Fixed Length Fingerprint Representation

A Registration Seminar Report
submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Alka Ranjan
[21CS91R01]

under the supervision of

Prof. Debasis Samanta

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Kharagpur-721302, India
February 26, 2023

Contents

1 Introduction

1.1 Background

Biometrics, also known as biometric recognition, is a process of automated recognition of any individual on the basis of some of their behavioral and/or physiological trait. [**?**] We, as humans, have used a person's face, gait, or voice to distinguish them from other people around us or recognize them for thousands of years. In the late 19th century, a need for an automated recognition system using biometrics, particularly fingerprints, arose for identifying criminals and storing them in a database. Some examples of biometrics are DNA, fingerprint, face, gait, voice, hand or ear geometry, hand or finger vein infrared thermograph, iris, retina, keystroke, odor, palmprint, and signature/handwriting and can be seen in figure 1 [**?**].

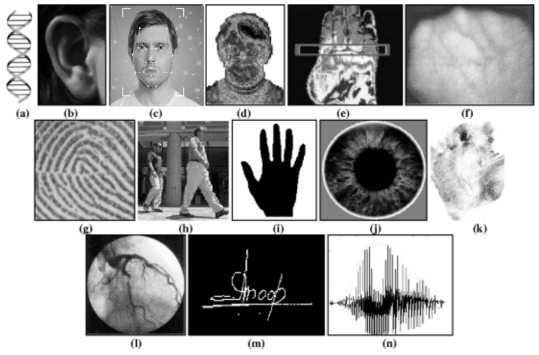


Figure 1: Examples of biometric characteristics: (a) DNA profiling, (b) ear shape, (c) facial features, (d) facial thermograph, (e) hand thermograph, (f) hand vein profile, (g) fingerprints, (h) gait, (i) hand geometry, (j) iris, (k) palmprint, (l) retina scans, (m) signature, and (n) voice.

1.2 Biometric Authentication System

A biometric authentication system is a type of pattern recognition system. It can recognize/authenticate a person by determining whether some behavioral characteristic and/or a specific physiological possessed by that person is authentic.[**?**]

Essentially, a biometric system uses acquired biometric data of an individual, extracts numerical features from it, and compares the feature set to that of another person/ or a template set in a database. Broadly we can classify biometric systems based on one of two operation modes.

- Verification: one-to-one matching, checks if a given biometric matches an existing enrollment. (returns True/False)
- Identification: one-to-many matching, recognizes/retries information using an input biometric over a set of existing enrollments. (returns the record or similar linked information)

Figure 2 shows that the first step in any biometric system is the acquisition of biometric data from the users of that system. The acquired data is stored is transmitted in a compressed manner and stored in a secure location. Various techniques can be used for processing the data. Usually, most systems process the incoming data to store them accordingly. This work will be focused on the data storage and signal processing part.

An interesting corner case for a biometric system is the distinction between identical twins. Several works show that accurate verification of an individual from their twin can be done using their fingerprint [**?**], palmprint [**?**], face [**?**], as well as multimodal traits [**?**]. An intriguing note is that each iris of the same person is unrelated to each other [**?**] just like fingerprints from different fingers of the same person, thus vary between twins as well.

1.3 Indexing

The definition of data indexing in simple terms is to re-organize/process data items in a database such that we have a key to data in the database that helps us to find the location of the item in the memory. In

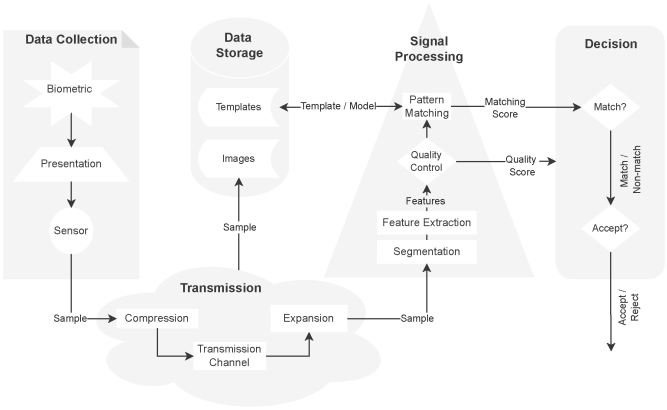


Figure 2: Block Diagram of a generic biometric system along with its subsystems [**?**]

indexing search query is taken as input and the collection of matching records is returned efficiently. This index (key) can be calculated by several techniques such that the searching process is as time-efficient and space-efficient as possible. However, there is usually an inverse relationship between time and space complexities when using an index.

Mainly Indexes can be classified into the following types [**?**] and is shown in 3:

- *Dense Index:* All the data is sorted based on a unique attribute, also known as the primary key. There exists an index record (that has the key and holds the pointer to the data with the key) for every search key in the data file. This can help to search faster but as a drawback it needs significantly more space in order to store the index records.

- *Sparse Index:* Only some index records pointing to the start of blocks are stored, which are like containers of data. This is to have better space complexity than dense indexing but the search time is slower. Reference to all the keys does not exist. Pointers point only to the first key of the block. Thus, it cannot be determined directly from the index if a key is present until the pointer is followed and the block is checked thoroughly.

- *Secondary Indexing:* This is a kind of a two-level indexing technique and is used to reduce the mapping size of the first level. Between index records and actual data blocks, there exist buckets in between. There are indexes on these buckets and these buckets contain pointers to the actual data. It is kind of a mix of sparse(index on buckets) and dense(buckets to data) indexing to be more efficient.

- *Hashing:* A Hash Function maps a key to a bucket in which actual data is stored. Simply put, a query key is passed into the hash function and the output is the pointer to the container where the data is present.

- *B-Tree(B+ Tree):* It is a data structure for tree-based indexing. It is constructed in such a way that all leaf nodes contain actual data pointers with key values in them. All leaf nodes are connected like a linked list to support random and sequential access. The non-leaf nodes contain keys and pointers to navigate to the required value. If a non-leaf node has n search keys then, it will have n+1 pointers (a pointer to the left and right of every key). If the query key is less than the search key then the left pointer is used to navigate further otherwise the right pointer is used.

- *kd-Tree:* It is a generalization of binary search trees to multidimensional data. In other words, a kd-tree is a space partitioning data structure in a k-dimensional space. Similar to a binary tree, a non-leaf node in the kd-tree divides the space into 2 parts. Points to the left of this space are represented by the left subtree of that node and points to the right of the space are represented by the right subtree. Each level in the tree goes on dividing the plane into separate planes and each non-leaf node contains the plane/attribute on which the space is divided. Leaf nodes are the blocks with records in the space.

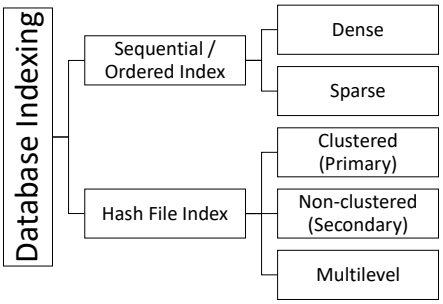


Figure 3: Classification of Database Indexing [**?**]

1.3.1 Indexing of Biometric data

Almost all applications of biometric systems involve dealing with large amounts of complex data which can be in the order of millions. Also, biometric data never follows any natural sorting order, because of which traditional/database indexing mechanisms do not work well for biometric systems. One of the major limitations of the existing biometric systems is query matching for large databases which often causes inefficient data retrieval.

Traditional indexing techniques like k-d tree-based indexing, geometric hashing, k-means clustering, etc. index the records in alphabetical or numeric order. These traditional algorithms have been used with multi-dimensional data, but, they fail with multidimensional biometric data as it is non-correlated. More importantly, biometric data has a huge variety of features, thus, it is important to consider unique identifiers which are more resilient to distortion and provide better accuracy even with the presence of noise.

This work proposes a novel indexing scheme for retrieving biometric data using features extracted from a single biometric trait as well as from multiple biometric traits. Going by the block diagram of a generic biometric system given in Figure 2, this work focuses on *Data Storage* and *Signal Processing* subsystems. At first, uni-modal biometric features are considered for indexing. Afterward, a multi-modal indexing approach has been proposed by taking into consideration more than one biometric trait. Several challenges posed by the biometric traits, such as distortion due to scaling and rotation, need to be resolved in order to achieve the best results from the index.

2 Literature Survey

In order to find if an entry exists, any biometric identification system needs to search through a huge volume of data entries for possible matches. It uses a unique identifier called a feature, to create a template that is used for the match. The overall performance of the biometric identification system depends on several factors, such as the efficiency of the data capturing, processing, storage, and retrieval steps. However, sequential matching of the input with all the enrolled entries is not possible for any real-time application. Thus, we need a system in place that can help us filter and reduce the search space.

2.1 Fingerprint-based biometric data indexing

The simplest/naive way to index fingerprint biometric data is to store biometric data independent of each other and then perform a linear search on the biometric data as they do not possess a natural sorting order [**?**]. Matching an input fingerprint in a database of N enrollments requires N separate matchings, which is viable only for small databases. However, for large databases, some notable approaches include continuous classification, clustering, hashing and indexing [**? ? ? ? ?**].

Germain et al. [**?**] extracted minutia points in the fingerprints to create all possible combinations of triangles. The features used for index key generation were: triangle length, count of ridges inside the triangle, and individual angles of the triangles. Bhanu et al. [**?**] too proposed a method using minutia triplets. The features used were its maximum side, handedness, individual angles, direction, and type

for each triangle. However, both approaches were too expensive computationally as $O(n^3)$ triplets were produced.

Classification in biometric indexing is a supervised learning approach to group the fingerprint images into different classes based on training done on a set of labeled fingerprints. A commonly used classification model was proposed by Henry et al. [**?**], which is utilized by Park et al. [**?**] in their approach for classifying images using FFT-based features.

There are a large number of identifying features in any fingerprint image, thus the indexing techniques must support multi-dimensional data. A data structure that supports multi-dimensional data, which is popularly used, is kd-tree [**?**]. Vandana et al. [**?**] used kd-tree for storing nine-dimensional features extracted from fingerprints. Their approach showed better results in comparison to the linear storage of the data, by reducing the overall search time. Their drawback was that the efficiency significantly decreased with the increase in the dimensions of data [**?**]. Another tree structure popularly used for indexing was R-tree and some variations of R-tree used by Kriege et al. [**?**]. Due to unbalance in the tree structure, their worst-case scenario for searching was close to linear searching. To improve on this, various clustering methods were proposed with the promise of making the storage system more efficient, amongst which k-means clustering gained the most popularity [**?**], [**?**], [**?**]. Jiang et al. [**?**] put forth a filtering algorithm for fingerprint identification. Their approach used "dominant ridge distance" and "orientation fields" as the retrieval features. The extracted features were clustered to help the retrieval process and removed to need to exhaustively compare an input fingerprint with the entire database. Liu and Yap [**?**] employed a "complex polar moments (PCMs)" for fingerprint ridge flow orientation and structure extraction. This was done for singular regions as well. Using the orientation fields, a feature vector in the form of "rotational moment invariants" was used to get a structural description of the complete fingerprint. Finally, an indexing scheme based on clustering was implemented, for efficient retrieval of likely candidates, in the fingerprint database. MV-Indexing approach proposed in [**?**] considers features obtained from the minutia points, such as direction, location, and information from ridge endings. An indexing method was based on score-level fusion which combined the features extracted with the "minutia cylinder code (MCC)". Finally, k-means clustering was employed to index the data.

In the clustering-based approaches, the accuracy and speed of the system depend highly on the number of clusters. To minimize such dependency and to improve retrieval accuracy, Ferreira et al. [**?**] proposed a fuzzy c-means clustering method. However, because of the high time complexity involved, the approach is not much explored.

To overcome the aforementioned limitations of the methods, LSH-based indexing methods were proposed [**?** **?**]. These methods bypass the curse of dimensionality while eliminating the need to decide the count of the storage locations. Despite these advantages, with the increase in the total size of the database, the data retrieval time of the LSH-based indexing increases significantly.

A modified geometric-based hashing technique was proposed by Jayaraman et al. [**?**]. The strength of this approach was that the same set of features was only stored once in the database and was never repeated. In order to improve the speed of data access, the approach stored only features from the core point of each fingerprint. The limitation was that such a system has a very high dependence on the core point and its identification.

2.2 Iris-based biometric data indexing

The indexing method proposed by Mukherjee et al. [**?**] used k-means clustering. A similar clustering approach was given by Puhan et al. [**?**]. They proposed a method to index the iris images on the basis of the iris color. Their proposed method used the chrominance component to partition the data into various groups, which in turn was used as a key for extracting the possible sets of images. To use texture-based features of iris images, a texture-based representation method called "Hierarchical Visual Codebook (HVC)" was given by Sun et al. [**?**]. For the classification of iris images, their approach integrated the concept of "Vocabulary Tree (VT)" and "Locality-constrained Linear Coding (LLC)".

Classification and clustering-based approaches require prior information of the classes or clusters for accurate data partition. If that information is not known then the clusters/classes may not divide the dataset efficiently which may perhaps lead to the worst-case scenario of linear search. To overcome this issue, Mehrotra et al. [**?**] used "Scale Invariant Feature Transform (SIFT)" features to index the data while also applying a geometric hashing-based scheme. The work was further extended in [**?**] which indexed the data using kd-trees. Jayraman et al. [**?**] proposed to index SURF features using kd-tree as indexing method. Similarly, Barbu et al. [**?**] used kd-tree to index content-based features.

Rathgeb et al. [**?**] proposed a unique hash-based approach. The search space was reduced to 3% with a hit rate of 90% by using 4-bit biometric keys. This key was generated from iris images. Requiring high storage was the drawback of the approach. Dey et al. [**?**] proposed "Gabor Energy", to make the identification process more efficient. For each iris template, a 12-dimensional index key was generated, which had a hit rate of 98.2% and the search space was limited to 12%. Hugo et al. [**?**] proposed an iris indexing scheme that works with unconstrained situations. They used "UBIRISv2 database" [**?**] for comparative evaluation.

A few other noteworthy works in iris-based biometric data indexing include the work by Gadde et al. [**?**]. They proposed an approach based on "Burrows-Wheeler transformation"[**?**], to achieve 99.8% hit rate with penetration rate of 12.3%, tested on a database of 249 subjects. Rathgeb et al. [**?**] used bloom filters to index iris images with a 93.5% hit rate and penetration rate of 6.2%. The drawback of the approach was the use of all samples at every level of the tree and any deletion operation required a full tree replacement (with $O(Nlog(N))$ complexity). Drozdowski et al. [**?**] extended the work of Rathgeb et al. [**?**] by using a binary search tree in conjugation with bloom filters. Damer et al. [**?**] have used multi-instance indexing for iris data, with a hit rate of 99.98% at a penetration rate of 0.1%. Khalaf et al. [**? ?**] proposed a "Scalable K-means++ algorithm" for the classification and partitioning of iris data. They also applied a parallel approach to divide the features groups to form an index key based on two b-trees for retrieval and searching.

2.3 Multi-modal biometric data indexing

It has been observed that some of the limitations of uni-modal biometric systems such as intra-class variations, low identification rate, and limited security [**?**] can be addressed by deploying multi-modal biometric systems. Such systems are based on the principle of combining information from two or more biometric modalities to ensure higher accuracy by eliminating any chance of spoofing.

Various commercial, civilian and forensic applications use multi-modal biometric systems to establish identity [**?**]. The deployment of multi-modal biometric systems on national scale results in a growing database size [**?**]. These databases are accessed extensively, and thereby require efficient ways of searching and retrieving relevant identities. Considering the large volume of data, the approach to limit the search to a small region is required. In literature, this is accomplished by using classification or indexing approaches. However, classification approaches may divide the database into uneven class sizes which further causes search limitations during query matching. For instance, Prabhu et al. [**?**] classify the sample as male or female to enhance the search efficiency for both identification and verification process.

Soheil et al. [**?**] proposed to classify images by multi-modal task-driven dictionary learning. The authors have shown the application of the proposed work in different applications: "multi-modal face recognition, multi-view face recognition, multi-view action recognition, and multi-modal biometric recognition". Goswami et al. [**?**] proposed an approach to classify fused features of multi-modal biometrics. In this work, they removed the requirement for a separate feature-level fusion mechanism and integrated representation using multi-feature for classification. However, their approach can not reduce the search to a minimal set of elements pertaining to the limited classes it has.

For fast data retrieval and searching various hashing approaches are used. In order to address the need for a fast similarity search, a compact indexing structure is proposed by Wang et al. [**?**]. One major advantage of using a hash-based index is that the lookup on a hash table takes constant time [**?**]. Its use in biometric indexing is only logical. For efficient and effective large-scale image retrieval on a large scale, many hashing algorithms have been introduced [**?**]. The use of hashing maps the data points of high dimension into a low-dimension binary code which makes searching for nearest neighbors in a hamming space very efficient.[**?**] The drawback of hashing methods is that the same hash is not generated for new instances of the same biometric.

3 Scope of Work and Objectives

3.1 Scope of Work

Biometric data indexing is a technique where an index key is generated for biometric data and is assigned to the corresponding template. Based on these index keys the biometric database can be categorized into groups of similar biometric templates. This allows reducing the 1:N matching for identification purposes to a smaller set of candidates for the query template. For a query image, the system will retrieve a smaller

set of candidate database templates to be matched with the query template, which can effectively reduce the search and identification time. An overview of the indexing system is shown in Figure 4.

Biometric indexing can facilitate several biomteric-based systems such above mentioned FBI database of fingerprints, UIDAI's AADHAAR-based national identity system, as these systems are very large, and without a proper indexing mechanism in place, the identification process can be very inefficient. The Government of India has generated a unique identity for each citizen through the biggest ever project called AADHAAR under the aegis of Unique Identification Authority of India (UIDAI). The Central Identity Data Repository (CIDR) in AADHAAR stores about 10+ Petabytes of data consisting of more than 1.33 billion people and their 13.3 billion fingerprints scans, 2.66 billion iris scans, and 1.33 billion facial photos [**?**]. For such a massive database, traditional indexing schemes fail, and thus there is a need for proper indexing mechanism for identification in large databases such as AADHAAR system.

On surveying the existing literature, it is observed that there is a scope to further improve the performance of existing methods for key generation. The robustness of the keys generated will affect the matching time taken when it uses that key as the index. In order to achieve this, a sophisticated key generation method is needed.

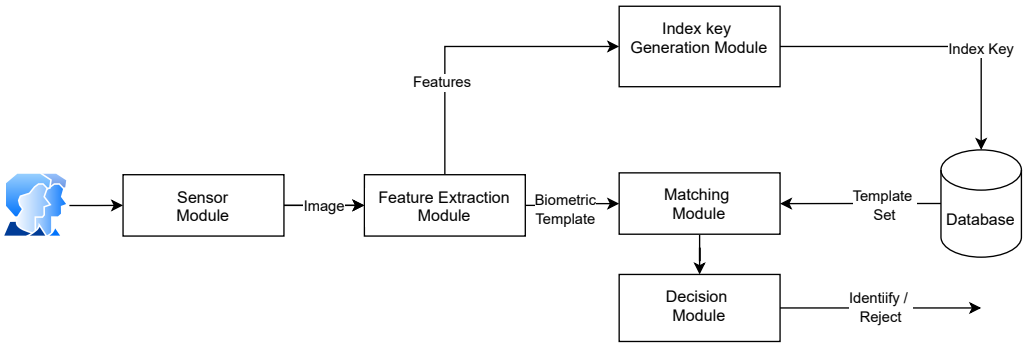


Figure 4: Steps in Identification with Indexing

3.2 Motivation

Identification suffers from two major limitations (a) false matching and (b) long response time due to a large candidate set. Traditional indexing schemes are not usable for indexing biometric databases as there is no numeric or alphabetical order in the biometric data. In most of the previously proposed methods, there are certain shortcomings such as [**?**] uses an extended triangulation-based method for indexing but the index vectors are highly correlated which in turn causes the penetration rate to increase. In [**?**] $O(N^3)$ triangles are used for generating index vectors which is computationally expensive. In [**?**] they used texture-based global features for indexing, global texture features are not very robust against noise, core, and delta points may also not be present in a fingerprint impression. These shortcomings motivated this work to produce an effective indexing method which is robust against noise, and also uses local features which are less expensive to compute.

This work is mostly focused on creating an efficient indexing scheme for fingerprint identification such that the penetration rate and false acceptance rate are reduced and at the same time the identification accuracy is improved. This work uses image enhancement to ensure stable minutiae point extraction. In this work, an effective way to compensate for the shortcomings of preexisting methods for different modules is explored to find a better solution for the problem.

Considering only one trait for identification can prove fatal in many cases. Thus, there arises a necessity for multi-modal biometric systems which can resolve many identification issues. For such systems, similar multi-modal approach is needed in terms of indexing as well in order to ensure quick and efficient identification.

3.3 Objectives

The primary objective of this work is to propose an indexing scheme for different biometric traits which allows for fast retrieval and reduces the search space as well as search time. This work considers generating the index on individual traits as well as a combination of multiple biometric traits in order to improve performance, thereby ensuring that the penetration rate of the indexing scheme is fairly low with reduced

false matching to allow for secure and efficient identification. Along with the above, this work also aims to create an open-source dataset of various synthetic biometric trains for testing and comparing performances of different indexing mechanisms. The objectives of the work are as follows:

1. To create a large-scale virtual biometric dataset featuring low-quality and high-quality samples for multiple biometric traits (eg. Fingerprints, Iris, Face) as the first thesis objective. The performance of traditional indexing schemes on the aforementioned synthetic data is to be evaluated and compared.
2. To design and implement an effective and scalable indexing scheme with distortion invariant features which enable large-scale biometric identification systems to reduce processing time in user identification in a uni-modal system as the second thesis objective.
3. To design and implement an effective and scalable indexing scheme with distortion invariant features which enable large-scale biometric identification systems to reduce processing time in user identification in a multi-modal system as the final thesis objective.

4 Work Done

4.1 Overview of the Work Done

The overview of the work done is as follows:

- A stable consistent region selection procedure is proposed using GLCM and statistical features to extract features invariant to sensor errors.
- Combination of different types of feature vectors gives an edge over the existing methods to obtain an irreversible and stable key from the fingerprint of a user.
- The proposed method works well over images of varying quality and resolution. The approach is robust, produces accurate results and is tested thoroughly with several fingerprint biometric datasets ([**?**], [**?**], [**?**]).
- The keys generated according to the proposed approach passed statistical tests like NIST tests [**?**] and Diehard tests [**?**] successfully, thus ensuring randomness.
- The proposed method is a simple technique for dynamic and scalable key generation which does not require complex hardware and is computationally inexpensive.

4.2 Proposed Approach

This section enlists the proposed approach of key generation using a fingerprint image. An overview of the proposed approach is shown in Fig. 5. The approach is divided into several modules. The objectives and details of each module are as follows:

- *Pre-processing:* A biometric data captured in an uncontrolled the environment is usually riddled with noise and has an improper alignment. Standard pre-processing procedures for image enhancement and alignment is carried out to deal with such input.
- *Consistent region selection:* In addition to noise, another problem with the input is the partial capturing of the fingerprint. A part of the input image is chosen meticulously to deal with this problem so that the region is consistent in most of the input data.
- *Hybrid feature extraction:* In this step, a feature set is extracted from the consistent region.
- *Prominent feature selection:* Extracted features are optimized to generate a stable feature vector.
- *Discriminant feature vector generation:* In order to make the key-generation process robust and reliable, the discriminant feature vector is extracted from the prominent features.
- *Key generation:* Finally, the key is generated from the discriminant feature vector.

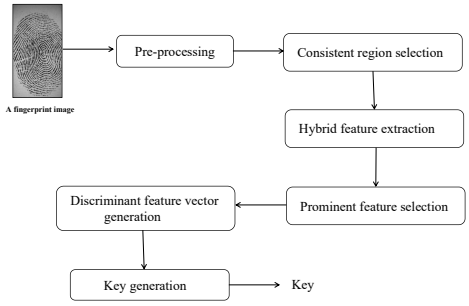
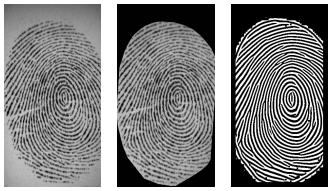


Figure 5: Overview of the key generation from a fingerprint image.

4.2.1 Pre-processing

The pre-processing step is crucial for stable feature extraction from a fingerprint image. Level-2 features such as the minutiae feature extraction depend heavily on accurate and standard pre-processing steps. The pre-processing steps consist of robust and accurate segmentation, alignment, and enhancement procedures.

This work adopts factorized directional bandpass (FDB) segmentation [**?**] technique to extract the fingerprint region from a fingerprint image. This segmentation technique helps in vertically aligning the rotated fingerprint effectively. Next, ASRA alignment technique [**?**] has been implemented to align the input fingerprint vertically. Finally, the fingerprint image is enhanced using the method proposed by Ding et al. [**?**]. A sample pre-processed fingerprint is shown in Fig. 6 as an illustration.



(a) Original fingerprint image. (b) Vertically aligned image. (c) Enhanced image.

Figure 6: An illustration of pre-processing of a sample fingerprint image.

4.2.2 Consistent region selection

The entire fingerprint not necessarily contribute evenly for robust feature extraction. The consistent region is selected from the fingerprint to obtain efficient features that can increase recognition accuracy. In this work, both Gray Level Co-occurrence Matrix (GLCM) [**?**] and statistical features-based approaches are followed to determine the consistent region of an image. The procedure consists of the following concepts.

4.2.2.1 Gray Level Co-occurrence Matrix (GLCM) and statistical features

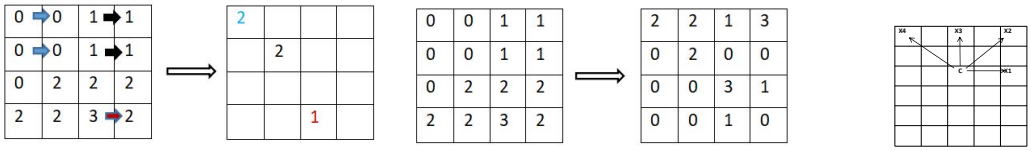
GLCM is a popular technique known for the extraction of texture-based features in an image [**?**]. This method estimates the relative position of pixels in a neighborhood by calculating the statistical properties of the image.

In this method, a co-occurrence of a matrix C is calculated as

$$C_{\Delta m,\Delta n}(x,y) = \sum_{i=1}^{M-\Delta m} \sum_{j=1}^{N-\Delta n} \begin{cases} 1 & if\ I(i,j) = x, I(i+\Delta m, j+\Delta n) = y \\ 0 & Otherwise \end{cases} \quad (1)$$

where $I(i,j)$ is the intensity of an image at position (i,j). Image size is $M \times N$. The vector v=$(\Delta m, \Delta n)$ covers 4 directions $(D1, D2, D3, D4)$ in an image. The working of GLCM process is shown in Fig. 7.

From the GLCM matrix, three statistical features, namely Correlation, Homogeneity and Energy are obtained using Equations 2, 3 and 4, respectively.



(a) GLCM calculation example for value pairs (0,1), (1,1) and (3,2) in the offset direction 0^o.

(b) GLCM co-occurrence matrix.

(c) GLCM matrix calculation along four directions with offset two.

Figure 7: An illustration of GLCM feature extraction.

Correlation:
Returns a measure of how correlated a pixel is to its neighbor compared to the whole image.

$$C\_Correlation = \sum_x \sum_y \frac{(x - \mu_x) \cdot (y - \mu_y) \cdot C(x, y)}{\sigma_x \cdot \sigma_y} \tag{2}$$

Homogeneity:
Returns a value that measures the closeness of the distribution of elements in the GLCM(C) to the GLCM(C) diagonal.

$$C\_Homogeneity = \sum_x \sum_y \frac{C(x, y)}{1 + |x - y|} \tag{3}$$

Energy:
Returns the sum of squared elements in the GLCM(C).

$$C\_Energy = \sum_x \sum_y C(x, y)^2 \tag{4}$$

4.2.2.2 Consistent region selection based on GLCM statistical feature descriptor

The proposed consistent region selection method uses GLCM statistical feature descriptor followed by a classification and k-fold validation step. The steps are as follows:

- **Step-1**: An input fingerprint image I is duplicated into four instances.
- **Step-2**: For each fingerprint instance, a central point o is determined. Let the core point in I be the central point o. Next, 64×64 region around the central point o is selected. Let the region be denoted as r.
- **Step-3**: For the region r in each fingerprint instance, the GLCM matrix C is calculated using Equation 1. The feature descriptor is derived using Equations 2, 3 and 4, respectively. Let the feature vector be called f, where $f = [f_{Correlation}, f_{homogenity}, f_{energy}]$ of length 12. The features are calculated along four directions $(0^0, 45^0, 90^0, 135^0)$ and with different pixel offsets $(1, 2, \ldots, 10)$. If there are 10 individuals in the dataset, each having 8 instances, then 4 instances of each user is used in this step. The feature vectors derived for all four instances for all individuals is combined into one vector (denoted as F). The resultant vector F has dimension 40×120.
- **Step-4**: One-vs-all SVM is applied on F with k-fold cross validation (k=10). The prediction accuracy, A_t, is calculated by averaging the accuracy obtained from all k folds.
- **Step-5**:The region r (from Step 1) of each instance in the fingerprint is scaled up 1.2 times (decided empirically). Steps 2, 3 and 4 are repeated until $A_{t+1} \leq A_t$ (where, t is the iteration number)
- **Step-6**:The resultant r is the required consistent region.

For a query image, this region r is used as a reference around central point to select the consistent region. The consistent region selection is shown in Fig. 8.

(a) Original ROI r around central point. (b) Scaled ROI of r. (c) Consistent region.

Figure 8: Consistent region selection.

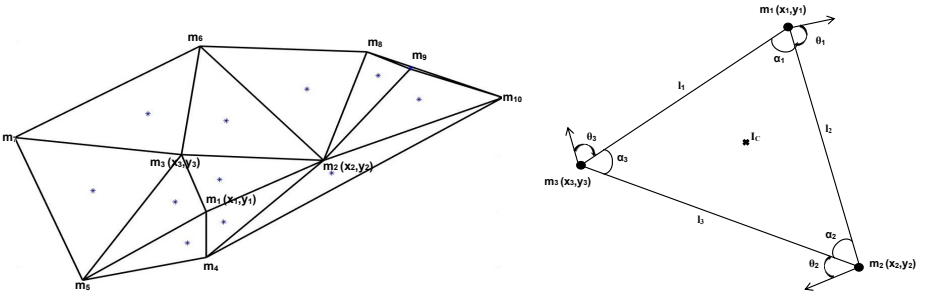
4.2.3 Feature extraction

In most of the work reported in the literature, minutiae point-based features are used in solving the verification and authentication problems [**?**]. Such a feature is called a direct (also called global) feature. However, the application of minutia features has the following two shortcomings: (i) the minutiae feature extraction accuracy and (ii) the size of the feature set, which depends on the number of minutiae points in an input image. For example, in wet or crushed fingerprints, the number of extracted minutiae is significantly lower, whereas, in dry fingerprints, false minutia points are often observed due to the cuts in the ridges. For better accuracy, it is advocated to follow indirect features and other features. Further, a combination of multiple features of different nature can be a better alternative. This work proposes a method of generating a hybrid feature vector combining the texture features and indirect features. These features are extracted from the consistent region of the fingerprint. The different feature extraction procedures are described in the following sub-sections.

4.2.3.1 Indirect feature vector extraction

An indirect feature vector derived from a set of minutiae points is considered for this work. Consider a fingerprint f; it consists of minutiae features, which are points on the fingerprint, specifically ridge endings and ridge bifurcations [**?**]. A single minutia point has coordinates (x, y), and angle of orientation denoted by θ [**?**]. Delaunay triangulation [**?**] net is formed over the minutiae points in the extracted region of interest. Delaunay triangulation [**?**] over a given set of discrete points is the triangulation such that no point lies on the circumference of any triangle. It maximizes the minimum angle among all angles for all the triangles belonging to the triangulation. The triangulation gives a stable structure that can be used to determine indirect features for a given set of points.

First, the center of the region of interest (ROI) is approximated and its distance is computed from the in-centers of all the triangles in the net. Nearest neighboring triangles from the ROI's center are considered for further processing. The indirect feature set for a particular fingerprint instance obtained from the triangulation contains the concatenated information of the relative area of each triangle, relative length (considering sides of individual triangle), relative angle, relative in-center with respect to individual triangle, relative position and orientation as shown in Fig. 9. Here, relative refers to normalized value with respect to the maximum value in that particular set, for example $relative\ area\ =\ area\ of\ a\ \triangle\ /\ max\ area\ of\ a\ \triangle\ in\ triangulation$. The algorithm for relative indirect feature vector generation is described as Algorithm 1.



(a) Delaunay triangular net from a set of minutia points. (b) Indirect feature extraction from a triangle.

Figure 9: An example of Delaunay triangulation-based indirect feature generation.

Algorithm 1: Relative indirect feature vector generation

Input : f is a fingerprint.
Output: feature vector f_v of f

(1) **Extract** the center of the ROI (c) (x_c, y_c)
(2) **Extract** the minutiae features from the ROI. Let i^{th} minutia in the ROI (having center (x_i, y_i)) be (m_i)
(3) **Form** TR = delaunayTriangulation (on the minutiae points).
Let no. of triangles in TR = t
Let in-center of j^{th} triangle in TR be (x_j, y_j)
(4) **for** *i=j to t* **do**
(5) $\quad d\,(c, t_i) = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2}$
(6) **Select** k nearest triangles from the center of ROI (c) (x_c, y_c)
Let the term **relative** be denoted by **r**
(7) **for** *i=1 to k* **do**
(8) $\quad$ r_area($\triangle z_i$) = area($\triangle z_i$) / max(area($\forall \triangle \in$ TR))
(9) $\quad$ r_length_of_sides($\triangle z_i$) = length_of_sides($\triangle z_i$) / max(length_of_sides($\forall \triangle \in$ TR))
(10) $\quad$ r_angles_b/w_sides($\triangle z_i$) = angles_b/w_sides($\triangle z_i$) / max(angles_b/w_sides($\forall \triangle \in$ TR))
(11) $\quad$ r_incenter($\triangle z_i$) = incenter($\triangle z_i$) / max(incenter($\forall \triangle \in$ TR))
(12) **for** *i=1 to 10* **do**
`/* Ten consistent minutia selected in the consistent region. */`
(13) $\quad$ r_position_value(m_i) = position(m_i) / max(position_value($\forall\ m_i\ \in\ m$))
(14) $\quad$ r_orientation(m_i) = orientation(m_i) / max(orientation($\forall\ m_i\ \in\ m$))
(15) **feature vector(f_v)**= [r_area($\forall \triangle \in$ TR) $\|$ r_length_of_sides($\forall \triangle i \in$ TR) $\|$ r_angles_b/w_sides($\forall \triangle \in$ TR) $\|$ r_incenter($\forall \triangle \in$ TR) $\|$ r_position($\forall\ m_i\ \in\ m$) $\|$ r_orientation($\forall\ m_i\ \in\ m$)]

4.2.3.2 Texture-based feature extraction

"Texture" is a characteristic item in any fingerprint image. Local texture descriptors are invariant to illumination changes, are discriminative, and resistant to noise and image blurring. The computational complexity of finding texture descriptors is low compared to other feature extractions [**?**]. The proposed work uses Hilbert curve-based descriptor (HCBD) [**?**], gray code-based descriptor (GCBD) [**?**], Local ternary pattern (LTP) [**?**] and Median ternary pattern (MTP) [**?**] to extract texture features from the fingerprint's consistent region.

Hilbert curve and Gray code-based descriptors: Hilbert curve and gray code curve are space-filling curves that are used to preserve the mapping of multidimensional points in a one-dimensional space. Two descriptors, namely Hilbert curve-based descriptor (HCBD) and Gray code-based descriptor (GCBD) are followed in this work. These descriptors are robust and efficient, insensitive to illumination and intensities, and rotation, scaling and shift invariant. These descriptors, unlike the traditional descriptors, consider a window of 4×4. Let I, $P_s^{(i,j)}$ be the starting pixel of a 4×4 window having gray value $I_1^{(i,j)}$ in a gray-scale image. The n^{th} neighbor of $P_s^{(i,j)}$ is denoted as $P_n^{(i,j)}$ in Hilbert and gray code pattern traversal, where n is a positive integer ($n \in [1, N]$) and $N = 16$. The Hilbert curve based operator is defined in Equation (5), (6) and (7).

$$HCBD_l(i,j) = \sum_{n=2}^{N/2} sign(I_n^{(i,j)} - I_{n-1}^{(i,j)}) \times 2^{n-1} \tag{5}$$

$$HCBD_u(i,j) = \sum_{n=N/2+2}^{N} sign(I_n^{(i,j)} - I_{n-1}^{(i,j)}) \times 2^{n-1} \tag{6}$$

$$HCBD = \begin{cases} [HCBD_l \oplus HCBD_u || 0], & if sign(I_{N/2+1}^{(i,j)} - I_{N/2}^{(i,j)}) = 1 \\ [HCBD_l \oplus HCBD_u || 1], & if sign(I_{N/2+1}^{(i,j)} - I_{N/2}^{(i,j)}) = -1 \end{cases} \tag{7}$$

where $\oplus$ is XOR operator and $||$ is the concatenation operator. The HCBD operator is illustrated in Fig.10.

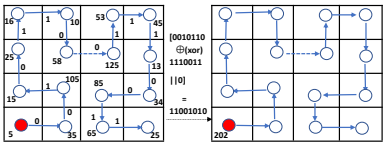


Figure 10: Illustration of HCBD operator.

Similarly, gray code curve-based operator is defined in Equation (8), (9) and (10).

$$GCBD_l(i,j) = \sum_{n=2}^{N/2} sign(I_n^{(i,j)} - I_{n-1}^{(i,j)}) \times 2^{n-1} \tag{8}$$

$$GCBD_u(i,j) = \sum_{n=N/2+2}^{N} sign(I_n^{(i,j)} - I_{n-1}^{(i,j)}) \times 2^{n-1} \tag{9}$$

$$GCBD = \begin{cases} [GCBD_l \oplus GCBD_u||0], & if sign(I_{N/2+1}^{(i,j)} - I_{N/2}^{(i,j)}) = 1 \\ [GCBD_l \oplus GCBD_u||1], & if sign(I_{N/2+1}^{(i,j)} - I_{N/2}^{(i,j)}) = -1 \end{cases} \tag{10}$$

where $\oplus$ is XOR operator and $||$ is the concatenation operator. The GCBD operator is illustrated in Fig.11.

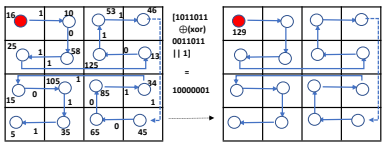


Figure 11: Illustration of GCBD operator.

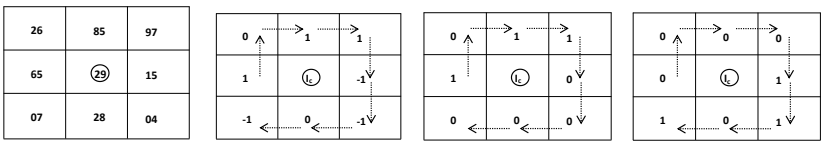
Local Ternary Pattern: Tan and Triggs [**?**] presented a new texture operator having three value codes $(-1, 0, 1)$ unlike two value codes in the local binary pattern. This operator is more robust towards noise. The mathematical expression of an LTP is given below.

$$LTP_{P,R} = \sum_{p=0}^{P-1} 2^P s(i_p - i_c) \tag{11}$$

where i_c denotes the gray values of the center pixel and i_p $(p = 0, ..., P-1)$ denotes the gray value of a neighbor pixel on a circle of radius R. P denotes the number of neighbors. Bilinear interpolation estimation is used to estimate the neighbors that do not lie exactly in the center of the pixels.

$$s(u) = \begin{cases} 1 & u \geq t \\ 0 & -t < u < t \\ -1 & u < -t \end{cases} \tag{12}$$

where t is a threshold value. Now, the LTP is the concatenation of histograms obtained from both LTP_u, LTP_l, where $LTP = LTP_u||LTP_l$ as shown in Fig. 12. A single LTP feature descriptor is of size 512.



(a) Orginal image block of size 3×3.

(b) LTP code $1011(-1)(-1)0(-1)$ with center pixel value 29 and threshold $t = 5$.

(c) LTP_u code 10110000 and I_c is assigned to 176.

(d) LTP_l code 00001101 and I_c is assigned to 13.

Figure 12: Illustration of LTP operator.

Median Ternary Pattern: Median filters perform well in the presence of noise. Therefore, a method that exploits the median can more accurately extract the information from any texture than the local binary patterns. Bashar et al. [**?**] proposed Median Ternary Pattern(MTP) that combines the advantages of median and quantization of the pixel's gray scale value. The MTP code is derived using the following equations:

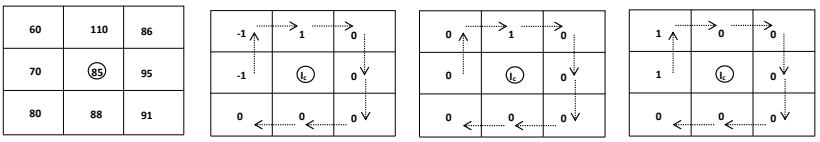
$$s'(u) = \begin{cases} 1 & u \geq m_c + t \\ 0 & m_c - t < u < m_c + t \\ -1 & u < m_c - t \end{cases} \tag{13}$$

Here, u is a neighbor gray level, m_c is the local median, and t is a user specified threshold. The feature code length is 3^8. To reduce the feature vector to 2×2^8, MTP is further split into MTP_u and MTP_l as shown in Fig. 13. Final MTP is concatenation of both MTP_u and MTP_l. The MTP_u can be derived as follows.

$$MTP_u = \sum_{p=0}^{7} f_u(s'(i_p)) \times 2^P, f_u(x) = \begin{cases} 1 & x = 1 \\ 0 & otherwise \end{cases} \tag{14}$$

MTP_l is derived using the following equation:

$$MTP_l = \sum_{p=0}^{7} f_l(s'(i_p)) \times 2^P, f_l(x) = \begin{cases} 1 & x = -1 \\ 0 & otherwise \end{cases} \tag{15}$$



(a) Orginal image block of size 3×3.

(b) MTP code $(-1)(-1)10000$ with median 86 and threshold $t = 10$.

(c) MTP_u code 0010000 and I_c is assigned to 16.

(d) MTP_l code 1100000 and I_c is assigned to 96.

Figure 13: Illustration of MTP operator.

Texture based feature extraction methodology: The proposed texture-based method combines the features from HCBD (256 bits), GCBD (256 bits), LTP (512 bits) and MTP (512 bits). The steps are as follows:

- **Step-1:** Around each minutiae point in the consistent region, HCBD features f_{hcbd}, GCBD features f_{gcbd}, LTP features f_{ltp}, and MTP features f_{mtp} are extracted with a window of size $w \times w$.
- **Step-2:** All f_{hcbd},f_{gcbd},f_{ltp},f_{mtp} features from all minutia patches are combined and these texture features are defined by building a histogram as F_{hcbd},F_{gcbd},F_{ltp},F_{mtp}.
- **Step-3:** Normalized feature vector F_{T1} is obtained by applying z-score normalization on the concatenated features F_{hcbd}, F_{gcbd} and F_{ltp} (not including F_{mtp}).

  F_{T1} = z-score($F_{hcbd}||F_{gcbd}||F_{ltp}$).
- **Step-4:** Normalized feature vector F_{T2} is obtained applying z-score normalization on all the concatenated features F_{hcbd}, F_{gcbd}, F_{ltp} and F_{mtp}.

  F_{T2} = z-score($F_{hcbd}||F_{gcbd}||F_{ltp}||F_{mtp}$).

  Here, z-score, denoted by z, is $z = (x - \mu)/(\sigma)$, where μ is the mean value of the features and σ is the standard deviation of the features.

Furthermore, normalized relative minutia feature vectors and texture-based feature vectors are fused to obtain a hybrid feature vector(F_h).

4.2.4 Prominent feature selection

In the method discussed so far, a combination of features (F_h) of a fingerprint image data is considered. An initial investigation revealed that all feature vector components do not equally contribute to a distinct feature vector generation. Thus the need to select the prominent feature arises. To select the prominent features from the combined feature vector, a genetic algorithm-based feature selection technique is proposed.

Genetic algorithm(GA) [**?** **?**] is a powerful optimization technique to select the best subset of features from a feature vector. The combined feature vector obtained from the fingerprint image is optimized through GA. This optimized vector, or prominent feature set, is smaller in size but contains more relevant information. This vector is less prone to noise and has distinct information. In this work, different GA parameters are chosen based on an empirical experiment and the optimized values of different GA parameters are shown in Table 1. The experimental procedure for the procedure is as follows.

- **Step-1:** An initial population size of 10 is generated. The length of each feature vector (F_h), henceforth referred as chromosome, is L.
- **Step-2:** The fitness value or accuracy of each chromosome is determined. For the proposed method, KNN classifier is used to evaluate this parameter by dividing the total feature set into 2 parts, namely training set and testing set.
- **Step-3:** Rank-based selection method is used to find the fittest chromosome for mating.
- **Step-4:** A 2-point crossover is applied on 80% of the selected chromosome, and the rest 20% is added through elitism. This process generates new offsprings.
- **Step-5:** Mutations in the chromosome is performed with probability 0.01 to generate a new population.
- **Step-6: Step-2** to **Step-5** is performed until number of generations reach 100.

Table 1: GA constraints information.

GA parameters	Value
Initial population size	10
No of generation	100
Crossover probability	0.80
Mutation probability	0.01

By applying GA on the combined feature set F_h, the prominent features are obtained.

4.2.5 Discriminant feature vector generation

The indirect feature vector should be distinct, that is, the intra-class fingerprint feature vectors must be distinguishable from the inter-class ones. For this reason, a metric learning-based approach [**?**] is used on the prominent features selected by GA.

Most of the metric learning-based methods depends on learning a Mahalanobis distance [**?**]. The Mahalanobis distance corresponding to the matrix M is the mapping d_M: $R^d \times R^d \rightarrow R$ which is defined as follows.

$$d_M = \sqrt{(x_i - x_j)^T M (x_i - x_j)} \tag{16}$$

where $(x_i, x_j) \in R^d$, M is a positive definite matrix and subject to constraints between pair of points which is given below.

$$\begin{cases} d_M(x_i, x_j) \leq u & if (x_i, x_j) \in S(similar) \\ d_M(x_i, x_j) \geq l & if (x_i, x_j) \in D(dissimilar) \end{cases} \tag{17}$$

Note that the distance metric, d_M, satisfies the following three criteria [**?**]:

- Coincidence: $d_M(x_i, x_j) = 0 \Leftrightarrow x_i = x_j, \forall x_i, x_j \in R^d$.
- Symmetry: $d_M(x_i, x_j) = d_M(x_y, x_i), \forall x_i, x_j \in R^d$.
- Triangle inequality: $d_M(x_i, x_k) \leq d_M(x_i, x_j) + d_M(x_j, x_k), x_i, x_j, x_k \in R^d$.

This work uses Information Theoretic Metric Learning (ITML) technique [**?**] to find a distance metric (M), which should be as similar as possible to an initial pre-defined distance M_0. ITML minimizes the Kullback-Leibler (KL) divergence [**?**] between $p(x|M_0)$ and $p(x|M)$, subject to two constraints on the data.

$$\min_{M} \; KL(p(x|M_0) \;||\; p(x|M)) \tag{18}$$

$$s.t. \begin{cases} d_M(x_i, x_j) \leq u & if (x_i, x_j) \in S(similar) \\ d_M(x_i, x_j) \geq l & if (x_i, x_j) \in D(dissimilar) \end{cases} \tag{19}$$

where $KL(p(x|M_0) \;||\; p(x|M)) = 0.5 \times (trace(MM_0^{-1}) - logdet(MM_0^{-1}) - n)$, for $n \times n$ matrices M and M_0. This is an optimization problem that can be solved using LogDet optimization [**?**], which takes the following form:

$$\begin{aligned} \min_{M,\xi} \; & 0.5 \times [(trace(MM_0^{-1}) - logdet(MM_0^{-1}) - n) + \\ & \eta((trace(diag(\xi) diag(\xi_0^{-1})) - logdet(diag(\xi) diag(\xi_0^{-1})) - n))] \end{aligned} \tag{20}$$

$$s.t. \begin{cases} trace((x_i - x_j)^T M (x_i - x_j)) \leq \xi_c(i,j) \\ \quad if (x_i, x_j) \in S(similar) \\ trace((x_i - x_j)^T M (x_i - x_j)) \geq \xi_c(i,j) \\ \quad if (x_i, x_j) \in D(dissimilar) \end{cases} \tag{21}$$

where ξ is slack variable and η is slack parameter. This problem can be optimized using gradient methods combined with iterated projections to fulfill the given constraints. This work adopts the solution proposed in [**?**] for Equation (20), subject to the constraints specified in Equation (21). The discriminative feature vector learning process is presented in Algorithm 2.

Algorithm 2: Discriminant feature vector generation

Input : $X_{40 \times L_g}$, set of feature vectors for $l(= 4)$ fingerprint instances of $q(= 10)$ individuals.
Y, class labels for $X_{40 \times L_g}$
M_0, an identity matrix of size $n \times n$ $(n = L_g)$
$\xi(\in [-4 : 4])$,the slack variable
$N(= 1000)$, the number of iterations

Output: Discriminant feature vector f_d.

(1) **Create** a constraint matrix $T_{(n(n-1)/2) \times 4}$ from Y,
where, columns 1, 2 contains the indices of the feature vector pairs,
column 3 contains similar(1) or dissimilar information(-1) and
column 4 contains the upper and lower bounds for similarity / dissimilarity constraints

(2) **foreach** $\xi_i \in [-4 : 4]$ **do**

(3) calculate M_i

(4) **Transform** X to $X' = X \times M_i$.
Then k-nearest neighbor with 3-fold cross-validation is applied and
average accuracy is stored in an array $acc[i]$.

(5) **Find** ξ_i having maximum accuracy, $max(acc[i])$.
Let $\xi_i = a \in [-4, 4]$.

(6) **Find** M_f from X and Y, with the values $\xi_i = a$, M_0, T .

(7) **Encode** any feature vector f to a discriminant feature vector f_d as $f \times M_f$.

(8) **return** f_d

This discriminant feature vector is stable and produces high similarity for intra-class feature vectors and more separability for inter-class feature vectors.

4.2.6 Key generation

The discriminant feature vector is stable and robust, but a fractional change leads to a different codeword. Different quantization methods are known in the literature to solve this issue. In this work, the feature vector are quantized into binary form to ensure its stability. Note that the binary feature vector does not reveal anything about raw biometric feature information and thus, it can be used as a key for different cryptographic applications.

The quantization method proposed in this work is stated with an example in the following.

- **Step-1:** Divide the discriminant feature vector f_d into non-overlapping segments(s). Each non-overlapping segment is of a given length, say 9, then the number of segments is $n = \lceil l/9 \rceil$, where l is length of feature vector.
  For example,
  Let $f_d = [20, 30, 40, 50, 55, 120, 43, 56, 77, 89, 56, 68, 97, \ldots]$
  Consider a segment $s = [20, 30, 40, 50, \underline{55}, 120, 43, 56, 77]$.

- **Step-2:** For each segment s, find the difference of consecutive feature components in it, i.e. $df_i = s_i - s_{i-1}$. where df_i is the difference of consecutive feature component of the i^{th} entry
  Thus for $i = 2, 3, \ldots, 9$ and $d_1 = 0$,
  $df = [0, 10, 10, 10, \underline{5}, 65, -77, 13, 21]$.

- **Step-3:** Compute the difference between all neighboring values from the central value df_c in that segment s. Let the resultant value be $rf_i = df_i - df_c$.
  $rf = [-5, 5, 5, 5, \underline{5}, 60, -82, 8, 16]$.

- **Step-4**: Compute the median of the resultant values obtained after **Step-3** in the segment. Let the median value be $m = median(rf_i)$,$i = 1, 2, \ldots, 9$.
  $m = 5$.

- **Step-5:** Quantize the feature component rf_i in the segment s using

  If $(rf_i \geq m)$, then 1 ; else 0.

  The binary vector for s is [011111011].

- **Step-6:** The binary feature vector f_b is the concatenation of binary vectors obtained from all n segments.
  $f_b = [011111011...]$

The stable key generation from f_b is as follows:

- **Step-1:** Obtain $F_{i,j}$, a combinations of binary feature sequences (f_b),
  where n-fingerprints $1 \leq i \leq n$ and l-samples of each fingerprint $1 \leq j \leq l$.

- **Step-2:** Combine f_b for l-instances of a fingerprint into a $l \times z$matrix M,
  ($l \times z$ is the feature vector dimension).

- **Step-3:** Select the consistent bit positions
  for values of feature sequences across l-columns of M that are same.
  The consistent bit position is set to 1 in a auxiliary vector (A) having dimension $l \times z$).

- **Step-4:** Using A, extract a **key**(b_k) from f_b.

4.3 Experimental results

The results observed vis-a-vis the objectives of the experiments are presented in the following subsections.

4.3.1 Randomness testing of keys

The randomness of the generated keys was checked using the NIST test suite [**?**] and Diehard test suite [**?**]. More than 10^6 bits were generated using the proposed key generation approach. NIST tests results are presented in Table 2, and Diehard test results are shown in Table 3.

Table 2: Randomness checking of key with NIST test suite

Statistical test	P-value	Conclusion
Frequency	0.134325	PASSED
Block frequency	0.832618	PASSED
Runs	0.832618	PASSED
Longest run	0.134325	PASSED
Rank	0.056732	PASSED
FFT	0.056732	PASSED
Non-overlapping template	0.644146	PASSED
Overlapping template	0.360785	PASSED
Universal	0.360785	PASSED
Linear complexity	0.832618	PASSED
Serial (1)	0.016812	PASSED
Serial (2)	0.553146	PASSED
Approximate entropy	0.360785	PASSED
Cumulative sums (1)	0.941324	PASSED
Cumulative sums (2)	0.832618	PASSED

Table 3: Diehard test results of proposed key.

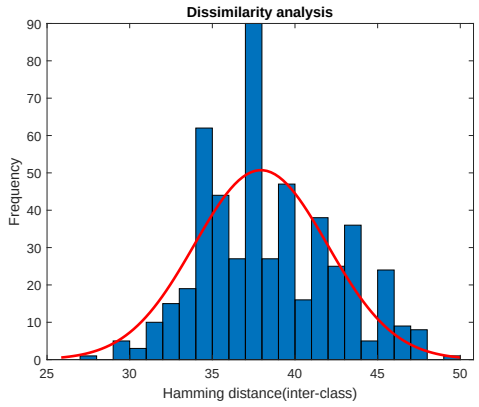
STATISTICAL TEST	P-VALUE	ASSESSMENT
Diehard Birthdays Test	0.87063932	PASSED
Diehard OPERM5 (Overlapping Permutation) Test	0.23147305	PASSED
Diehard 32x32 Binary Rank Test	0.25887712	PASSED
Diehard 6x8 Binary Rank Test	0.77824656	PASSED
Diehard Bitstream Test	0.86793168	PASSED
Diehard OPSO (Overlapping Pairs Sparse Occupancy) Test	0.69799364	PASSED
Diehard OQSO (Overlapping Quadruples Sparse Occupancy) Test	0.73596844	PASSED
Diehard DNA Test	0.42705856	PASSED
Diehard Count the 1s (stream) Test	0.03716524	PASSED
Diehard Count the 1s (byte) Test	0.95128245	PASSED
Diehard Parking Lot Test	0.28185127	PASSED
Diehard Minimum Distance (2d Circle) Test	0.78042741	PASSED
Diehard 3d Sphere (Minimum Distance) Test	0.69806026	PASSED
Diehard Squeeze Test	0.98213760	PASSED
Diehard Sums Test	0.05850323	PASSED
Diehard Runs Test	0.58383737	PASSED
Diehard Craps Test	0.93474779	PASSED
Marsaglia and Tsang GCD Test	0.69006537	PASSED
STS Monobit Test	0.35021856	PASSED
STS Runs Test	0.18441023	PASSED
STS Serial (Generalized) Test	0.76728552	PASSED
RGB Bit Distribution Test	0.67057707	PASSED
RGB Generalized Minimum Distance Test	0.62776791	PASSED
RGB Permutations Test	0.77556779	PASSED
RGB Lagged Sum Test	0.75416640	PASSED
RGB Kolmogorov-Smirnov Test	0.64178582	PASSED
DAB Byte Distribution Test	0.72990963	PASSED
DAB DCT (Frequency Analysis) Test	0.58474525	PASSED
DAB Fill Tree Test	0.62571825	PASSED
DAB Fill Tree 2 Test	0.28374025	PASSED
DAB Monobit 2 Test	0.88985527	PASSED

Table 4: Dissimilarity analysis.

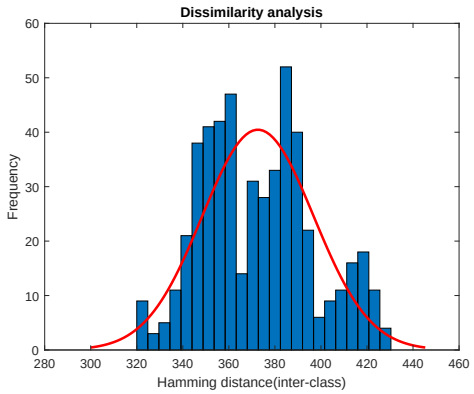
Feature vector	Hamming distance(inter-class)
Minutia based relative feature	36-38 bits out of 72 bits
Minutia and texture1 combined features	370-380 bits out of 720 bits
Minutia and texture2 combined features	540-550 bits out of 1024 bits

4.3.2 Dissimilarity of keys

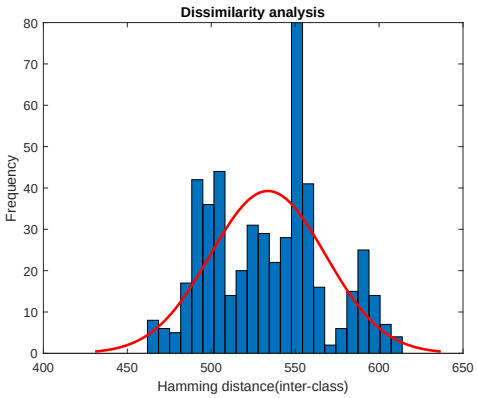
Hamming distance function was used to calculate the dissimilarity among individual keys. For this purpose, hamming distances were computed between the key generated from a fingerprint instance of one user and the keys from inter-instances of all other users. Two sets of hybrid feature vector combinations, that is, MT1 (minutia and texture1 (HCBD, GCBD, LTP)) and MT2 (minutia, texture2 (HCBD, GCBD, LTP, MTP)) were considered to check the dissimilarities among keys. The dissimilarities between different keys generated from different feature combinations is shown in Table 4. From this analysis, it is evident that an adversary cannot guess the key of another user by analysing their own keys. Dissimilarity analysis for different feature vector combinations is illustrated in Fig. 14.



(a) Dissimilarity analysis for minutia based relative feature vector.



(b) Dissimilarity analysis for combination of minutia and three texture based feature vector.



(c) Dissimilarity analysis for combination of minutia and all four texture based feature vector.

Figure 14: Dissimilarity analysis.

4.3.3 Distinctiveness analysis of keys

In this experiment, the performance of the proposed method was analyzed through key regeneration rate for different feature combinations, such as proposed minutia features only (M), combination of minutia and texture features MT1(minutia and texture1 (HCBD, GCBD, LTP)) and MT2(minutia, texture2 (HCBD, GCBD, LTP, MTP)). The key regeneration rate (kgr) is the number of times the key

is generated for successful authentication. The key regeneration rate for different feature combinations on different datasets is shown in Table 5.

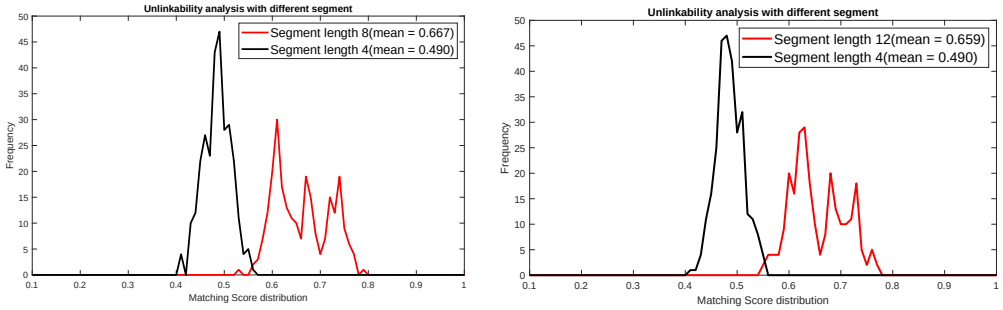
Table 5: Key regeneration rate using different feature(s).

Dataset	M-kgr	MT1-kgr	MT2-kgr
FVC 2002	92.43%	94.44%	97.22 %
FVC 2004	92.66%	93.02%	96.13 %
High Resolution-Fingerprint(HRF) DB1	96.25 %	96.35 %	98.00%
High Resolution-Fingerprint(HRF) DB2	90.68 %	91.45 %	95.68 %

Table 6: Key regeneration rate with or without metric learning.

Dataset	Without metric learning(kgr)	With metric learning(kgr)
FVC 2002	91.43%	97.22%
FVC 2004	90.66%	96.13%
High Resolution-Fingerprint(HRF) DB1	92.25 %	98.00 %
High Resolution-Fingerprint(HRF) DB2	90.88 %	95.68 %

The missing key scenario would be the worst, as it means that either the user has missed the secret key or the intruder knows the secret key. A new key can be generated by changing the segment length in the key generation mechanism described in Section 4.2.6 or by changing the binarization scheme. So, the proposed method ensures the revocability of the key which is not dependent on the missing one.



(a) (b)

Figure 15: Illustration of unlinkability analysis of keys generated for different segments.

4.3.4 Performance analysis with different fingerprint images

Analysis of key regeneration rate on fingerprint images of different resolutions and qualities were performed. For this purpose, the datasets were categorized into four categories: (i) low-quality (FVC 2002 DB3, FVC 2004 DB2), (ii) synthetic (FVC 2002 and 2004 DB4), (iii) normal (other FVC 2002 and 2004 databases) and (iv) high-resolution (HRF). The key regeneration rate of different categories of fingerprint datasets are shown in Table 7. From the table, it is clear that the proposed method is suitable for any kind of fingerprint image.

4.3.5 Computational complexity analysis

The proposed method can be used to generate keys of different lengths according to the individual cryptographic application's requirements. For low-cost and small applications, key size should be less compared to large application. The computational times vary in the range 152ms-600ms. The break-up times of different computations are shown in Table 8.

Table 7: Key regeneration rate on various fingerprint category.

Fingerprint category	M-kgr	MT1-kgr	MT2-kgr
Low-quality	92.69%	93.24%	95.52 %
Synthetic	92.75%	94.45%	97.92%
High Resolution	93.46 %	93.90 %	96.84 %
Normal resolution and quality	93.13 %	93.65 %	96.74%

Table 8: Computational complexity analysis.

Sub-steps	Computational time(in ms)
Pre-processing	0.534
Relative feature vector generation	38.212
Texture feature vector generation	112.245
Discriminant feature vector and key generation	113.074

5 Work in progress and Future plan of work

We are currently improving the uniqueness of the extracted features. Using just the direct features, we are not able to get the same feature for different snapshots of a particular biometric. We propose using a combination of indirect and direct features.

The plan of work is as follows:

1. Creation of a virtual dataset for evaluation of traditional as well as proposed methods
2. Extracting indirect features from a given biometric
3. Using a combination of indirect and direct features for key generation.
4. Generating an index space using the keys and analysing the search time and penetration rate for the index.
5. Use of multi-index to further reduce the search time.
6. Technique to combine multiple modalities of biometric to further improve the search time.

References