CJIS. These fingerprints are grouped into very good, good, bad and ugly fingerprints based on the global image quality score assigned by Verifinger SDK [38]. In this work, fingerprints from very good and good groups are considered because of their quality. A single unique fingerprint is considered from multiple impressions of same finger and a database of 18,649 fingerprints is formed out of these good quality fingerprints.

3.4.1 Input Files

The FBI fingerprints are processed into text files called fts files. The feature extraction tool used was Neurotechnology's Verifinger SDK [38]. These files are similar to the minutia xyt files discussed in the previous section and contain x and y coordinates, angle of orientation (theta),

quality of the minutia, type of minutia, ridge curvature, ridge density and the ridges involved in forming the minutia point. The quality range of the minutia points is 0-100. Some additional information related to core points and delta points is also extracted into these files. A sample fts file is shown in Figure 3.12. In this figure, *'Q'* represents the quality of the fingerprint, 'C' represents the core points, *'D'* represents the delta points and *'M'* represents the minutia points in the fingerprint. For the core points, the location and angle of orientation are shown. For delta points, the locations are shown. In the Figure 3.12, an fts file of a fingerprint with two core points and two delta points is shown. The core points are located at (390, 334) and (456, 361) coordinates and with angles of orientation 19^{0} and 148^{0} respectively. The delta points are located at (546, 404) and (210, 455). The first minutia point is located at coordinates (285, 108) with angle of orientation of 120^{0}, a quality of 30, minutia type of 1(ridge ending), ridge curvature of 124, ridge density of 6 and ridge ID of the ridge forming this minutia is 195. Ridge endings are represented by 1 and ridge bifurcations are represented by 2. The average of the qualities of all the minutiae points in a fingerprint fts file is calculated and all the minutiae points greater than or equal to average quality are considered. Hence, the quality factor considered for each fingerprint fts file is different.

```
Q   255
G   138
C   390   334    19
C   456   361   148
D   546   404    -1    -1    -1
D   210   455    -1    -1    -1
M   285   108   120    30     1   124     6   195
M   569   151   153    20     1    86     9   198
M   471   159   146    43     2   121     5   193   192   189
M   314   176   116    86     1   128     6   186
M   495   177    25    84     1   122     5   184
M   269   180   237    22     2   118     8   190   183   188
M   361   202   254    90     1   137     8   180
M   394   202   130    87     1   144     8   180
```

Figure 3.12 Sample fts file

Similar to the FVC 2000 database, ridge count files are used for this database too. The ridge count files are in the form of a two dimensional array with each element in a line giving the number of ridges crossed between two minutia points. For example, let *n* be the total number of minutiae in the fingerprint. The first line in the file gives the number of ridges crossed between

minutia 0 and 0-*n* minutiae in the fingerprint, second line gives the number of ridges crossed between minutia 1 and 0-*n* minutiae in the fingerprint and so on. In Figure 3.13, a sample ridge counts file with 11minutiae is shown. The first line shows the number of ridge counts between first minutia and all the 11 minutiae.

0	7	7	8	6	6	11	12	9	12	9
7	0	5	10	6	13	13	13	14	15	10
7	5	0	5	1	6	8	8	8	10	4
8	10	5	0	7	1	5	5	2	5	10
6	6	1	7	0	7	7	7	12	8	4
6	13	6	1	7	0	6	6	3	7	11
11	13	8	5	7	6	0	0	4	2	5
12	13	8	5	7	6	0	0	5	2	4
9	14	8	2	12	3	4	5	0	5	11
12	15	10	5	8	7	2	2	5	0	6
9	10	4	10	4	11	5	4	11	6	0

Figure 3.13 Sample ridge count file

3.4.2 3-point GPU Comparisons

In section 3.3, it is discussed that the limit for the largest side is ≤ 150 and the limit for the smallest side is > 10 pixels. Instead of applying this condition while matching the fingerprint features, the reference triples are filtered out before the actual comparisons. So, when the reference triples are formed, all those triples whose largest side > 150 and smallest side < 10 pixels are not considered. This would reduce the number of triples and thus many unnecessary comparisons can be avoided and therefore reduces the time taken. For example, the number of triples for a reference fingerprint from FBI database before this condition was applied is 1140 and it reduced to 865 on applying this condition and for another reference fingerprint from FBI database with originally 39,711 triples, the number of triples reduced to 14,818 on applying this condition. This is one of the optimizations used when compared to the sequential comparisons and also the GPU comparisons of FVC 2000 fingerprints discussed in section 3.3

For GPU matching, the reference and target fingerprint data have to be transferred from CPU to GPU. The time taken for this transfer is high because the data structure used for triples is very large. The data structure size used for triples is 84 bytes. The original data structure for a triple is

```
struct Triple{
int i;
int j;
int k;
int sum_dist;
int smallest_dist;
int largest_dist;
float sum_dist_thresh;
float smallest_dist_thresh;
float largest_dist_thresh;
int d1, d2, d3;
int th1, th2, th3;
int rc1, rc2, rc3;
int sum_rc;
int small_rc;
int large_rc;
};
```

Figure 3.14 Original triple data structure

shown in Figure 3.14.

On the GPU kernel, not all of the data members in the Triple data structure are used. Since only largestDistance, smallestDistance, sumofDistances, largestDistanceThreshold, smallestDistanceThreshold, sumDistanceThreshold and the ridges crossed rc1, rc2, rc3 are compared, a smaller data structure can be used with data members as shown in Figure 3.15. The thresholds are calculated on the GPU kernel instead of copying it to the GPU. Further, the ridge counts are always less than 100 and it is a 2-digit number. The ridge counts rc1, rc2 and rc3 are sorted and can be combined to form a single integer value shown below. This reduction of the

data structure size does not affect the number of matches because all the required data are copied to the GPU.

$$rc_{combined} = largest_{rc} * 10000 + intermediate_{rc} * 100 + smallest_{rc}$$

The new data structure of size 16 bytes is shown in Figure 3.16.

```
Struct smallTriple{
int rc1, rc2, rc3;
int largestDistance;
int smallestDistance;
int sumofDistances;
};
```

Figure 3.15 Modified triple data structure

```
Struct smallTriple{
int rcCombined;
int largestDistance;
int smallestDistance;
int sumofDistances;
};
```

Figure 3.16 New triple data structure

As discussed in Chapter 2, a GPU consists of 30 streaming multiprocessors and every block of GPU grid executes on a single streaming multiprocessor. At any time, 32 threads from a block, known as a warp, run on a streaming multiprocessor. Hence, the number of threads in a block is chosen to be a multiple of 32. The implementation is similar to the discussion in section 3.3 with additional usage of shared memory. The number of threads per block is set as 512 which is also the maximum number of threads per block (ThreadsPerBlock = 512). Also, all the blocks in a row use the same set of reference triples.

In all the blocks of row 0, 0-512 reference triples are used for comparisons. In each thread of a block, a reference triple is compared with a set of target triples. Let *x* be the total number of

triples in reference fingerprint and *y* be the total number of triples in target fingerprint. The number of rows in the grid is calculated based on the number of the reference triples given by the following equation.

$$Number\ of\ rows\ in\ grid\ = \frac{(x\ )}{ThreadsPerBlock} + (x\ \%\ ThreadsPerBlock\ \neq\ 0)$$

Usage of shared memory: A shared memory of 16KB is available for each streaming multiprocessor. In other words, 16KB shared memory is available for each block. In all the threads of a block, the same set of target triples are compared with reference triple. For example, let *b* be the number of target triples compared in a block. In thread 0 of block 0, the reference triple 0 is compared with 0-*b* target triples. In thread 1, the reference triple 1 is compared with 0-*b* target triples. Similarly, in each thread of the block, 0-*b* target triples are compared with a reference triple. Since the same set of target triples are compared in all the threads of a block, by moving these triples into shared memory, the time to fetch this memory is reduced to a great extent. Otherwise in each thread, the triples will have to be fetched from global memory. Because only 16KB is available in shared memory, the number of target triples that can be copied into shared memory is limited and it can be calculated based on the size of 'smallTriple' which is given by the following equation.

$$a\ = \frac{16000\ bytes}{size\ of\ smallTriple} = \frac{16000}{16} = 1000$$

Although the number of target triples that can be copied into shared memory from the above equation is 1000, all the 1000 triples cannot be copied because of a small buffer memory used for shared memory. So, 992 target triples are copied into shared memory. Whenever memory is fetched from shared memory, it is done in 32 bytes at a time. Hence, the number 992 is selected which is also a multiple of 32. So, the number of comparisons per thread in a block is 992 (ComparisonsPerThread = 992).

The number of columns in the grid is calculated based on number of comparisons per thread in a block and it is given by the following equation.

$$Number\ of\ columns\ in\ grid = \frac{(y)}{ComparisonsPerThread} + (y\ \%\ ComparisonsPerThread \neq 0).$$

GPU Kernel Function: Firstly, the 992 triples are copied into shared memory. Since there are 512 threads in each block, in each thread, 2 reference triples can be copied into shared memory. Figure 3.17 shows the copying of reference triples into shared memory.

```
idx = threadIdx.x * 2;
tidx = blockIdx.y * 992 + idx;
for(i = 0; i < 2; i++)
        targetTriplesSharedMemory[idx+i] = targetTriples[i+tidx];
```

Figure 3.17 Shared memory

In thread 0, reference triples 0 and 1 are copied into shared memory. In thread 1, reference triples 2 and 3 are copied into shared memory and so on. The comparisons cannot be immediately started after copying reference triples into shared memory because it has to be ensured that all the threads finish copying triples into shared memory. The function __syncthreads () ensures that all the threads finish copying triples into shared memory. In each thread, a reference triple is compared with 992 target triples. These 992 target triples are now fetched from shared memory; hence the data transfer takes lesser time when compared to fetching the data from global memory.

In first row of the block, 0-511 reference triples are used for comparison. In second row of the block, 512-1023 reference triples are used for comparison and so on. Similar to CPU implementation, the smallest distance, largest distance, sum of distance are copied into local variables and their thresholds are also calculated.

Matching Criteria: Unlike section 3.3 where type of minutia, angle of orientation (which gives the direction of minutia) was used, here, only side lengths of the triples and the ridges crossed are used. The minutiae type is not taken into consideration because in practice, ridge endings may appear as bifurcations and vice versa depending on the finger pressure against the surface

where the fingerprint impression is formed [13]. Also, a large variation was observed in the minutia direction when corresponding minutiae are compared in two different impressions of the same finger. Hence, largest side, smallest side, sum of sides and sorted ridges crossed are used.

In the reference fingerprint, only a set of 50 high quality minutiae are selected. This is because of the number of possible 6-point features that can be formed with 50 minutiae is $^{50}C_6 = 158$ million. Some of the fingerprints in the FBI database contain more than 100 minutiae and the number of 6-point features formed with 100 minutiae is very huge. To avoid this data complexity, only a set of 50 minutiae are considered for reference fingerprints. By this approach, some of the rare features which may be identified by using minutiae outside the set of 50 minutiae may be missed out. But by considering only 50 minutiae, rare features were still identified in this work (Chapter 4). Hence, the main aim of this work is not lost. It should be noted that all the good quality minutiae are considered for the rest of the target fingerprints in the database, the limit of 50 minutiae is set only for reference fingerprint.

The total number of threads running on GPU is blocksPerGrid_x * blocksPerGrid_y * threadsPerBlock where threadsPerBlock = 512, where the maximum of blocksPerGrid_y could be 512. Some of the fingerprints have as many as 500,000 3-point features and given that number of target triples compared in a block in any given row is 992, the maximum value of blocksPerGrid_y = 500,000/992 ≈505. The value of blocksPerGrid_x is less than 40 because only 50 minutiae are considered for reference fingerprint and the total triple combinations with 50 minutiae are 19600. So, given that threads per block is 512, the maximum value of blocksPerGrid_x = 19600/512 ≈ 39. So, the maximum number of total number of threads is given by 39 * 512 * 512 = 10,223,616 ≈ 10 million. In each thread, comparisons of 992 target 3-point features are made. So, total number of comparisons on GPU is 10,223,616*992 ≈10 billion. It is also observed that in each thread, a reference triple has just 4 matches when compared to 992 target triples. So, an array of 5 short integers is allocated to each of the threads running on GPU. Whenever a match is found on GPU, the index of the matched target 3-point feature ranging between 0-991 is copied into the 5 integer array starting from index 1. Index 0 of the 5 integer array is used to keep track of the number of matches for that reference triple. The target 3-point feature index ranges between 0-991 because in each block, the 992 target 3-point features

are copied into shared memory. This index should be decoded into the original target 3-point index on CPU.

The matched reference and target triples are stored in two arrays- reference matched array and target matched array similar to the implementation in section 3.3.

3.4.3 6-point GPU Comparisons

```
struct minTriple{
short centroid x-coordinate;
short centroid y-coordinate;
unsigned char i; //minutia 1
unsigned char j; //minutia 2
unsigned char k; //minutia 3
int tri_id; // index of the original reference triple
};
```

Figure 3.18 Data structure for 6-point comparisons

From the results of 3-point comparisons, for each reference triple, the matching indices of the target triples are known. For comparisons on GPU, the minutiae of all the triples, centroid distances, ridges crossed should be passed to the GPU. Ridges crossed are passed as a separate data structure. The minutiae and centroid coordinates are passed to GPU as a data structure as shown in Figure 3.18.

The data for all the matched reference 3-points and target 3-points is copied onto GPU. The ridges crossed between all the minutiae points of reference and target triples are also copied onto GPU. Similar to 3-point comparisons, a two-dimensional grid is launched. The number of threads per block and the number of blocks in y-direction (columns) is fixed at 512. The number of rows of the grid is calculated based on total number of 3-point matches and number of columns and it is given by the following equation.

$$blocksPerGrid\_x = \frac{total\_3point\_matches}{512} + (total\_3point\_matches\%512 \neq 0)$$

The number of comparisons per thread is given by the following equation.

$$CompPerThread = \frac{total\_3point\_matches}{512} + (total\_3point\_matches\%512 \neq 0);$$

Shared Memory: Shared memory is used to store ridges crossed of the reference fingerprint. The number of ridges crossed for all minutia combinations is stored in a 2-dimensional array as already discussed and the number of minutiae in fingerprint could be a maximum of 255. But for the reference fingerprint, only a few 50 high quality minutiae are considered and the ridges crossed array is an array of 50x50 with 2500 elements. So, it is possible to store the ridges crossed of only the reference fingerprint in shared memory. Shared memory of data type 'unsigned char' of 2500 array size is used. In each thread of a block (512 threads per block), 5 ridges crossed can be copied into shared memory. After the copy, __syncthreads() function is called to make sure that all the threads finished copying reference ridges crossed into shared memory.

Matching Criteria: The centroids are calculated for both the triples involved in the 6-point feature and the distance between these two centroids is calculated. There are 9 ridges crossed between the minutia points of first triple and the second triple. Adding matching criteria from 3-point comparisons, totally 6 side lengths, 15 ridge counts, centroid distances are used for 6-point comparisons.

GPU Kernel Function: In each block, the elements of an index of the matched reference and target arrays are combined with the remaining elements of the matched array. This combination is a 6-point feature and it should be noted that all these 6 points do not form a valid combination i.e. all these 6 points may not be different and the validity of 6-point is checked before the actual comparison is made. The 6-point feature is formed and comparison is made at the same time. For example, in block 0, index 0 of reference matched array is combined with indices 1, 2, 3...n as discussed above, where n is the total number of 3-point matches and the same combination is formed from elements of target matched array and the reference and target 6-points are compared. In each thread, specific number of combinations given by compPerThread is made. The validity of the 6 point feature is checked by comparing all the minutia points of the two triples being combined. For the valid 6-point features, the 9 ridges crossed arrays of reference and target features are sorted and compared in a one-one manner. For the matched 6-point

features, centroid distances are calculated for reference and target features and compared. The matches for 6-point features are stored in a 1-dimensional array such that each index of this array is given by ref1* Number of reference triples + ref2 where ref1 is the index of the 1st reference triple and ref2 is the index of the 2nd reference triple in the 6-point feature.

3.4.4 Results

Table 3.2 shows results for 3-point comparisons using the FBI fingerprint database. The results are obtained by comparing a reference fingerprint of 50 minutiae and 19408 triple combinations with 10 randomly selected fingerprints.

Table 3.2 Results for 3-point comparisons of a reference fingerprint compared with 10 target fingerprints from FBI database on GPU

Fingerprint	# Triples	CPU matches	GPU matches	CPU time (ms)	GPU time (ms)	Speedup
1	67379	5172	5172	18366	233	78.8
2	39589	3325	3325	6680	211	31.65
3	113216	7596	7596	30901	273	113.1
4	73075	5557	5557	16702	237	70.4
5	15136	1486	1486	2334	191	12.2
6	62125	3876	3876	16805	226	74.3
7	37760	3576	3576	6340	205	30.9
8	91719	5772	5772	25479	247	103.1
9	61701	4691	4691	15554	222	70
10	67162	5911	5911	17929	227	78.9

The number of triples is all the possible triples formed by all the good quality minutiae in fingerprints. The CPU and GPU matches are the total number of matches of all the triples of reference fingerprint when compared with a target fingerprint. The average speedup obtained for

comparisons of these 10 fingerprints is 66.38. It can be observed from Table 3.2 that for fingerprint 5, the speedup obtained is the lowest. This is because the number of triples for this fingerprint is just 15K which is small compared to other fingerprints. GPUs give the best results when the data is large and the application is computation-intensive. In this case, the data is small and hence the time taken for comparisons on GPU and CPU is almost the same. The CPU and GPU results for the fingerprints are also provided in the table to verify the correctness of GPU comparisons.

Table 3.3 shows the results for 6-point GPU comparisons. From the reference fingerprint, 50 minutiae are chosen which give 19408 triple combinations and 155 million valid 6-point features.

Table 3.3 Results for 6-point comparisons of a reference fingerprint compared with 10 target fingerprints from FBI database on GPU

Fingerprint	**CPU matches**	**GPU matches**	**CPU time (ms)**	**GPU time (ms)**	**Speedup**
1	3	3	14910	332	44.90
2	0	0	8990	263	34.18
3	7	7	34920	451	77.42
4	2	2	21078	335	62.91
5	1	1	2533	197	12.85
6	2	2	16587	284	58.40
7	2	2	8670	255	34
8	4	4	28339	358	79.15
9	0	0	17429	312	55.86
10	4	4	20743	349	59.43

The CPU and GPU matches shown in the table are the total number of matches for all the 6-point features of the reference fingerprint when compared with a target fingerprint. It should be noted that the time taken for the comparisons shown in Table 3.3 is the sum of the times taken for 3-

point and 6point comparisons, i.e. the execution times shown here include time taken for 3-point comparisons. The average speedup obtained for these 10 fingerprints is 51.91. The CPU and GPU matches are provided to verify the correctness of GPU matches. The number of matches is quite low for 6-point comparisons because of the stringent conditions that were used. It was assumed that the ridges crossed between any two minutiae points remains constant irrespective of the distortion, hence the matching criteria was to find exact ridges crossed between two minutia points without any threshold. But, it was observed later that the ridges crossed do not remain constant for two different fingerprints of the same finger (Chapter 4).

Table 3.4 shows results for the overall 3-point and 6-point comparisons when a reference fingerprint is compared with all the fingerprints in the FBI database. The number of good quality unique fingerprints considered from the FBI database is 18,649 (section 3.4). The table shows the results for 4 fingerprints from the database. Although only 4 fingerprints are selected to demonstrate the speedup, the GPU time taken for any reference fingerprint follows the same pattern. These 4 fingerprints are used as a reference one-by-one and compared with the entire database of 18,649 fingerprints.

Table 3.4 Results after comparing with 18,649 fingerprints

Fingerprint	**3 points**			**6 points**			**GPU time (minutes)**
	#	Matched	Unmatched	# (million)	Matched	% Unmatched	
1	19408	19408	0	155	44011	99.97	86.3
2	19552	19552	0	158	118332	99.92	96.8
3	19409	19409	0	155	30001	99.98	86.1
4	19504	19504	0	157	76498	99.95	89.0
Average GPU time							89.55

The results in Table 3.4 show that through this GPU implementation technique, thousands of fingerprints can be compared with a single fingerprint in less than 2 hours and also ensuring accuracy and precision in matching. The average time taken for these 4 fingerprints is 89.55 minutes. The time taken for the same comparisons to be done on CPU per fingerprint is around 60 hours which is nearly 2.5 days. The overall speedup obtained by this implementation is 40.2

$$Overall\ Speedup = \frac{60 * 60}{89.55} = 40.2$$

The table also shows the number of unmatched 3-points and 6-points after comparing with the entire database. It can be observed from the table that the number of unmatched 3-points is 0 for all the 4 fingerprints after comparing with the database. This shows that rare 3-point features do not exist. But the percentage of unmatched 6-points left after comparing with the database is >99%. This is because of the stringent thresholds that were used for side lengths and ridge counts for 6-point comparisons. The distortion in fingerprints and thresholds required to overcome this distortion is discussed in chapter 4. The results at the end of the chapter 4 show that fingerprints do not have rare 6-point features but have a few rare 9-point features.

The significance of this GPU algorithm is to show that the triplet-based matching techniques can be accelerated through GPUs. The methods used in this algorithm can be used for future implementations and equivalent speedups can be achieved.

Chapter 4: Mining Rare Features

4.1 Overview

The technique that was used for fingerprint triple-based feature matching is used to search for rare features in fingerprints. A feature is considered rare if it is statistically uncommon; that is, the rare feature should be unique among *N* randomly sampled prints. This rare feature could be a 3-point feature, 6-point feature, 9-point feature, 12-point feature or even higher. A feature which is rare in a database of 1000 fingerprints may find a match in a database of 2000 fingerprints. So, the rare features identified by the method proposed in this work may find a match if a slightly larger database is considered. However, the importance of this method is establishing the fact that fingerprints contain a small percentage of rare features that are unique to a set of fingerprints, i.e. the rare feature occurs only once in some *N* fingerprints. When comparing two fingerprints, when the matching features contain some rarity that was previously identified within the print, the confidence in the match is increased. In other words, a match that contains rare features would yield higher confidence in the conclusion than a match that does not contain any rare features.

4.2 Challenges in Mining Rare Features

From the results obtained from comparisons on FBI database (Chapter 3), it was observed that for all the reference fingerprints, there are around 99% of the 6-point features that are rare. The results of the algorithm discussed in Chapter 3 are verified by making two simple comparisons of reference fingerprint. Two cases for a reference fingerprint are considered a) different impression of the same finger b) fingerprint from a different finger. For a), the reference fingerprint and its impression should have many matches because they are fingerprints of the same finger, although it would not be a complete match because some minutiae in the reference fingerprint may not be present in the second impression due to the portion of fingerprint acquired. But, it was observed

that fingerprints a) and b) both have very few matches with reference fingerprint. This is problematic, since we would expect that many features should match when two images of the same finger are being compared. This problem is because of the stringent matching criteria that were used. It was assumed that the number of ridges crossed between any two minutiae would be identical for two fingerprints of the same finger. So, one-one exact comparisons of ridge counts for corresponding sides of a triple are made. However, it is observed that the number of ridges crossed between corresponding minutiae in different impressions of the same finger varied. This is because of the non-linear distortion of fingerprint minutiae. Feature extraction errors also account for variations in ridge counts. It was also observed that a constant distance threshold could not be used for all spatial distances in the fingerprint. Different kinds of distortion will be discussed in the next section.

4.3 Distortion

This section describes the extent of distortion and types of distortion between fingerprints of the same finger. The main factors that cause distortion in different impressions of the same finger (intra-class variations) are displacement, rotation, partial overlap, non-linear distortion, variable pressure, changing skin condition, noise and feature extraction errors [13].

4.3.1 Minutiae Distance Distortion

This distortion is observed by considering the spatial distance between any two minutia points. Let $(x1, y1)$, $(x2, y2)$ be two minutiae points, then the distance between these points is calculated as $\sqrt{((x1-x2)^2 + (y1-y2)^2)}$. A set of 8 impressions of the same finger is considered to analyze distortion in minutiae distances. 50 corresponding minutiae are identified in these fingerprints manually. First, the minutiae that match around the core region are identified and depending on these, remaining minutiae are identified. The distance between minutiae pairs is calculated and the distortion of this distance is analyzed for all the fingerprints. A graph showing the minutiae distance distortion is shown in Figure 4.1. With 50 minutiae identified in the fingerprints, the largest minutiae distance among all the minutiae pairs is 316 which is the final value on X-axis. This value can differ for different fingerprints depending on the minutiae considered and the

distance between them. The graph shown here is an example of the distortion in minutiae distances.

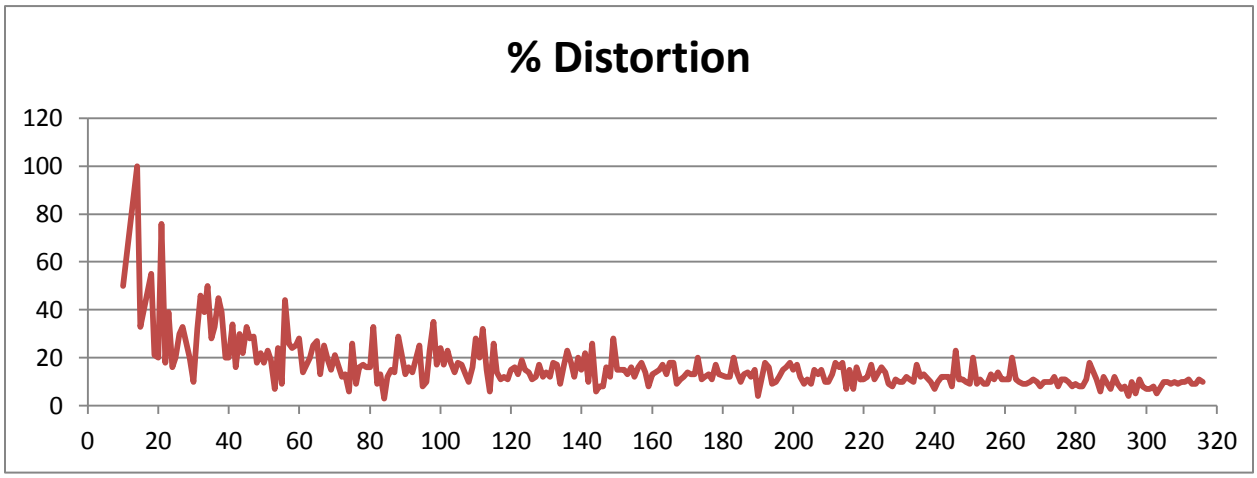

Figure 4.1 Minutiae distances distortion

X-axis shows the spatial distance between minutia pairs. Y-axis shows the maximum amount of distortion observed across 8 fingerprints as a percentage of the actual distance. It can be seen that for lower lengths, the amount of distortion is very high. For example, for a distance of 14, the distortion is 100%. For higher lengths, the distortion is around 20% of the distance. So, the threshold is dependent upon the side length of the triple. The threshold is selected such that as the distance increases, the threshold decreases. The chart showing distance threshold that is used in this matching algorithm is shown in Figure 4.2

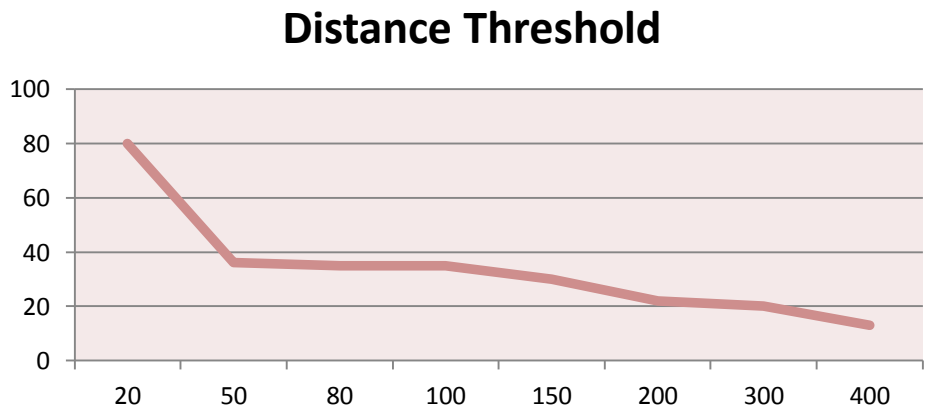

Figure 4.2 Distance threshold

The X-axis of the graph shows minutiae distances and the Y-axis shows the threshold used. The graph shown in figure 4.1 is plotted by considering different impressions of the same finger. But, the graph shown in Figure 4.2 is plotted by considering multiple impressions of different fingers. Around 40 different fingerprints are considered and multiple impressions of these 40 fingers are analyzed to calculate the maximum distortion for every minutia distance. The graph is particularly plotted for X-axis values 20, 50, 80, 100, 150, 200, 300, 400. It should be noted that maximum distortions are chosen for the minutiae distances.

4.3.2 Displacement of Minutiae

For two impressions of the same finger that are already aligned, the minutiae that are almost at the same level in the fingerprints, tend to move higher or lower which changes their positional coordinates because of the pressure variations. When minutiae extracted from two impressions (aligned) of the same finger are compared, they do not follow the same order. By using triplet based features which are translation invariant, the displacement of minutiae can be overcome. Also, the minutiae distance thresholds will subdue this displacement effect.

4.3.3 Ridges Crossed Distortion

The number of ridges crossed between any two corresponding minutiae points of two fingerprints of the same finger varies to a great extent. The factors that cause this distortion are given below.

- The number of ridges crossed between two minutiae points varies because of the movement of minutiae coordinates. For example, consider two minutiae points separated by some distance. Assume that there exists a ridge ending between the line joining these minutiae points. Consider the same two minutiae points and the ridge ending in a different impression of the same finger. Because of the movement of minutiae coordinates, the ridge ending might move lower or higher and it would not be present between the line joining the two minutiae points anymore. Hence, the number of ridges crossed for the two minutiae points for these two fingerprints differs by 1.

- For a triplet, it is observed that the number of ridges crossed between two minutiae is not proportional to the side length joining the two minutiae. In many cases, it is observed that the largest side does not always have higher ridge counts. Further analysis showed that the sides which have ridge flow almost parallel to them have fewer ridges crossed on them. In Figure 4.3, some examples of ridge counts which have fewer ridges crossed are shown. In such cases, the feature extracting tool does not always give the true ridges crossed and hence a different threshold must be used.

To solve this problem, the ridge counts are divided into two types, high confidence and low confidence ridge counts. In Figure 4.4, examples of high confidence ridge counts which have ridge flow direction almost perpendicular to the line segment joining the minutiae are shown. To differentiate between high confidence and low confidence ridge counts, an optimum ratio of side length to ridge count should be calculated. For most of the high confidence ridge counts in a fingerprint, this ratio is 10. There are many sides whose ratio falls below 10; however they still have high confidence ridge counts. A ridge count is considered to be low confident if it satisfies $(10 * (rc + 4) < d)$ where rc is the ridge count and d is the side length. This condition is deduced after observing multiple ridge counts and their distances. For high confidence ridge counts, a lower threshold is used and for low confidence ridge counts, higher threshold is used because low confidence ridge counts vary to a great extent across different impressions of the same finger. For threshold calculation, different impressions of the same finger are observed. For the analysis of distortion, a set of 35 different fingerprints are randomly selected. For these 35 different fingerprints, multiple impressions are considered and the corresponding minutiae are located. Finally, for high confidence ridge counts, a threshold of 5 is selected and for low confidence ridge count, a threshold of 12 is selected to account for distortion and feature extracting errors. By the condition used to check low confidence ridge count, it is possible for an originally high confidence ridge count to be considered low confidence ridge count. This is acceptable because by considering it as a low confidence ridge count, just the threshold is increased. This would not affect the comparison of different impressions of same finger. But, it would increase the number of triple matches for which some optimizations are proposed which will be discussed later.

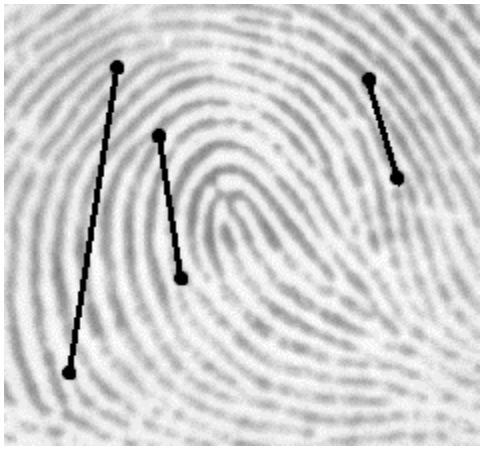

Figure 4.3 Low confidence ridge count

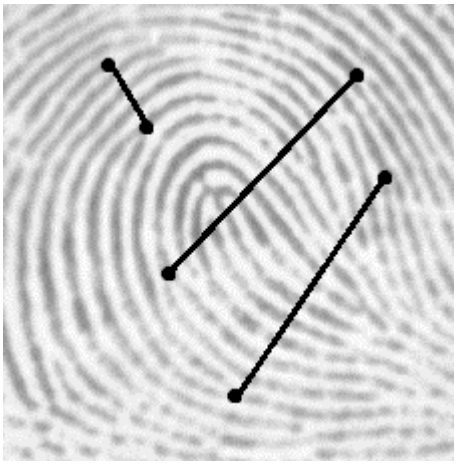

Figure 4.4 High confidence ridge count

4.3.4 Angular Distortion

From the previous discussion, we know that the thresholds for distance and ridges crossed are higher because of the distortion in fingerprints. Hence, it is possible for an acute angled triplet to match an obtuse angled triplet based on just distances and ridges crossed. However, the difference between the corresponding angles should not be too high. For example, a triple with one of its angles as 30° should not match a triple whose same angle is 110°. In Figure 4.5, some examples of triples matched based on distance and ridges crossed are shown. The triples in Figure 4.5 (a) match with each other and triples in Figure 4.5 (b) match with each other based on distances and ridges crossed as matching criteria. Δpqr matches Δijk in both (a) and (b). Angle q

is an acute angle and angle *j* is an obtuse angle in both cases. Though these triangles are dissimilar, they match because of the higher thresholds of distance and ridge counts. Though Figure (a) can be considered a match because the triples look similar, Figure (b) cannot be considered a match. For this, internal angles of the triples are used in comparison and threshold is calculated. Angular distortion is analyzed for fingerprints of the same finger (35 different fingerprints are considered) and the maximum angular distortion between two corresponding angles of any two triples that ideally match is observed to be 65°. Most of the matching triples have a very small angular distortion of 5°-15°. But there are some triples whose angular distortion is higher because of movement of minutiae. Using angles as one of matching criteria eliminates some false matches. Hence, Figure 4.5 (b) becomes an invalid match.

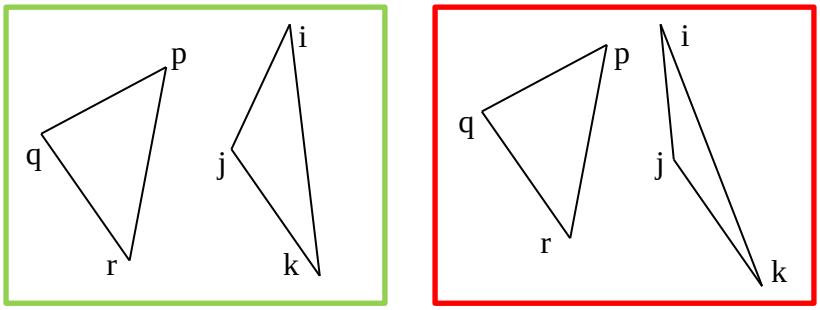

Figure 4.5 Angular distortion

4.3.5 Minutia Orientation Distortion

The orientation or direction of a minutia point is extracted as the angle made by the tangent to the ridge with the horizontal axis. This angle differs across fingerprints of the same finger. For example, it is observed that the orientation of minutia point in a fingerprint is 4 and it is 232 for the same minutia point in a different impression of the same finger. This distortion is mostly observed for ridge endings. By using direction differences of the minutiae in a triple, the number of false matches can be decreased. For a triple, three direction differences can be calculated (one for each side). Since, direction is not completely reliable, only one of the three direction differences is used as a matching criterion. A threshold is calculated after analyzing fingerprints of the same finger (35 different fingerprints are considered). A threshold of 40° is used.

4.4 Triplet-based Matching using Core Points

Since the thresholds for side lengths and ridges crossed are higher, each triple of the reference fingerprint has many matches in the target fingerprint. By using just side lengths and ridges crossed as matching criteria, different fingerprints also had higher matching scores. It was observed that most of the matches were false matches. Mining rare features requires matching criteria, which are neither stringent nor relaxed. By using a few matching criteria, it is not possible to extract rare features in fingerprints. The matching criteria should be relaxed such that two impressions of the same finger match should still match and they should be stringent enough to ensure that dissimilar triples do not match because of relaxed thresholds. The thresholds for distances, ridges crossed, angles and directions are already relaxed and these thresholds cannot be modified. So, other matching criteria should be used to reduce the number of false matches. Though these falsely matched triples look similar, they are scattered at different locations in the fingerprints. Level 1features (core-points and delta points) would help in identifying the position of the triple in the fingerprint. The rest of this section describes the novel parameters that are used for triplet matching.

4.4.1 Novel Parameters for Triplet-based Matching

Combination of Ridges crossed and Side lengths: In general, smallest side, largest side and sum of the sides of triples are used as matching criteria because comparing every side of reference triple with every side of target triple requires more comparisons. Using just these criteria, for two fingerprints of the same finger, some of the reference triples may not have exact matches. This is because, for some features, the smallest side is almost equal to the intermediate side or the largest side is almost equal to the intermediate side because of distortion and the corresponding sides are not compared. Similarly, sorting of ridges crossed also does not ensure that the corresponding ridges crossed are compared. In these cases, it is difficult to identify the corresponding matching minutiae in the triples. By identifying the corresponding minutiae between two triples, the amount of randomness can be reduced when the comparisons are

extended for 6-points and 9-points. It would also ensure that the corresponding sides, ridge counts, angles and directions are compared.

In this work, a combination of distance and ridge count is used to compare triples. Let *d1*, *d2*, *d3*, *rc1*, *rc2*, *rc3* be the three sides and three ridges crossed of a triple, then the combinations used for comparisons are (*d1*, *rc1*), (*d2*, *rc2*), (*d3*, *rc3*). Every combination from reference triple is compared with every combination of target triple. Hence, the combinations of (*d1*, *rc1*), (*d2*, *rc2*), (*d3*, *rc3*) can match the target triple in any order, so all such orders are considered. Let *r1*, *r2*, *r3* represent distance and ridges crossed combinations ((*d1*, *rc1*), (*d2*, *rc2*), (*d3*, *rc3*)) for reference triple and *t1*, *t2*, *t3* be the distance and ridges crossed combinations for target triple. There are six combinations in which a reference triple can match the target triple. The following are the six combinations.

1. *r1* matches *t1*, *r2* matches *t2*, *r3* matches *t3*
2. *r1* matches *t1*, *r2* matches *t3*, *r3* matches *t2*
3. *r1* matches *t2*, *r2* matches *t1*, *r3* matches *t3*
4. *r1* matches *t2*, *r2* matches *t3*, *r3* matches *t1*
5. *r1* matches *t3*, *r2* matches *t1*, *r3* matches *t2*
6. *r1* matches *t3*, *r2* matches *t2*, *r3* matches *t1*

The triples that have two sides and ridges crossed almost equal to each other (isosceles) usually match in 2 of the above combinations. The triples that have all sides and ridges crossed almost equal to each other (equilateral) match in all the above combinations. Other matching parameters discussed below are used to check for the valid combination after this step.

Core Distance: The centroids of all the triples are calculated and the distance between the centroid of the triple and the core-point is used as a matching criteria. These centroid distances are analyzed for different fingerprints of the same finger (35 different fingerprints are used). It is observed that the maximum distortion of core distance is 60 pixels and hence the threshold is set to 60 pixels. In Figure 4.6, the red-colored line joining core and centroid of the triple shows the core distance. For fingerprints with two cores, the core distances are calculated with respect to both the cores. When the fingerprints are not aligned, the location and the order in which the core points are extracted may not be the same for two fingerprints. So, rotation invariant core distance

should be used. For this, the distances from core 1 in reference fingerprint are compared with distances from core 1 and core 2 in target fingerprints and the triples are considered to match if reference core 1 distance matches with either core 1 or core 2 target distances. Similarly, two triples are considered to match if core 2 distance in reference triple matches with either core 1 or core2 target distances.

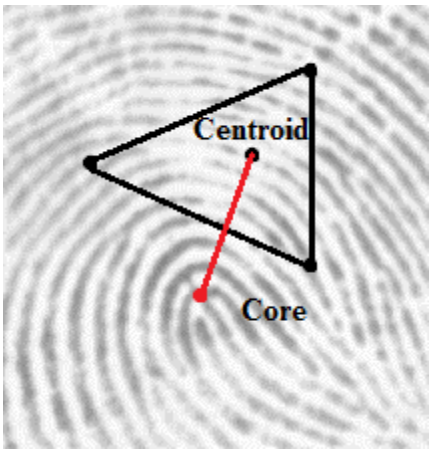

Figure 4.6 Distance between core and centroid of triple

Minutia Distance: The distances of all the vertices from the core point are calculated and used as matching criteria. By analyzing different fingerprints of the same finger, the maximum distortion is observed to be 60 pixels and the threshold is set to the same. In Figure 4.7, the distance between core point and the minutiae of a triple is shown by red-colored lines. For fingerprints with 2 cores, if minutiae distances from core 1 and core 2 of reference triple match with minutiae distances from either core 1 or core 2 of target triple, then the two triples are considered to be a match.

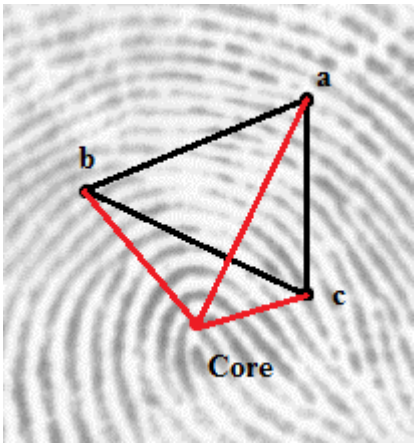

Figure 4.7 Distance of minutiae from core

Farthest vertex from the core: For all the triples, the vertex which is farthest from the core is calculated as the one whose distance from the core is greater than that of the other two vertices. In cases where the height of the triple is small or the distances of all the vertices from the core are almost equal, the farthest vertex cannot be definitely set and it is set to 0 in such cases. In Figure 4.8, the farthest vertex from core is 1.

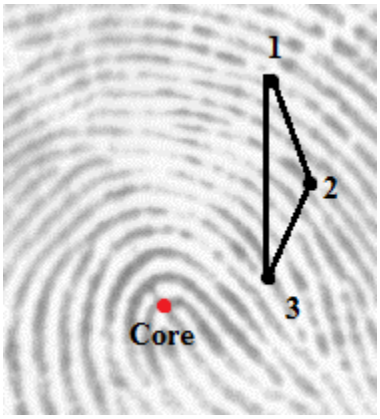

Figure 4.8 Farthest and nearest vertices

Nearest vertex to the core: For all the triples, the vertex which is closest to the core is calculated as the one whose distance from the core is lesser than that of the other two vertices. In cases where the height of the triple is small or the distances of all the vertices from the core are almost equal, the farthest vertex from the core is set to 0. In Figure 4.8, the nearest vertex to the core is 3.

For fingerprints with 2 cores, farthest and nearest vertices are identified for both the cores. If the farthest and nearest vertices from core 1 and core 2 of reference triple match with the farthest and nearest vertices respectively either from core 1 or core 2 of target triple, then those triples are considered to match.

Delta distance: The distance between the centroid of the triples and the delta point is calculated and used as a matching criterion. The threshold is calculated by analyzing different impressions of the same finger and the maximum distortion is observed to be 55 pixels and the threshold is set to the same. It should be noted that all the fingerprints do not have delta points. So, the delta distance is compared only if both reference and target fingerprints have delta points. For fingerprints with 2 cores, if delta distances from core 1 and core 2 of reference triple match with delta distances from either core 1 or core 2 of target triple, then those triples are considered to match. In Figure 4.9, the delta distance is shown by a red-colored line joining delta point and the centroid of triple.

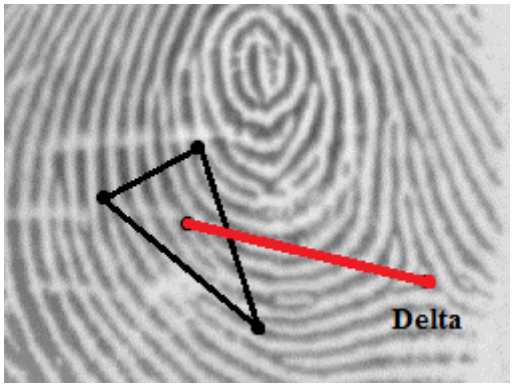

Figure 4.9 Delta distance

The parameters core distances, minutiae distances, farthest and nearest vertex to the core help in identifying the location of triple.

Handedness of the triple: In Figure 4.10(i) a reference triple (a) and a target triple (b) that incorrectly match are shown. Here (b) is the reflection of (a). Even when fingerprints are rotated, these triples should not match. The cross product of the shortest side and the longest side is calculated for both triples and sign of the cross product is used to compare the two triples. Cross Product is given by $\overline{pq}$ x $\overline{pr}$. In the Figure 4.10(i), the cross product of the reference triple is positive while the cross product of the target triple is negative and hence they do not match.

Figure 4.10(ii) shows another example where the cross products are positive for both reference and target triples and they match. The handedness is not valid for triples with any of its two sides almost equal. Also there should be a minimum difference between the smallest side and the intermediate side, intermediate side and largest side. Also, it is possible that the minutiae points may be collinear or almost collinear. In such a case, for the reference triple Δpqr in Figure 4.10(i), the minutiae point p could be on either side of $\overline{qr}$. So, the possibility of collinearity is also checked and if it is exists, such triples are not considered.

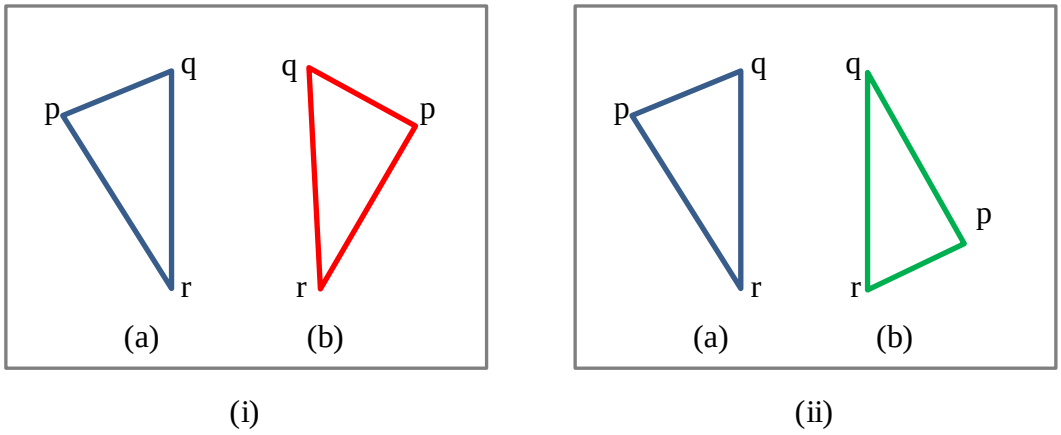

Figure 4.10 (i) Incorrectly matched triplets because (b) is reflection of (a). (ii) Correctly matched Triplets

High confidence and low confidence ridge counts: From section 4.3.3, we know that the ridges crossed between any two minutiae points is divided into two types, high confidence and low confidence and they are used as a matching criterion.

Sum of ridge counts: Sum of ridge counts is also used as a matching criterion. It eliminates many false matches; hence it is used in the beginning before all the other matching criteria.

Internal angles: The slopes of the sides of the triples (*m1*, *m2*, *m3*) are calculated and the angles of inclination ($\theta 1$, $\theta 2$, $\theta 3$) are calculated for all the sides. The internal angles $\alpha 1$, $\alpha 2$, $\alpha 3$ are calculated as the difference of the angles of inclination of the sides forming that angle. The internal angles of the triple are analyzed for different fingerprints of the same finger and it is observed that the maximum angular distortion is 65°. Let (*x1*, *y1*), (*x2*, *y2*), (*x3*, *y3*) represent

coordinates of the minutiae of the triple. The slopes, angles of inclination, internal angles are given by the following equations.

$$m_1 = \frac{(y_2 - y_3)}{(x_2 - x_3)}, \qquad \theta_1 = \tan^{-1} m_1$$

$$m_2 = \frac{(y_3 - y_1)}{(x_3 - x_1)}, \qquad \theta_2 = \tan^{-1} m_2$$

$$m_3 = \frac{(y_2 - y_1)}{(x_2 - x_1)}, \qquad \theta_3 = \tan^{-1} m_3$$

$$\alpha_1 = \min(abs(\theta_2 - \theta_3), 360 - abs(\theta_2 - \theta_3))$$

$$\alpha_2 = \min\big(abs(\theta_1 - \theta_3), 360 - abs(\theta_1 - \theta_3)\big)$$

$$\alpha_3 = \min(abs(\theta_2 - \theta_1), 360 - abs(\theta_2 - \theta_1))$$

Direction of Minutia: The direction of minutia is also considered as one of the matching criteria. For any side of the triple, the difference in the direction of minutiae forming the side is considered as the direction of that side. The direction is calculated for all the three sides of the triple. However, the comparison is only performed for one among the three direction differences because the direction of minutiae is not reliable in some cases. The direction difference of a line segment joining minutiae m and n is calculated as follows

$$Direction = min(abs(\theta_m - \theta_n), (360\ abs\ (\theta_m - \theta_n))$$

Here θ_m and θ_n are the directions of minutiae m and n respectively.

Selection of best order: Though some parameters of the triple require higher thresholds for matching, other parameters of the triple do not have much distortion. For example, consider a triple with side lengths 80, 99, 120 and ridges crossed 5, 5, 9. Consider the same triple in a different fingerprint of the same finger whose side lengths are 104, 110, 130 and ridges crossed are 4, 5, 10. Though the difference between the corresponding side lengths is higher, the difference of corresponding ridges crossed is lower. So, the distortion is not on the same levels for all the parameters.

When two triples are matched in more than 1 order, the other parameters can be compared to select the best order. However, all the parameters cannot be considered to check the amount of variation in the fingerprints. Angles cannot be considered because of greater distortion and direction is not reliable. Distance of vertices from a core and ridges crossed are considered to select the best order. For fingerprints with 2 cores, the matching cores are first identified. This is done simply by checking the minutiae distances from both the cores for all the orders. If the minutiae distances from reference core1 match minutiae distances from target core2 for all the orders, then reference core 1 and target core 2 are the matching cores. The amount of variation is calculated for all the 6 parameters (3 minutiae distances + 3 ridges crossed) for all the orders. It is the difference between corresponding parameters of the two triples. Consider two triples that match in orders 1 and 2. Let ref_rc1, ref_rc2, ref_rc3, ref_leg11, ref_leg12, ref_leg13 be the ridges crossed and the distance of vertices from the core point for reference triple and let tar_rc1, tar_rc2, tar_rc3, tar_leg11, tar_leg12, tar_leg13 be the ridges crossed and the distance of vertices from the core point for target triple. The amount of variation for both the orders is calculated as follows. For first order, differences are calculated as abs (ref_rc1 – tar_rc1), abs (ref_rc2 – tar_rc2), abs (ref_rc3 – tar_rc3), abs (ref_leg11 – tar_leg11), abs (ref_leg12 – tar_leg12), abs (ref_leg13 – tar_leg13). For second order, the differences are calculated as abs (ref_rc1 – tar_rc1), abs (ref_rc2 – tar_rc3), abs (ref_rc3 – tar_rc2), abs (ref_leg11 – tar_leg11), abs (ref_leg12 – tar_leg13), abs (ref_leg13 – tar_leg12). The corresponding differences of first and second orders are compared. The order with less distortion is selected as the final order of that reference triple. Similarly, the best order is selected when the triples match in more than 2 orders.

The matching criteria used for 3-point comparisons and their thresholds are listed out in Table 4.1.

Encoding Matched Combinations: By using the parameters above, the reference triple can match the target triple in any combination or order. Since there are 6 combinations, the naïve method to store them would be to use an array of 6 elements for every combination of reference and target triples that match. To reduce the data complexity, these 6 combinations are encoded. The total number of combinations possible with 6 elements is 63. (The triples can match in 1 or 2 or 3 or 4 or 5 or 6 combinations. Total = 6C1 + 6C2 + 6C3 + 6C4 + 6C5 + 6C6).

Table 4.1 Matching criteria for 3-point comparisons

Parameters	**Threshold**
Side lengths or Distance	35% for smaller distances. Gradually decreases with distance. 20% for larger distances
Ridge Count	High confidence – 5, Low confidence – 12
Core distance	60 pixels
Minutia Distance	60 pixels
Delta Distance	55 pixels
Farthest Vertex from Core	-
Nearest Vertex to Core	-
Handedness of Triple	-
Internal Angles	65°
Direction of Minutia	40°

All the combinations and their encoded indices are shown in Figure 4.11. For example, if the triples match in 1, 2, 3, 4 combinations, then they are stored as 45. These combinations are later decoded in 6-point and 9-point comparisons.

0 – {1}	16 – {3, 5}	32 – {2, 3, 5}	48 – {1, 3, 4, 6}
1 – {2}	17 – {3, 6}	33 – {2, 3, 6}	49 – {1, 3, 5, 6}
2 – {3}	18 – {4, 5}	34 – {2, 4, 5}	50 – {1, 4, 5, 6}
3 – {4}	19 – {4, 6}	35 – {2, 4, 6}	51 – {2, 3, 4, 5}
4 – {5}	20 – {5, 6}	36 – {2, 5, 6}	52 – {2, 3, 4, 6}
5 – {6}	21 – {1, 2, 3}	37 – {3, 4, 5}	53 – {2, 3, 5, 6}
6 – {1, 2}	22 – {1, 2, 4}	38 – {3, 4, 6}	54 – {2, 4, 5, 6}
7 – {1, 3}	23 – {1, 2, 5}	39 – {3, 5, 6}	55 – {3, 4, 5, 6}
8 – {1, 4}	24 – {1, 2, 6}	40 – {4, 5, 6}	56 – {1, 2, 3, 4, 5}
9 – {1, 5}	25 – {1, 3, 4}	41 – {1, 2, 3, 4}	57 – {1, 2, 3, 4, 6}
10 – {1, 6}	26 – {1, 3, 5}	42 – {1, 2, 3, 5}	58 – {1, 2, 3, 5, 6}
11 – {2, 3}	27 – {1, 3, 6}	43 – {1, 2, 3, 6}	59 – {1, 2, 4, 5, 6}
12 – {2, 4}	28 – {1, 4, 5}	44 – {1, 2, 4, 5}	60 – {1, 3, 4, 5, 6}
13 – {2, 5}	29 – {1, 4, 6}	45 – {1, 2, 4, 6}	61 – {2, 3, 4, 5, 6}
14 – {2, 6}	30 – {1, 5, 6}	46 – {1, 2, 5, 6}	62 – {1, 2, 3, 4, 5, 6}
15 – {3, 4}	31 – {2, 3, 4}	47 – {1, 3, 4, 5}	

Figure 4.11 Encoding the combinations

4.4.2 Algorithm for 3-point Comparisons

The algorithm for 3-point comparisons is detailed in Figure 4.12.

```
For every triple in reference fingerprint
  For every triple in target fingerprint
    Check Sum of Ridge counts and the Core distances
      Check all Distance and Ridges Crossed combinations
        For all combinations of Distance and Ridges crossed that are true
          Check Minutiae Distances from core, Angles, Direction Differences
            For all combinations that are true from above step
              Check if the target triple is a reflection of reference triple
                Check Farthest Vertex from Core
                  Check Nearest Vertex to Core
                    If Fingerprints have delta points, check delta distance
                      Select the best combinations in which reference and target triples match
                        Check the matching score for the combinations
                          Triples match
```

Figure 4.12 Algorithm for triple comparisons

4.4.3 Parameters for 6-point and 9-point Comparisons

As discussed before, a 6-point is formed by the combination of 2 triples and a 9-point is formed by the combination of 3 triples. The results of triple matches are used for 6-point and 9-point comparisons. For every reference triple, the index of the matching target triple and the combination in which it matches is stored. The matching criteria used for 6-point and 9-point comparisons are given below.

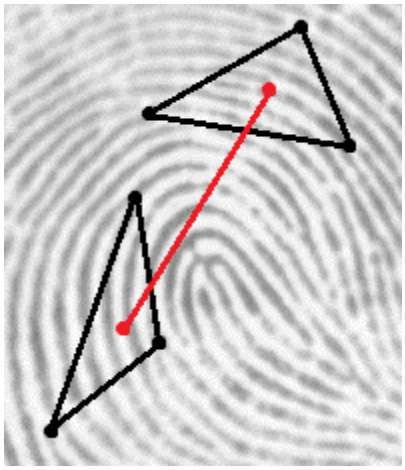

Figure 4.13 Centroid distance shown for a 6-point feature

Centroid Distance: The centroids are calculated for all the triples in a 6-point feature and a 9-point feature and the distance between the centroids of all the triples is calculated. In Figure 4.13 and Figure 4.15, the centroid distances for 6-point and 9-point features respectively are shown. Similar to the threshold for the side lengths of triples, the centroid distance is not set to a constant. It gradually decreases with the distance. The lower centroid distances have higher threshold and higher centroid distances have lower threshold. After analyzing different impressions of the same finger, threshold values are calculated. The graph showing the thresholds used for centroid distances is shown in Figure 4.14.

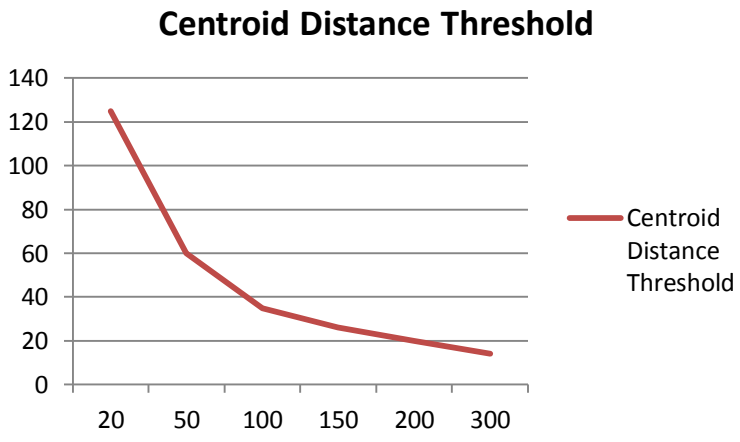

Figure 4.14 Centroid distance thresholds

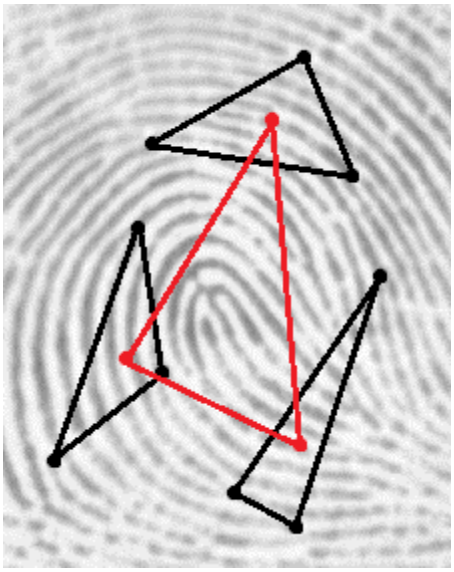

Figure 4.15 Centroid distances shown for a 9-point feature

Minutiae Distances: The distance between the all the minutiae in the 6-point and 9-points are calculated. Since the order in which a reference triple matches a target triple is already known, the corresponding distances of a reference combination (either 6-point or 9-point) are compared with the distances of a target combination. In Figure 4.16, the comparison of a reference with a target 6-point combination is shown. In the figure, the order in which reference triple 1 matches target triple 1 is 2 and the order in which reference triple 2 matches target triple 2 is 4. So, minutiae 1, 2, 3 from reference triple 1 match minutiae 1′, 3′, 2′ from target triple 1 respectively. And, minutiae 4, 5, 6 from reference triple 2 match minutiae, 5′, 6′, 4′ from target triple 2 respectively. The minutiae distances which are compared are ($\overline{14}$ vs $\overline{1'5'}$), ($\overline{15}$ vs $\overline{1'6'}$), ($\overline{16}$ vs $\overline{1'4'}$), ($\overline{24}$ vs $\overline{3'5'}$), ($\overline{25}$ vs $\overline{3'6)'}$, ($\overline{26}$ vs $\overline{3'4'}$), ($\overline{34}$ vs $\overline{2'5'}$), ($\overline{35}$ vs $\overline{2'6'}$), ($\overline{36}$ vs $\overline{2'4'}$). Without identifying the corresponding minutiae in the triples, each minutia distance from reference 6-point has to be compared with every minutiae distance from target 6-point. But by using the method in this work, the corresponding minutiae are identified and the randomness in matching the 6-points and 9-points can be reduced.

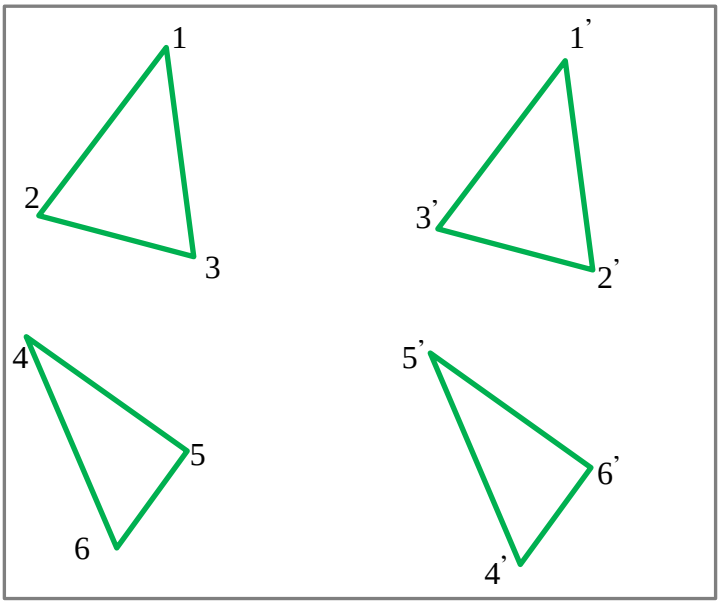

Figure 4.16 Matched 6 point features

Ridges Crossed across all minutiae: The number of ridges crossed across all the 6-point and 9-point combinations is used as a matching criterion. The thresholds for high confidence and low confidence ridge counts are same as those for 3-point comparisons.

Farthest and nearest minutia point: Figure 4.17 shows incorrectly matched 6-point and 9-point features. These features match because of the higher thresholds that were considered for distance and ridge counts. The triples marked in red are the target triples which almost look like reflections of their corresponding reference triples. Many of such features are already eliminated by the farthest and nearest vertex to core point from triple comparisons (3-point). To eliminate any other features that might have been missed, for each 6-point and 9-point feature, farthest and nearest minutiae point with respect to the centroid of the other triple are used. In Figure 4.17 (i), the farthest minutia in Δabc with respect to the centroid of Δpqr is minutia *b*, but the farthest point in Δabc of target with respect to Δpqr is *a*. Since the farthest points with respect to Δpqr are not equal, it can be concluded that these 6-points feature do not match. In the 9-point feature in Figure 4.17 (ii), Δpqr and Δabc are the already matched 6-point features. There is no definite farthest point in Δijk with respect to Δpqr because both *j* and *k* are almost equidistant to Δpqr. But there exists a farthest point in Δijk with respect to Δabc which is minutia *k* in reference. In target, the farthest point in Δijk with respect to Δabc is *j*. Since the farthest points with respect to Δabc do not match, it can be concluded that the 9-point features do not match. The nearest points are calculated only if the results from the farthest point calculation are true.

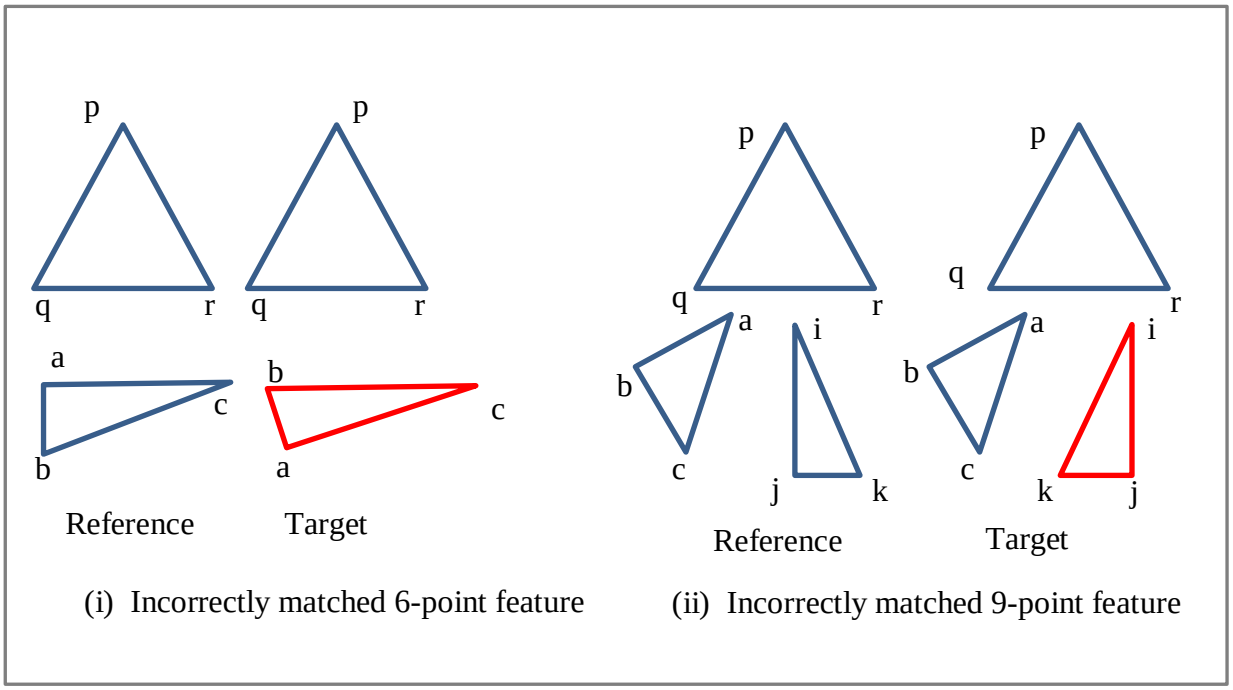

Figure 4.17 Incorrectly matched 6-point and 9-point features

The matching criteria used for 6-point and 9-point comparisons and their thresholds are shown in Table 4.2

Table 4.2 Matching criteria for 6-point and 9-point comparisons

Parameter	Threshold
Centroid Distance	Gradually decreases with distance
Minutiae Distances	Gradually decreases with distance
Ridges crossed	High confidence – 5, Low confidence - 12
Farthest Minutia point	-
Nearest Minutia point	-

4.4.4 Algorithm for 6-point Comparisons

The algorithm for 6-point comparisons is shown in Figure 4.18. Whenever a condition is checked and if it is not satisfied, then the reference and target 6-points do not match and the control exits the loop.

For every i^{th} reference triple

For every $(i+1)^{th}$ reference triple

Check if i^{th} and $(i+1)^{th}$ reference triples form a valid 6-point feature

For every match of i^{th} triple in target

For every combination in which target triple matches i^{th} reference triple

For every match of $(i+1)^{th}$ triple in target

For every combination in which target triple matches $(i+1)^{th}$ reference triple

Check if centroid distances of reference and target 6-points match

Check if Ridge Counts of reference and target 6-points match

Check if Minutiae distances of reference and target 6-points match

Check if Farthest and Nearest minutiae points of reference and target 6-points match

6-points match and save the indices of reference and target triples

Figure 4.18 Algorithm for 6-point comparisons

4.4.5 Algorithm for 9-point Comparisons

The algorithm for 9-point comparisons is shown in Figure 4.19. The results from 6-point comparisons are used for 9-point comparisons. Every 6-point feature is combined with a 3-point feature to form a 9-point feature. For a reference and target 9-point features to match, 3 centroid distances, 27 Minutiae distances, 27 Ridge Counts should be compared. But only 2 centroid distances, 18 Minutiae distances, 18 Ridge counts are compared because the triples in 6-point feature are already matched.

For all 6-point features (Formed by i^{th} and $(j)^{th}$ 3-point features) with matches

For every $(j+1)^{th}$ 3-point feature

Check if i^{th}, $(j)^{th}$ and $(j+1)^{th}$ reference triples form a valid 9-point feature

For every match of $(j+1)^{th}$ triple in target

For every combination in which target triple matches $(j+1)^{th}$ reference triple

Check if centroid distances of reference and target 9-points match

Check if Ridge Counts of reference and target 9-points match

Check if Minutiae distances of reference and target 9-points match

Check if Farthest and nearest minutiae points of reference and target 9-points match

9-points match; go to the next 9-point feature

Figure 4.19 Algorithm for 9-point comparisons

4.4.6 Algorithm for Mining Rare Features

Some of the fingerprints in the FBI database have as many as 100 minutiae. With just 50 minutiae, number of 3-points formed is 19,600, number of 6-points formed is 15.89 million and the number of 9-points formed is 2.5 billion. Because of the data complexity and longer run-times, a small set of 35 high quality minutiae is considered. With 35 minutiae, number of 3-points formed is 6545, number of 6-points formed is 1.6 million and the number of 9-points formed is 70.6 million. As discussed before, there is a limit on the smallest side and the largest side of the triple. Triples with smallest distance >= 25 and largest distance <= 200 are considered. The limit on the smallest distance is chosen to be 25 because the minutiae distances that are <25 have as high as 100% distortion (section 4.3.1). The limit on the largest distance is chosen to be 200 to form smaller local structures (triples) which are lesser prone to distortion [22]. Also, a limit on the largest angle of triple is used. Some triples are in the form of a straight

line (largest angle 180°) and the matching triple in another impression of the same finger forms a triangle. It is observed that these triples have higher number of matches and hence all the triples with largest angle > 170° are not considered. Finally, the number of triples considered is also limited to 1500. The algorithm for mining rare features in a fingerprint is shown in Figure 4.20. In the first step, the reference fingerprint is compared with a few target fingerprints and only 3-point features and 6-point features are compared. This is to eliminate some of the 6-point features that have more matches. Such 6-point features may not be rare features and it is less likely that these 6-points might form a rare 9-point feature. After the first step, a set of 3000 unmatched 6-points is considered and all possible 9-point combinations are formed. A limit of 1million is chosen for the number of 9-point combinations considered. All the limits are chosen to reduce data complexity and longer run-times. It should be noted that because of these limits, it is possible to miss some of the rare features from identification. However, we were able to identify a small set of rare features for every fingerprint with these limits. The 9-point combinations thus formed are the final set of features that are compared across the fingerprints in the database to find rare features. The number of 9-points matched is checked after every 100 files to check if there are any unmatched features left. If all the 9-points have matches, then there are no rare features in the selected 9-points of the reference fingerprint. Also, for every new fingerprint, the number of core points is checked to ensure that fingerprints of the same type are compared.

4.4.7 Database Profiling

The FBI database consists of single-core (right loop or left loop), two-core (whorl and two loops) and arch-type (tented arch and arch) fingerprints. The feature extracting tool [38] that was used extracts all the minutiae and core-points. To verify that the number of cores extracted from the fingerprint is true, different impressions of the same fingerprint with high quality are analyzed. The database is formed by considering all the good quality fingerprints with true core points. For the fingerprints with arch-type core, core points were not extracted by the tool. The core points for all the arch-core fingerprints were identified manually. A database consisting of 11,036 fingerprints whose location and number of core points are reliable is formed. This database consists of 2143 two-core, 8313 single-core and 580 arch-type fingerprints. It should be noted

that 18,649 fingerprints from FBI database are used in Chapter 3; however, because of unreliable detection of location and number of core points, only 11,036 fingerprints from this database are used for mining rare features. All the fingerprints in the database are manually checked to verify the number of core points. The reference fingerprints that are used to mine rare features and target fingerprints used in comparisons are selected from these 11,036 fingerprints from FBI database.

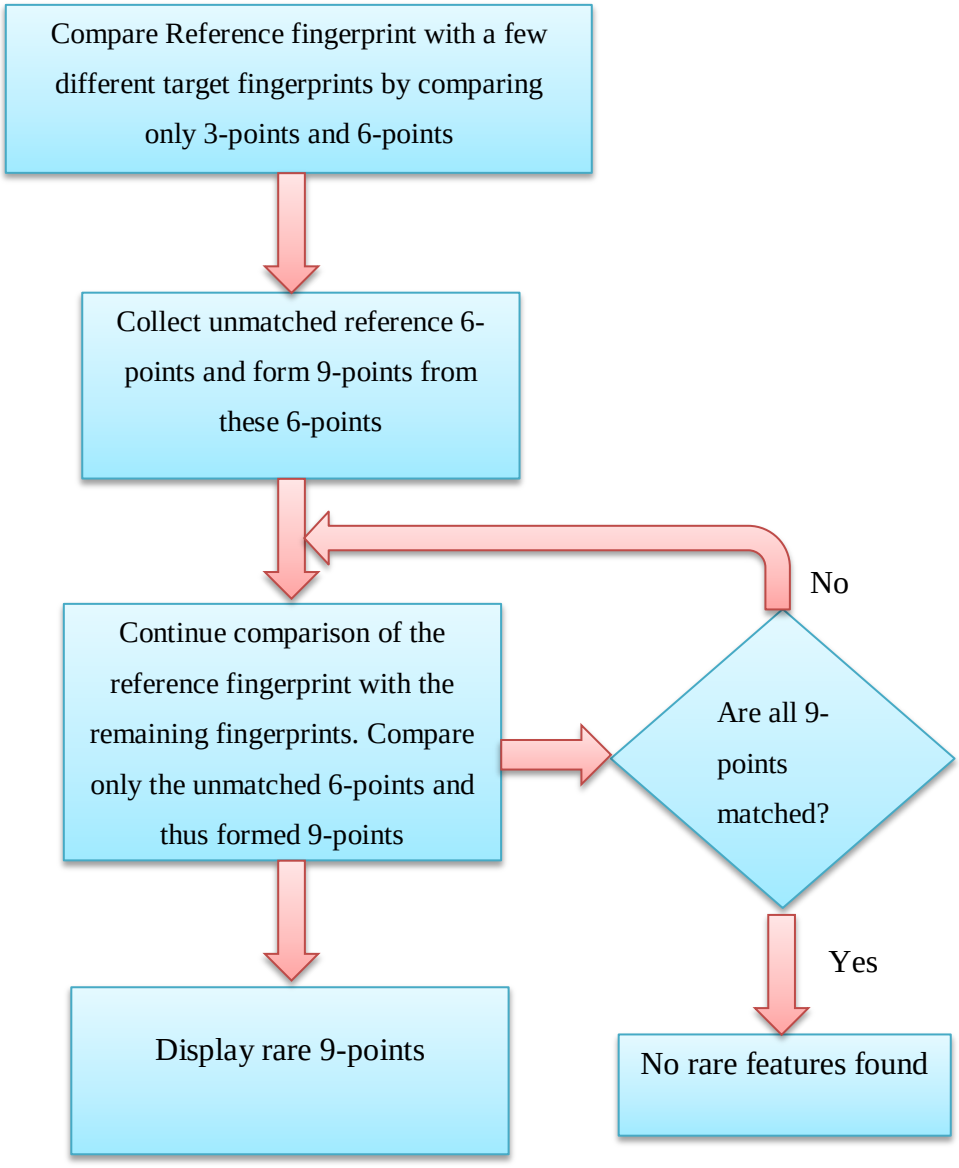

Figure 4.20 Algorithm for mining rare features

4.4.8 Results

For each core type, 10 fingerprints are randomly selected from the FBI database formed with 11,036 fingerprints (section 4.4.7). So, totally 30 fingerprints are selected for mining rare features. Although only 30 fingerprints are selected, rare features that are unique to some *N* fingerprints can be identified in any randomly selected fingerprint by using this method. These 30 fingerprints are used as reference and the database is searched to identify statistically rare features in these fingerprints. It is observed that for all core types, rare 3-point features and rare 6-point features do not exist. But rare 9-point features exist. For a reference fingerprint, as the number of fingerprints compared increases, number of 9-point features with matches also increases. From a million 9-point features, a few features are left after comparing with hundreds of fingerprints. The number of unmatched 9-point features left after comparing with every 100, 500, 1000, 1500 fingerprints is shown in the results below. For a 9-point feature which is left unmatched after comparing with 1000 fingerprints, it can be concluded that such feature occurs only once in 1000 fingerprints which makes it a rare feature. It should be noted that this 9-point feature may find a match if a slightly different database is used. But, the probability of finding a match for this 9-point in randomly sampled 1000 fingerprints is less.

For all the tables below, the number of minutiae considered for the reference fingerprint and the number of 3-points formed from those minutiae are shown. In the third column, the number of identical-core fingerprints compared with the reference is given. The fourth column represents the sum of the number of identical-core fingerprints and fingerprints of other core types. The reference fingerprint is compared with the entire database, but whenever a fingerprint of a different core type is encountered, the feature matching is not performed because of core dissimilarity and the next fingerprint is compared. The table also gives the number of 9-point features considered for that fingerprint and rare 9-point features are given in the last column.

Results for Arch-Core Fingerprints: The results for arch-type fingerprints are shown in Table 4.3. Each row represents a reference fingerprint. For Fingerprint 1 (with 35 minutiae and 1164 triples, from a set of 128638 9-point features), 115 features are left unmatched (rare features) after comparing with 580 arch-core fingerprints. It can be observed from the table that fingerprints 1-4 have some rare 9-point features left after comparing them with the entire

database. Fingerprint 4 has 545 rare 9-point features when a set of 1 million 9-point features is considered. The fingerprints 1-6 have at least 1 rare feature in 500 arch-core fingerprints and all the fingerprints have at least 1 rare feature in 100 fingerprints (see fourth column). For example, fingerprints 5 and 6 are compared with 502 and 536 arch-core fingerprints respectively and 1 rare 9-point feature is identified for both the fingerprints. So, both these fingerprints have at least 1 rare feature when compared with 500 arch-core fingerprints.

Table 4.3 Rare 9-point features in arch-core type fingerprints

Fingerprint	**# Minutia**	**#3-points**	**#Arch-core fingerprints compared**	**# Fingerprints compared**	**#9-points considered**	**Rare 9-points**
1	35	1164	580	11036	128638	115
2	35	1500	580	11036	882863	15
3	35	1500	580	11036	233522	4
4	34	1500	580	11036	1000000	545
5	35	1164	502	10958	166420	1
6	33	1500	536	10992	921097	1
7	35	1500	400	10856	1000000	1
8	35	938	200	10656	471692	16
9	29	970	100	10556	784687	3
10	34	956	100	10556	215473	16

In Table 4.4, the number of 9-point features that are left unmatched after comparing with fingerprints of arch-core type at different intervals are shown. Since the number of arch-core type fingerprints is 580 in the database, two intervals, one at 100 and another at 500 are shown. It can be seen from the table that for fingerprint 1, 2,856 9-point features are left unmatched after comparing with 100 fingerprints and 123 9-point features are left unmatched after comparing with 500 fingerprints.

Table 4.4 Rare 9-point features in arch-core fingerprints at different fingerprint intervals

Fingerprint	100 Fingerprints compared	500 Fingerprints compared
1	2856	123
2	24353	15
3	4063	12
4	37857	665
5	2432	1
6	742	1
7	4505	0
8	459	0
9	3	0
10	16	0

Table 4.5 Rare 9-point features in two-core fingerprints

Fingerprint	# Minutia	#3-points	#Two-core fingerprints compared	# Total fingerprints compared	#9-points considered	Rare 9-points
1	35	1150	2143	11036	660919	2
2	32	1500	1964	10857	638523	1
3	35	1398	1027	9920	250809	1
4	35	1197	1439	10332	643892	1
5	35	892	500	9393	581095	3
6	35	1500	500	9393	632564	2
7	35	1500	500	9393	947147	6
8	34	1500	200	8923	717588	1
9	35	775	700	9423	273409	6
10	35	1500	400	9123	1000000	1

Results for Two-Core Fingerprints: The results for two-core fingerprints are shown in Table 4.5. It shows that fingerprint 1 has 2 rare features when compared with the entire database. For fingerprints 1-4, at least 1 rare feature was found when compared with 1000 fingerprints. For fingerprints 1-7, at least 1 rare feature was found in 500 fingerprints.

For two-core fingerprints, Table 4.6 shows the number of unmatched 9 point features identified after comparing with 100, 500, 1000, 1500 fingerprints.

Table 4.6 Rare 9-point features in two-core fingerprints at different fingerprint intervals

Fingerprint	100 fingerprints compared	500 fingerprints compared	1000 fingerprints compared	1500 fingerprints compared
1	286590	2585	301	80
2	68777	1107	26	1
3	12010	74	1	0
4	23668	31	1	0
5	15765	3	0	0
6	14603	2	0	0
7	5807	6	0	0
8	41	1	0	0
9	48679	29	0	0
10	3356	0	0	0

Results for Single-Core Fingerprints: The results for single-core fingerprints are shown in Table 4.7. It shows that for 1-2 fingerprints, at least 1 rare feature was found when compared with 1000 fingerprints. For the fingerprints from 1-6 and 10, at least 1 rare feature was found when compared with 500 fingerprints.

For single-core fingerprints, Table 4.8 shows the number of unmatched 9 point features present after comparing with 100, 300, 600, 900, 1000 fingerprints.

Table 4.7 Rare 9-point features in single-core fingerprints

Fingerprint	# Minutia	#3-points	#Single-core fingerprints compared	# Total fingerprints compared	#9-points considered	Rare 9-points
1	29	1437	1300	4023	1000000	5
2	35	1500	1000	3723	1000000	3
3	35	1500	900	3623	706722	15
4	35	1500	800	3523	787825	16
5	35	1500	500	3223	1000000	11
6	28	896	800	3523	790150	1
7	30	811	100	2823	761774	231
8	33	1500	100	2823	1000000	12
9	35	1500	300	3023	1000000	1
10	35	1500	900	3623	818980	1

Table 4.8 Rare 9-point features in single-core fingerprints at different fingerprint intervals

Fingerprint	100 Fingerprints compared	300 Fingerprints compared	600 Fingerprints compared	900 Fingerprints compared	1000 Fingerprints compared
1	63472	940	17	7	7
2	169299	28444	503	4	3
3	15625	380	19	15	0
4	223982	3468	75	0	0
5	26297	379	0	0	0
6	21962	279	15	0	0
7	231	0	0	0	0
8	15	0	0	0	0
9	159	1	0	0	0
10	40266	470	14	1	0

From the above results, it can be seen that arch-core fingerprints have more rare features compared to single-core and two-core fingerprints. This is because of their lower proportion in the database. Also two-core fingerprints have more rare features compared to single-core fingerprints because of the comparatively lower proportion of two-core fingerprints in the database and also because of more matching criteria; with 2 cores, total 6 minutiae distances of which 3 are from a core are considered and with 1 core, only 3 minutiae distances from the core are considered. The matching criteria of two-core fingerprints help in locating the position of triple in a fingerprint more accurately.

In Figures 4.21 and 4.22, some rare 9-point features identified in a single-core fingerprint are shown. The fingerprint used in this figure is taken from SD-27 (Special Database).These rare features are statistically rare 9-point features.

From the above results, for fingerprints of different cores, it can be seen that some of the fingerprints have at least one rare feature when compared with 1000 fingerprints of identical core. This means that this rare feature occurs only once in 1000 fingerprints. Also, these 30 fingerprints have many rare 9-point features when compared with a small set of 100 fingerprints. The rare features identified by the above algorithm are statistically rare features. These rare features can be very valuable when they are identified in the latent fingerprint. They will boost the confidence of the evidence and help in making a confident court judgment. It should be noted that the rare features identified in N randomly sampled fingerprints may find a match in a set of different N randomly sampled fingerprints. There is always a probability of finding a match for these rare features. However, their occurrence would be still statistically rare. Even if these features find a match in a different set of fingerprints, they will have a few matches which still makes them rare.

The algorithm used in this work can be applied to any fingerprint database to extract rare features. The tolerance parameters are relaxed enough to ensure that impressions of the same finger match, so they need not be changed when a new database is used.

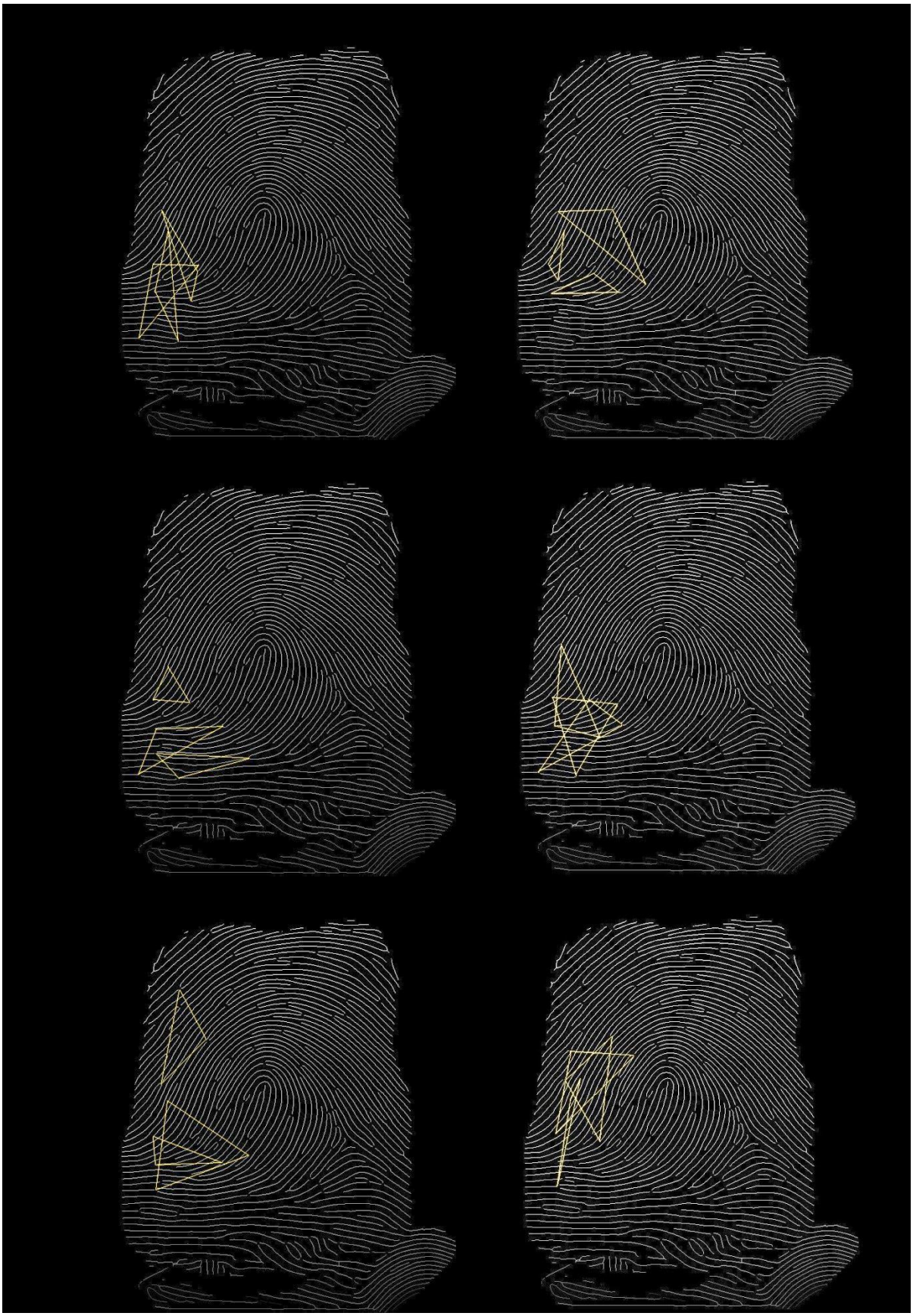

Figure 4.21 Rare 9-point features identified in a single-core fingerprint

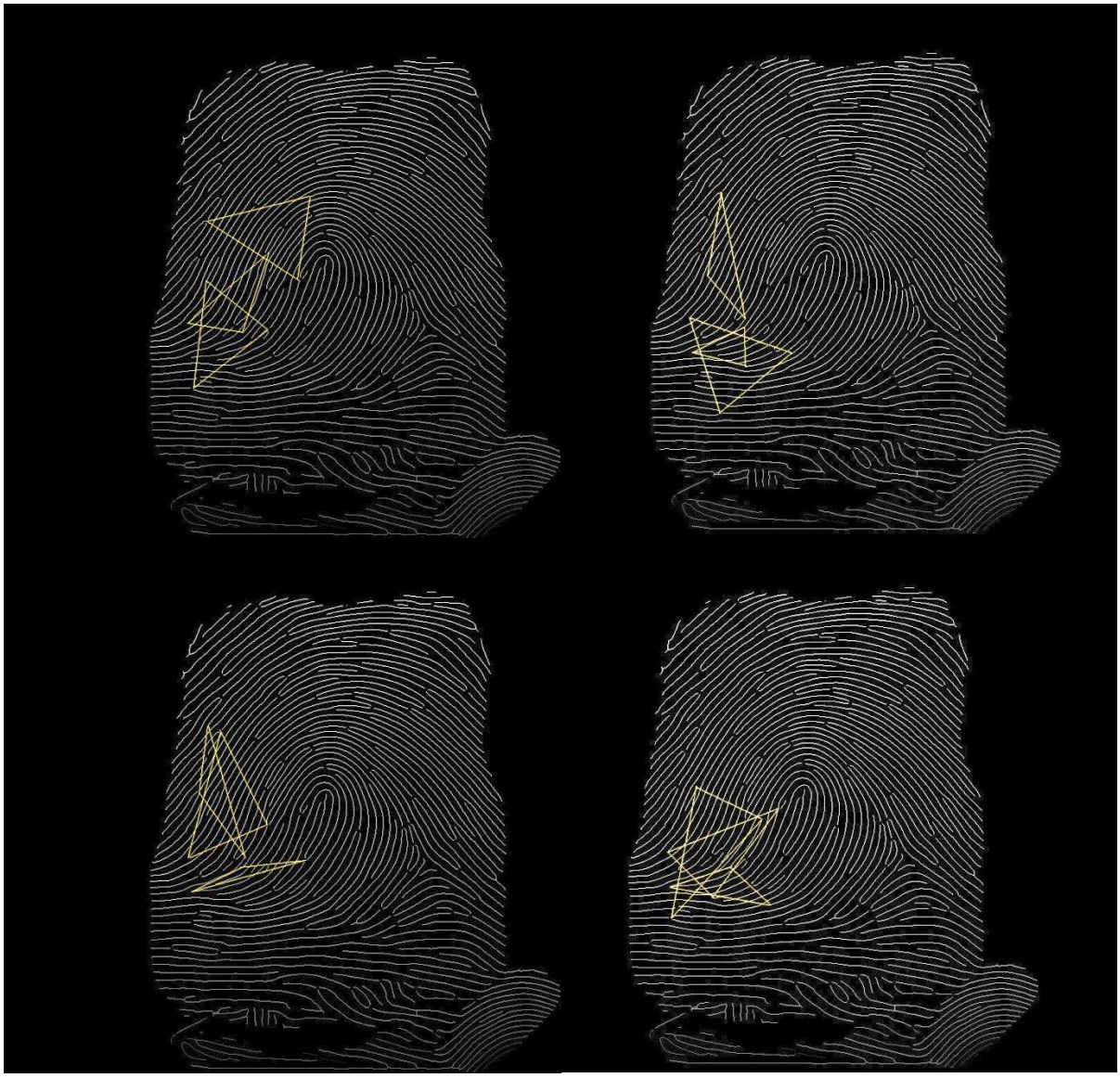

Figure 4.22 Additional rare 9-point features identified in a single-core fingerprint

Chapter 5: Conclusions and Future Work

In this thesis, a method was proposed to accelerate fingerprint triplet-based matching techniques using GPUs. Also, another method was proposed for mining rare features in fingerprints using core points.

In Chapter 3, an existing algorithm used for fingerprint matching is parallelized and implemented on GPUs. Shared Memory and other optimization techniques were used to accelerate the algorithm. The sequential and parallel algorithms are applied on a large database of 18,649 fingerprints and the execution times are compared. The parallel algorithm takes less than two hours whereas the sequential algorithm takes 60 hours. A speedup of 40.2 times was obtained by using the parallel algorithm. Thus, this method shows that using GPUs for fingerprint matching can greatly reduce the matching time for larger databases.

In Chapter 4, various kinds of distortion that are observed in fingerprints are discussed. It was observed that the distortion is higher for spatial distances and ridges crossed between minutiae and methods were proposed to cope with these distortions. By using higher thresholds, the number of matches between two fingerprints is large even though the fingerprints are from different fingers. To reduce the number of matches, usage of core points was proposed which helped in identifying the location of triples in the fingerprints and thereby reduced number of matches. A set of novel parameters was used for triplet matching and it was extended to 6-point and 9-point comparisons. This algorithm helped in eliminating false matches and it can be used as a fingerprint matching tool. The same algorithm was used for mining rare features in fingerprint and it was observed that in every fingerprint that was used for mining rare features, there were a small percentage of rare 9-point features. Some rare 9-point features that were identified in some of the fingerprints occur only once in 1000 fingerprints. These rare features when identified in a latent fingerprint can help in increasing the confidence of the evidence.

The rare features identified by this method are truly statistically rare and there is no possibility for false positives. This is because of the relaxed thresholds used. Although the rare feature can find a match in a slightly different database, it would still have a few matches which would still make that feature rare. However, there is a chance that some of the rare features might not have

been identified (presence of false negatives) because of relaxed thresholds. A feature which is rare might have found a match because of relaxed thresholds.

There is lot of scope for future work in this area. There are many 3-points, 6-points and 9-points that still have false matches. There is scope to reduce such false matches by using more matching parameters. Because of these false matches, a feature which may be originally rare may find a match in the database. The key to find rare features is to reduce the number of false matches.

Ridge Flow as Matching Parameter: The ridge flow patterns form the non-minutiae features of the fingerprints. These ridge flow patterns have higher discriminatory power when the triples are compared. A combination of minutiae and non-minutiae based features would help in reducing the false matches in fingerprint matching.

Minutiae Density as Matching Parameter: The neighboring minutiae around a triple can be used in some way to reduce the number of matches. A circle of some fixed radius can be considered around a triple allowing some tolerance levels and various characteristics of minutiae that fall in this circle can be analyzed such as the number of minutiae, distance, ridge counts, location of minutiae with respect to the core and the triple.

Quantifying Rarity: This thesis work shows that for the 30 fingerprints that were randomly selected to mine rare features have a small set of rare features. But how these rare features influence the matching score between two fingerprints has to be quantified. Depending on the number and occurrence of these rare features, a method has to be developed to understand their effect on the matching score. For example, a rare feature whose occurrence is 1 in 1000 fingerprints has higher confidence level than a rare feature whose occurrence is 1 in 100 fingerprints.

Parallelization using GPUs: By using the techniques used for parallelizing fingerprint matching algorithm discussed in Chapter 3, a new parallel algorithm can be developed for triplet based matching using core points and for mining rare features.

Calculation of Tolerance Parameters: In this work, to calculate the tolerance parameters, the corresponding minutiae were identified manually, which is the most time-consuming process. There is a need for a tool, which would identify the corresponding minutiae and calculate the

tolerances for all kinds of distortions. There are some existing methods [39, 40]which help in selecting the thresholds to some degree, but development of a fully functional tool would aid fingerprint matchers.

Reducing the search space of 9-point features: It was identified that some triples, which are almost equilateral or almost isosceles in shape have more matches across fingerprints. It is less likely that any 9-point feature formed by such triples is a rare feature. By removing 9-point features formed by such triples from the search space, time taken for comparisons can be greatly reduced.

Handling Data Complexity: In this work, only a small set of minutiae and only a maximum of 1 Million 9-point features from the reference fingerprint are considered. A task for future implementations would be to consider all the possible 9-point features from all the minutiae to mine rare features. There is a need to find methods to handle data complexity and longer run-times involved in this kind of approaches.

Bibliography

[1] A. Newman, Fingerprinting's Reliability Draws Growing Court Challenges, N.Y. Times, April 7, 2001.

[2] D. R. Ashbaugh, Quantitative-qualitative friction ridge analysis: An introduction to basic and advanced ridgeology, Boca Raton, FL: CRC Press, 1999.

[3] M. R. Hawthorne, Fingerprints: Analysis and Understanding, CRC Press, 2008.

[4] S. Pankanti, S. Prabhakar and A. K. Jain, On the Individuality of Fingerprints, vol. 24, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, pp. 1010-1025.

[5] National Institute of Justice, Forensic Sciences: Review of Status and Needs, U.S Department of Justice, 1999.

[6] National Institute of Justice, Forensic Friction Ridge (Fingerprint) Examination Validation Studies, U.S. Department of Justice, Solicitation, 2000.

[7] National Research Council of the National Academies, Strengthening Forensic Science in the United States: A Path Forward, Washington, DC: The National Academies Press, 2009.

[8] K. E. Hoyle, N. J. Short, M. S. Hsiao, A. L. Abbott and E. A. Fox, "Minutiae+ friction ridges= triplet-based features for determining sufficiency in fingerprints," London, Proceedings: 4th International Conference on Imaging and Crime Detection (IDCP-11), Nov. 2011.

[9] K. Hoyle, Minutiae Triplet-Based Features with Extended Ridge Information for Determining Sufficiency in Fingerprints, Blacksburg, VA, USA: MS Thesis, Virginia Polytechnic Institute and State University, 2011.

[10] M. A. Medina-Pérez, M. García-Borroto, A. E. Gutierrez-Rodriguez and L. Altamirano-

Robles, Improving Fingerprint Verification Using Minutiae Triplets, vol. 12, Sensors, 2012.

[11] X. Tan and B. Bhanu, Fingerprint matching by genetic algorithms, vol. 39, Pattern Recogn, 2006, pp. 465-477.

[12] G. Parziale and A. Niel, A Fingerprint Matching Using Minutiae Triangulation, Springer ed., vol. 3072, Berlin/Heidelberg: International Conference on Biometric Authentication (ICBA), 2004, pp. 1-50.

[13] D. Maltoni, D. Maio, A. K. Jain and S. Prabhakar, Handbook of Fingerprint Recognition (2nd Edition), Springer, 2009.

[14] H. C. Lee and R. E. Gaensslen, Advances in Fingerprint Technology, New York: Elsevier, 2001.

[15] C. Champod, C. J. Lennard, P. Margot and M. Stoilovic, Fingerprint Detection Techniques, CRC Press , 2004.

[16] C. Wilson, G. Candela and C. Watson, Neural network fingerprint classification, vol. 1, Journal of Artificial Neural Networks, 1994, pp. 203-228.

[17] S. Chikkerur and N. Ratha, Impact of Singular Point Detection on Fingerprint matching performance, Proc. Fourth IEEE Workshop Automatic Identification Advanced Technologies, 2005, pp. 207-212.

[18] X. Jiang and W. Y. Yau, Fingerprint minutiae matching based on the local and global structures, vol. 2, Proceedings. 15th International Conference on Pattern Recognition, 2000, pp. 1038-1041.

[19] N. Ratha, R. Bolle, V. Pandit and V. Vaish, Robust Fingerprint Authentication Using Local Structural Similarity, Proc. Fifth IEEE Workshop Applications of Computer Vision, 2000.

[20] R. Germain, A. Califano and S. Colville, Fingerprint matching using transformation parameter clustering, vol. 4, IEEE Computational Science and Engineering, 1997, pp. 42-

49.

[21] B. Bhanu and X. Tan, A Triplet Based Approach for Indexing of Fingerprint Database for Identification, Springer ed., vol. 2091, Berlin/Heidelberg: Audio- and Video-Based Biometric Person Authentication, 2001, pp. 205-210.

[22] Z. M. Kovacs-Vajna, A fingerprint verification system based on triangular matching and dynamic Time Warping, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, pp. 1266-1276.

[23] T. Jea and V. Govindaraju, A minutia-based partial fingerprint recognition system, vol. 38, Pattern Recognition, 2005, pp. 1672-1684.

[24] T. Jea, Minutiae-Based Partial Fingerprint Recognition, NY, USA: MS Thesis, State University of New york, 2005.

[25] M. Ghazvini, H. Sufikarimi and K. Mohammadi, Fingerprint Matching Using Genetic Algorithm and Triangle Descriptors, Tehran, Iran: Proceedings of the 19th Iranian Conference on Electrical Engineering, 2011, pp. 1-6.

[26] X. Chen, J. Tian, X. Yang and Y. Zhang, An algorithm for distorted fingerprint matching based on Local Triangle Feature Set, vol. 1, IEEE Transactions on Information Forensics and Security, 2006, pp. 169-177.

[27] J. Zheng, Y. Gao and M. Zhang, Fingerprint Matching Algorithm Based on Similar Vector Triangle, Tianjin, China: Proceedings of the 2nd International Congress on Image and Signal Processing (CISP '09), 2009, pp. 1-6.

[28] Y. Feng, J. Feng, X. Chen and Z. Song, A Novel Fingerprint Matching Scheme Based on Local Structure Compatibility, vol. 4, Hong Kong, China: Proceedings of the 18th International Conference on Pattern Recognition, 2006, pp. 374-377.

[29] W. Xu, X. Chen and J. Feng, A Robust Fingerprint Matching Approach: Growing and Fusing of Local Structures, vol. LNCS 4642, Seoul, Korea: Proceedings of the 2nd

International Conference on Biometrics, 2007, pp. 134-143.

[30] N. Vandal and M. Savvides, CUDA accelerated iris template matching on Graphics Processing Units (GPUs), IEEE Conference on Biometrics, Theory, Applications and Systems (BTAS), 2010.

[31] A. Abate, M. Nappi, S. Ricciardi and G. Sabatino, GPU accelerated 3D face registration / recognition, Springer ed., vol. 4642, Heidelberg: International Conference on Biometrics (ICB), 2007, pp. 938-947.

[32] P. Gutierrez, M. Lastra, F. Herrera and J. Benitez, A high performance fingerprint matching system for large databases based on GPU, IEEE Transactions on Information Forensics and Security, 2013.

[33] NVIDIA, "GPU Computing," [Online]. Available: www.nvidia.com.

[34] NVIDIA, Popular GPU-Accelerated Applications.

[35] NVIDIA, "GeForce GTX 200 GPU Technical Brief," 2008.

[36] NVIDIA, CUDA Programming Guide.

[37] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman and A. K. Jain, "FVC2000: Fingerprint Verification Competition," vol. 24, IEEE Transactions on Pattern Analysis and Machine Intelligence, March 2002.

[38] Neurotechnology, "Verifinger SDK," 2012. [Online]. Available: http://www.neurotechnology.com.

[39] L. Shen, A. Kot and W. Koo, Quality Measures of Fingerprint Images, Springer ed., vol. 2091, J. Bigun and F. Smeraldi, Eds., Berlin / Heidelberg: Audio- and Video-Based Biometric Person Authentication, pp. 266-271.

[40] Y. He, J. Tian, Q. Ren and X. Yang, Maximum-Likelihood Deformation Analysis of Different-Sized Fingerprints, Springer ed., vol. 2688, J. Kittler and M. Nixon, Eds., Berlin /

Heidelberg: Audio- and Video-Based Biometric Person Authentication, 2003, pp. 1062-1062.