A novel fingerprint indexing scheme using dynamic clustering

Ashima Jain¹ · Munaga V. N. K. Prasad²

Received: 27 May 2016 / Accepted: 2 August 2016 / Published online: 19 August 2016
© Springer International Publishing Switzerland 2016

Abstract With the increase in adoption of biometric data for various identification processes, the need for efficient indexing techniques is also on the rise as it is computationally impossible to match the query template against every enrolled entry in the database. In this paper, we have proposed two robust feature extraction techniques from fingerprint images followed by an indexing method based on clustering and m-ary trees. We extract robust translation and rotation invariant features from $O(n)$ triangles obtained from the triangle spiral constructed for every fingerprint. The other method involves construction of M rectangles around every reference minutiae and calculation of features based on neighboring relation. Experiments with the benchmark databases confirm the superiority of our approach to the other existing techniques.

Keywords Biometrics · Fingerprints · Indexing · Dynamic clustering · Trees

1 Introduction

As advancements are being made in every field of science and technology, the need for keeping these systems safe and secure is also increasing. As a result, the use of biometric databases for various security purposes is gaining importance. Among the many physiological (face, iris, fingerprints,

DNA) and behavioral (voice, gait, typing pattern) biometrics [1], fingerprints are the most acceptable due to their accessibility, stability, uniqueness, etc.

Biometric systems can be put to use both in verification and identification. In verification, the id with which the template is to be matched is already available, that is, it is 1:1 matching. However, identification involves searching the whole database to retrieve the matching template. This 1:N matching is not only time consuming but also prone to various errors [2]. The idea of how big the databases are can be got from Aadhar, which is issued by UIDAI [3] and consists of data of more than 730 million people. To make the identification process more efficient, different classification and indexing techniques are used. Classification involves dividing the complete set of fingerprints into distinct classes and then searching only in that particular class. But, the major drawback with this system is that one can only have a countable number of distinct classes, namely arch, tented arch, right loop, left loop and whorl [4]. Further, the density in each is also not uniform. Indexing, on the other hand, works on a handful of robust features extracted from the templates and then searching the database with the same features [5].

We have proposed a novel indexing approach for fingerprints in this paper. The rest of the paper is organized as follows. Section 2 describes the existing state-of-the-art techniques used for fingerprint indexing. Our proposed methodology including the two techniques for feature extraction are presented in Sect. 3 and the results of the same in Sect. 4. Finally, we conclude our work in Sect. 5.

2 Related work

There have been many fingerprint indexing approaches that have been proposed before using minutiae [10–15], local

✉ Munaga V. N. K. Prasad
mvnkprasad@idrft.ac.in

Ashima Jain
ashimajain2595@gmail.com

¹ Department of EEE, IIT Guwahati, Guwahati, India

² IDRBT, Hyderabad, India

ridge orientation [7], singular points [6], SIFT features [9] and matching scores [8]. The ones based on minutiae have proved to be the most robust and we discuss some of the state-of-the-art techniques of fingerprint indexing below.

Germain et al. [10] used the minutiae in the fingerprints to construct all possible combination of triangles. The length of the triangle, the ridge count between the sides, and the angles of the triangles were the features used for index key generation. Fast Look Up Algorithm for String Homology (FLASH) was used to compute the index. Bhanu et al. [11] also utilized minutiae triplets in their method but used different features which included the angles of the triangle, its longest, triangle handedness and direction. However, these approaches are computationally costly as they produce $O(n^3)$ triplets. Bebis et al. [12] improvised this cost, reducing it to $O(n)$ using Delaunay triangulation. The features used were the ratio of the largest side with the smallest sides and the cosine angle between the smallest sides. Low-order Delaunay Triangulation [13] reduced the effect of spurious and missing minutiae thereby further increasing the performance. The authors in this paper included features such as triangle handedness, minimum and median angles and length of the longest side in the triangle. In [14], Delaunay Triangulation was used to extract ridge-associated features upon which clustering is performed using K-means algorithm.

Minutia Cylinder Code (MCC) representation was used in [15] where each fingerprint was represented by a set of bit vectors and the indexing was done by Locality Sensitive Hashing (LSH). Each minutia point was represented by a 384-bit binary feature vector. But, Wang et al. [16] observed this representation to be bit correlated and reduced the length to 24 bits from 384 using compact binary hash codes.

In [17], Jayaraman et al. used hash table to index invariant spatial (distance) and directional (angle) information. A fixed length vector called Minutia Binary Pattern (MBP) is also built for each minutia for accurate matching at the time of searching. Briseno et al. [18] proposed a method to find reference points by taking the difference of orientations around a block to locate singular points, that is, core and delta. The indexing features used were ridge count, triangle sign, reference point features of minutiae and its relative direction. Another indexing technique [19] involves construction of fingerprint shell which was a special triangle curve formed from the minutiae features. Hausdorff distance was used to match the query and database templates. Out of the two methods proposed in this paper for feature extraction, one of them involves the construction of a similar triangle spiral for each fingerprint. These $O(n)$ triangles are then used to form the feature vector.

There are several other methods that have been proposed in the literature based on minutiae quadruplets instead of triplets. Iloanusi et al. [20] used all possible quadruplets that can be formed from the minutiae. They have used seven fea-

tures in the vector $F = \{\phi_1, \phi_2, \delta_1, \delta_2, \rho_1, \rho_2, \eta\}$ where, ϕ_1 and ϕ_2 are the differences of two opposite angles in the quadruplet, δ_1 and δ_2 are its diagonals, ρ_1 and ρ_2 are the heights of the inner parallelogram whose vertices are the mid-points of the quadruplet. Further, $\eta = 100\log_{10}(\tau v)$ where $\tau = \sqrt[3]{A_p} + \sqrt[4]{x_1 \times x_2 \times x_3 \times x_4}$ and $v = \sqrt[3]{A_q} + \sqrt{y_1 \times y_2}$. A_p is the area of the parallelogram, x_1, x_2, x_3 and x_4 are the lengths of the sides of the quadruplet, A_q is the area of the quadruplet, and y_1 and y_2 are the lengths of the sides of the parallelogram.

In [29], authors proposed a fingerprint pose estimation algorithm which can register fingerprints into a common finger coordinate system. A fingerprint indexing is designed by combining the binary template and Locality Sensitive Hashing indexing algorithm in [30]. A fixed length feature vector built from each minutia, known as Coaxial Gaussian Track Code (CGTC) was proposed in [31]. The feature vector is inserted into a Quantized Lookup Table (QLT).

The above techniques suffer mainly from two disadvantages. 1. The computation cost is very high as using all possible combinations of minutiae triplets and quadruplets result in $O(n^3)$ or $O(n^4)$ complexity. 2. Moreover, when a dynamic database is to be considered, that is, where new users are enrolled while certain old ones have to be removed on a periodic basis, the index space has to be generated afresh again and again. Our proposed technique overcomes both these difficulties. Using the triangle spiral for feature extraction, gives us only $O(n)$ triangles leading to low computation cost. Second, since dynamic clustering is employed, it is not necessary to disturb all of the existing clusters but only the ones in which new admissions are to be made. In addition, m-ary trees facilitate further in decreasing the search space and hence a high hit rate at a low penetration rate.

3 Proposed method

This section describes the proposed indexing approach for the fingerprints. The proposed approach follows these steps:

1. Feature extraction from a fingerprint
2. Dynamic clustering of the feature vector
3. Generation of a tree from each cluster
4. Retrieval of the candidate set.

3.1 Feature extraction from a fingerprint

The first step involves extraction of minutiae from the fingerprint image. Each minutia can be represented by $m_i = \{x_i, y_i, \theta_i, t_i\}$, where $\{x_i, y_i\}$ denote its coordinates, θ_i is its orientation with respect to x-axis and t_i the type of minutia, that is, bifurcation or end. Figure 1 shows a sample fingerprint image having different types of minutiae along with the

Fig. 1 Singular points and minutiae types in a fingerprint

singular points, core and delta. In this paper, we have proposed two methods for feature extraction which have been discussed in detail below.

3.1.1 Neighboring relation using rectangle

This method for feature extraction utilizes the multiline neighboring relation generation approach as proposed in [28]. Let $M = \{m_1, m_2, \dots, m_n\}$ denote the set of minutiae in the fingerprint. We select a reference minutia $m_r = \{x_r, y_r, \theta_r, t_r\}$ and construct L rectangles of length l and width w centered at (x_r, y_r) with orientations $\theta_r, \theta_r + \frac{\pi}{L}, \theta_r + \frac{2\pi}{L}, \dots, \theta_r + \frac{(L-1)\pi}{L}$. Figure 2 shows L rectangles drawn at different orientation angles around a reference minutia. We should choose three rectangles at best, because if we choose more number, the rectangles will overlap each other, so that each minutiae point may fall in more than one rectangle,

Fig. 2 L rectangles with different orientations around the reference minutiae

which causes redundancy. All the minutiae falling inside the constructed rectangle are selected and the rotation invariant distance and orientation angle of the selected minutiae with the reference minutiae are computed as:

$$\chi = (x_j - x_r) \cos \theta_r + (y_j - y_r) \sin \theta_r$$

$$\gamma = (x_j - x_r) \sin \theta_r - (y_j - y_r) \cos \theta_r$$

$$d_{ij} = \sqrt{\chi^2 + \gamma^2}$$

$$\psi_{ij} = \begin{cases} \theta_j - \theta_r & \theta_j > \theta_r \\ \theta_j - \theta_r + 360 & \text{otherwise} \end{cases}$$

Here, d_{ij} denotes the distance from reference minutiae to the i th minutia in the j th rectangle and ψ_{ij} denotes the orientation angle of the i th minutia in the j th rectangle. The above steps are repeated for all the minutiae present in a fingerprint template being considered as the reference minutia. Finally, from each fingerprint, we get a set of feature vectors, $F = \{f_1, f_2, \dots, f_n\}$, where $f_i = \{d_{ij}, \psi_{ij}, \theta_i, t_i, id_i\}$. θ_i is the orientation of the i th minutiae falling inside the j th rectangle, t_i its type and id_i is the fingerprint id of the user.

3.1.2 Triangle spiral

The second approach for feature extraction involves the construction of $O(n)$ triangles for each fingerprint. Let $M = \{m_1, m_2, \dots, m_n\}$ denote the set of minutiae in the image. The distance of each of the minutiae from the core is calculated and sorted from lowest to highest. Let $d = \{d_1, d_2, \dots, d_n\}$ be the set denoting the sorted distances. A user-defined parameter d_0 is chosen to be added to all the elements of the set d resulting in the set $d' = \{d_1 + d_0, d_2 + d_0, \dots, d_n + d_0\}$. These distances are used to construct the triangle spiral which consists of n contiguous right-angled triangles. The lengths in the set d' act as the lengths of the hypotenuses of these triangles. The triangle spiral is made such that the distance d_0 serves as the leg of the first triangle and $d_1 + d_0$ as the hypotenuse. The leg of the second right-angled triangle is the same as the previous one's hypotenuse and the length $d_2 + d_0$ in turn becomes its hypotenuse. The process thus continues with the legs of the current triangle same as the hypotenuse of the previous one. The heights of all these triangles can be calculated using Pythagoras Theorem. The first three right-angled triangles of the spiral are shown in Fig. 3a. Finally from each fingerprint having n minutiae, we obtain a feature vector set $F = \{f_1, f_2, \dots, f_n\}$, where $f_i = \{\theta_i, \phi_i, h_i, t_i, id_i\}$. h_i is the height of the i^{th} triangle, ϕ_i is the angle opposite to height, θ_i is the orientation of the minutia which is i^{th} farthest from the core, t_i its type and id_i is the fingerprint id of the user. Figure 3b shows the triangle spiral constructed from a fingerprint.

Fig. 3 Illustration of construction of triangle spiral. **a** Geometric features of the right triangles, **b** Complete spiral of a fingerprint. Core is marked in blue and minutiae points in red in the fingerprint

3.2 Dynamic clustering of the feature vector

Let the set F_i denote the feature vector set extracted from the i^{th} enrolled image. Thus, from p enrolled images, we will get the feature vector set as $F = \{F_1, F_2, \dots, F_p\}$.

If the feature extraction is carried out using the first method, the feature vector F looks like

$$F = \begin{pmatrix} d_{11} & \psi_{11} & \theta_1 & t_1 & id_1 \\ d_{12} & \psi_{12} & \theta_2 & t_2 & id_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{mL} & \psi_{mL} & \theta_m & t_m & id_m \end{pmatrix}$$

where m th minutia lies in the L th rectangle when the last minutia of the p th image is taken as the reference minutia.

Alternatively, let there be a total of q minutiae in the p images, then the features from all the fingerprints extracted using triangle spiral are stored in a matrix of size $q \times 5$ as shown.

$$F = \begin{pmatrix} \theta_1 & \phi_1 & h_1 & t_1 & id_1 \\ \theta_2 & \phi_2 & h_2 & t_2 & id_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \theta_q & \phi_q & h_q & t_q & id_q \end{pmatrix}$$

Clusters are formed using only the first three elements of the feature vector (d, ψ, θ) if obtained from 3.1.1 or (θ, ϕ, h) from 3.1.2. Linde–Buzo–Gray (LBG) algorithm [21] is used to carry out the dynamic clustering. As the size of biometric databases is not fixed, the more commonly used clustering technique, k-means algorithm is less preferable than dynamic clustering as here the user does not have to set the number of clusters k , beforehand. In the proposed approach, a threshold T is set such that if the number of members in a given cluster is more than the threshold, it gets divided into two using k-means where k is fixed at 2. The mean of the whole cluster is computed after which it is displaced by a small value, ϵ . We do not want the new initial means to go very far from each other or out of the cluster as a result of which the value of ϵ is chosen to be very small. For instance, in this paper, we have used $\epsilon = 10^{-15}$. After obtaining the two initial means,

k-means is applied to get the new clusters. This process is repeated till all the clusters have less than T elements. We finally get a list of the means of all clusters denoted by code vectors and the respective elements as well. The algorithm for the clustering procedure is presented in Algorithm 1 and its working depicted in Fig. 4.

Algorithm 1: lbg(data)

Input : $data = \{f_1, f_2, \dots, f_n\}$ – feature vector obtained
Output: Code Vectors and the corresponding elements

- 1 $m = \text{mean}(data(:, 1 : 3))$.
- 2 **if** $\text{length}(data) > T$ **then**
- 3   Perform k-means on the $data(:, 1 : 3)$ ($k = 2$) where the initial means of the cluster are $(1 + \epsilon) \times m$ and $(1 - \epsilon) \times m$.
- 4   Go to step 1 for performing clustering on the two newly generated clusters.
- 5 **else**
- 6   Add the mean of the data in the code vectors list and note the corresponding elements of the cluster.
- 7 **end**

3.3 Generation of a tree from each cluster

For each cluster, a generic (m -ary) tree is built. M -ary tree implies that each parent can have m or less children. The children having the same parent are called siblings. A node in the tree contains 5-D data viz.

$\{d_{ij}, \psi_{ij}, \theta_i, t_i, id_i\}$ (from 3.1.1) or $\{\theta_i, \phi_i, h_i, t_i, id_i\}$ (from 3.1.2) and points to two other nodes, one to its first child and second to its next sibling. A 3-ary tree with both the pointers is illustrated in Fig. 5. The root of the tree is formed by the code vector that is the mean of the cluster and it does not have a sibling. The siblings are such that the value of their first element is less than or equal to that of their parent. Further, to make the tree balanced, the siblings are selected in such a way that the number of descendants is nearly the same for each sibling. Let the order of the tree be m and the number of members in the cluster be e denoted by $\{f_1, f_2, \dots, f_e\}$. First, we sort these e elements on the basis of the first dimension of the data (θ or d). While fixing the children of the root, it is done in a way that the number of descendants of

Fig. 4 Illustration of the working of Linde–Buzo–Gray (LBG) Algorithm

Fig. 5 Structure of an m-ary tree showing the pointers to the first child and the siblings ($m=3$)

the m siblings remain close to each other. The member at the (e/m) th position is appointed as the first child of the root, the one at $2 \times (e/m)$ th position as its next sibling and the process continues till the last element, i.e., e th of the sorted list becomes the last sibling. In a similar fashion are the rest of the nodes of the tree filled up. The m-ary tree provides an additional advantage of a lower height compared to a binary one. The algorithm for the above-described process is given in Algorithm 2.

Algorithm 2: buildTree(elements)

Input : Elements of a given cluster

Output: An m-ary tree corresponding to the given cluster

- 1 Assign the mean of the cluster as the root of the m-ary tree.
- 2 Sort the elements of the cluster on the basis of the first column (i.e., θ or d) and select the indices such that the data are divided into m equal parts.
- 3 For e elements, the indices will be $\{e/m, 2 \times e/m, \dots, (m-1) \times e/m, e\}$.
- 4 Choose the element at these indices to be the children of the parent. The first sibling has the minimum θ (or d) while the last having the maximum θ (or d) of all.
- 5 The descendants of each sibling are chosen such that their θ (or d) are equal to or less than that sibling but more than the θ (or d) of the preceding sibling.
- 6 Go to step 2 till the number of descendants of each sibling does not reach zero.

3.4 Retrieval of the candidate set

For a query image, it is important to retrieve an accurate candidate list for a high hit rate. Let us denote the query image by Q and its feature vector set by F_q . F_q is computed by the same method as explained in Sect. 3.1. Next, the euclidean distance between each of the feature vector element of F_q and the code vectors is calculated. The clusters chosen for

retrieval of the candidate list are the ones closest to the corresponding feature vectors, i.e., with lower euclidean distances. Since there is always a high probability of distortions in biometric data, we not only search in the first nearest cluster but in the next ones as well. The probable candidates for that feature element is searched for in the respective trees and the top b matches are retrieved. The minimum distances of the top b matches are initialized to a very large value like 10000. As the search for the candidate is started in the tree, the first child of the root becomes both the *current* and the *head* node. We keep traversing to the next sibling until the value of the first dimension (θ or d) of the current node data is less than the query one. The searching part is then done for child of the *current* node that it, *current.child*, becomes the new *current* node. The siblings before and after the current node are called as *prev* and *next* and they are also marked. In case, the euclidean distance between the first three dimensions of the *head* node and the query element is less than the top b matches and the type of minutiae of the two also matches, we replace the maximum of the b distances by this value and retrieve the id. In addition, if the above-calculated distance is less than a threshold D , the children of the *head* node are also searched thus updating the *current* node to *head.child*. The next sibling of the current *head* node is appointed as the new *head*. Further, the *current* node is updated to *prev.child* and *next.child* if the absolute difference between the first dimensional data of the *current* and query element is less than t . The procedure of retrieval of candidate list is presented in Algorithm 3.

The probable matches for all members of the feature vector are found using the method described above after which the score corresponding to each id in the database is computed as the percentage of votes it receives out of all the enrolled ids. A threshold s is set and if the score of any *id* is more than s , it is retrieved to be a part of the candidate list for that query. Apart from the matching score strategy, rank-based retrieval has also been done where the *score* of each *id* decides its rank and then the ids with top r ranks are retrieved.

4 Experimental results

The proposed approaches have been evaluated on the Fingerprint Verification Competition Databases (FVC) 2002 and 2004 each comprising four databases, viz., DB1, DB2, DB3 and DB4. Each of these contain eight fingerprints each from 100 different individuals making a total of 800 images.

For every user, six out of the eight images are randomly chosen to belong to the enrolled dataset while the remaining two images are used as test images. The information about the coordinates and the orientation with respect to the x-axis of the minutiae is extracted using the software Verifinger SDK. However, since core is required for the construction of

---

Algorithm 3: [*mindist*, *id*] = retrieve(*node*, *query.data*, *mindist*)

Input : *node* - first child of the root, *query.data* - an element of the feature vector of query image, *mindist* - minimum Euclidean distance

Output: *id* of the retrieved candidate, *mindist* - minimum Euclidean distance

```

1 current = node.
2 head = node.
3 while current.data(1) < query.data(1) do
4   current = current.sibling.
5 end
6 Keep track of the previous(prev) and the next(next) siblings of current.
7 while head is not empty do
8   dist = euclidean norm of (head.data(1 : 3) - query.data(1 : 3)).
9   if dist < mindist and head.data(4) == query.data(4) then
10    mindist = dist.
11    id = head.data(5).
12  end
13  if absolute difference of (query.data(1) - current.data(1)) <  $t$  then
14    if dist <  $D$  or head.data == current.data or head.data == prev.data or head.data == next.data then
15      [mindist, id] = retrieve(head.child, mindist)
16    end
17  else
18    if dist <  $D$  or head.data == current.data then
19      [mindist, id] = retrieve(head.child, mindist)
20    end
21  end
22  head = head.sibling.
23 end

```

---

triangle spiral, we have only used the templates having core in them.

Hit Rate and Penetration Rate are used to quantitatively measure the performance of the two methods. Hit Rate which is defined as the percentage of queries for which the correct id is retrieved in the candidate list is calculated as $H_r = (N_c/N) \times 100\%$, where N_c = correctly indexed queries and N = total number of queries in the database. Penetration Rate gives the percentage of the enrolled templates searched for finding the correct match. It is computed as $P_r = (1/N \sum_{i=1}^N d_i/M) \times 100\%$, where d_i = size of candidate list retrieved for i th image, N = total number of queries and M = size of the database.

The following two subsections present the results obtained upon indexing using the two proposed techniques.

4.1 Neighboring relation using rectangles results

To find the optimal value of the various parameters, numerous experiments were conducted. The values of the hit and pen-

Table 1 Hit rate and penetration rate for different values of T for FVC 2002 DB2

HR	PR	T
94.89	1.4	50
97.31	2.4	70
100	3.1	100
100	3.4	150
99.27	3.4	200
98.54	3.5	300
98.54	3.5	500
99.27	3.6	1000

Table 2 Hit rate and penetration rate for different values of m for FVC 2002 DB2

HR	PR	m
100	3.2	3
100	3.3	4
100	3.1	5
99.27	3.3	6
100	3.3	7
100	3.3	10

Table 3 Hit rate and penetration rate for different values of t for FVC 2002 DB2

HR	PR	t
99.27	3.1	1
100	3.1	2
100	3.2	3
100	3.2	5
100	3.2	7
100	3.2	10
100	3.2	15

Table 4 Hit rate and penetration rate for different values of NN I and NN II for FVC 2002 DB2

HR	PR	NN I	NN II
97.06	2.1	70	–
97.08	2.1	50	–
100	7.6	50	50
100	6.7	30	30
100	5.7	30	20
100	4.8	30	15
100	4.2	25	15
100	3.1	25	10
97.8	2.3	20	10
98.54	3.0	24	10

Table 5 Hit rate and penetration rate for different values of D for FVC 2002 DB2

HR	PR	D
98.54	2.9	15
99.27	3.0	16
100	3.1	17
100	3.2	18
99.27	3.2	20

Table 6 Values of the user-defined parameters used in the experiments

	HR	PR	s	T	NN I	NN II	NN III	D	t	m
2002 DB1	100	2.1	60	30	22	9	6	18	2	5
2002 DB2	100	3.1	70	100	25	10	–	17	2	5
2002 DB3	100	3.1	70	30	14	9	2	18	2	5
2002 DB4	100	6.9	70	100	30	20	2	18	2	5
2004 DB1	100	5.0	70	70	26	19	–	18	2	5
2004 DB2	100	7.4	70	100	30	18	6	18	3	5
2004 DB4	100	6.3	70	70	21	10	–	16	1	5

Table 7 Hit rate and penetration rate at various matching score values for FVC 2002 databases

Score	DB1		DB2		DB3		DB4	
	HR	PR	HR	PR	HR	PR	HR	PR
0	100	25.6	100	24.2	100	43.8	100	26.5
10	100	15.3	100	16.0	100	16.6	100	16.1
20	100	11.8	100	13.8	100	15.3	100	15.0
30	100	8.1	100	10.9	100	12.6	100	13.4
40	100	5.3	100	8.2	100	9.4	100	11.5
50	100	3.3	100	6.0	100	6.9	100	9.9
60	100	2.1	100	4.4	100	4.7	100	8.2
70	98.45	1.3	100	3.1	100	3.1	100	6.9
80	98.45	0.9	97.8	2.1	96.05	2.1	99.20	5.6
90	97.67	0.7	94.89	1.5	96.05	1.4	98.41	4.5

etration rate at different values of the parameters are listed in Tables 1, 2, 3, 4, 5. The so obtained optimum values are presented in bold in these tables.

The length and breadth of the three rectangles are set at 320 and 70, respectively, as obtained from [28]. The threshold T is fixed at 100 such that if the size of the cluster > 100 , it will divide into two new clusters, otherwise not. The best results are obtained when the order of the tree m is kept at 5,

i.e., a 5-ary tree is made for all the clusters. For retrieval of the candidate list from the corresponding trees, top b matches are extracted. Owing to the high distortion in biometric databases, the search is carried out in the three nearest clusters thereby extracting NN I, NN II and NN III number of matching ids. For FVC 2002 DB1, their values come out to be 25, 16 and 0, respectively. As can be concluded from the table, the values of the other user-defined parameters D and t are 17 and 2, respectively. The best values of all the parameters involved in the method for all the databases have been tabulated in Table 6.

Further, the hit rate and penetration rate at various matching score values and ranks for both FVC 2002 and 2004 databases are presented in Tables 7, 8, 9, 10 and Figs. 6, 7, 8, 9.

Table 8 Hit rate and penetration rate at various ranks for FVC 2002 databases

Rank	DB1		DB2		DB3		DB4	
	HR	PR	HR	PR	HR	PR	HR	PR
1	88.37	0.3	75.91	0.2	86.84	0.4	61.90	0.3
2	97.67	0.5	83.21	0.5	92.10	0.8	73.81	0.5
3	98.45	0.7	89.78	0.7	94.73	1.3	82.54	0.8
4	98.45	1.0	91.97	1.0	97.36	1.8	85.71	1.0
5	98.45	1.3	94.16	1.2	97.36	2.2	88.89	1.3
6	99.22	1.5	94.89	1.4	98.68	2.6	90.48	1.6
7	100	1.8	94.89	1.7	98.68	3.5	92.06	1.8
8	100	2.0	96.35	1.9	100	3.9	92.85	2.1
9			96.35	2.2			94.44	2.4
10			96.35	2.4			94.44	2.6
11			96.35	2.7			94.44	2.9
12			97.81	2.9			96.03	3.4
13			98.54	3.1			96.03	3.7
14			100	3.3			96.03	3.9
20							98.41	5.3
25							99.20	6.6
29							100	7.6

4.2 Triangle spiral results

For the method of feature extraction using triangle spiral too, we found the suitable values of the parameters to be used during indexing. The results so obtained for FVC 2002 DB2 have been compiled in Tables 11, 12, 13, 14, 15, 16. The values that have been finally used are shown in bold.

The optimal value of the leg of the first right triangle d_0 is 10. Here, the limiting threshold above which the cluster divides into two is 70. The parameters used during retrieval viz., D and t should be fixed at 17 and 2, respectively. The order of the tree, m , at which the best results are obtained is five. We thus construct 5-ary trees for each cluster. During retrieval, the number of matches that are considered for matching in the three nearest clusters are denoted by NN I, NN II and NN III whose optimum values are 21, 14 and 7 as can be seen from the Table. The values of these parameters are present in Table 17 along with those obtained for the rest six databases.

Using the above values, the hit rate and penetration rate at different score and rank have been found out and the results of the same have been tabulated in Tables 18, 19, 20 and 21, and depicted in Figs. 10, 11, 12 and 13.

The proposed methodology has been compared to various other indexing techniques over FVC 2002 and 2004 databases and the results of the approach which outperforms others have been presented in bold. As can be concluded from Table 22, our technique outperforms most of the proposed

Table 9 Hit rate and penetration rate at various matching score values for FVC 2004 databases

Score	DB1		DB2		DB4	
	HR	PR	HR	PR	HR	PR
0	100	26.8	100	25.6	100	41.7
10	100	16.5	100	16.6	100	16.1
20	100	15.7	100	16.0	100	14.8
30	100	14.1	100	14.5	100	13.7
40	100	11.6	100	12.5	100	11.9
50	100	9.0	100	10.5	100	9.4
60	100	6.9	100	8.9	100	8.6
70	100	5.0	100	7.4	100	6.3
80	95.93	3.6	98.45	6.1	95	5
90	94.31	2.5	96.12	4.9	91.25	4.1

Table 10 Hit rate and penetration rate at various ranks for FVC 2004 databases

Rank	DB1		DB2		DB4	
	HR	PR	HR	PR	HR	PR
1	60.97	0.3	41.86	0.2	50.00	0.4
2	75.61	0.5	57.36	0.5	58.75	0.8
3	82.92	0.8	67.44	0.8	70.00	1.2
4	86.99	1.1	73.64	1.0	76.25	1.7
5	86.99	1.3	77.51	1.3	80.00	2.1
6	89.43	1.6	81.39	1.5	85.00	2.5
7	90.24	1.8	82.94	1.8	87.50	2.9
8	91.87	2.1	85.27	2.1	87.50	3.3
9	91.87	2.4	85.27	2.3	91.25	3.8
10	91.87	2.7	87.59	2.6	93.75	4.2
15	95.93	4.0	91.47	3.8	98.75	6.2
16	96.75	4.3	93.79	4.1	100	6.7
20	99.18	5.4	96.12	5.1		
24	100	6.4	97.67	6.2		
25			97.67	6.4		
28			100	7.2		

methods in the literature. We have reached a penetration rate lower than all the techniques except the one proposed by Jayaraman et al. [17] which achieves a lower penetration rate at the same hit rate. However, since they are not able to achieve a 100 % hit rate, we can conclude that our proposed approach can perform better than the existing state-of-the-art fingerprint indexing techniques.

5 Conclusions

In this paper, we have proposed two robust techniques for extracting features to be used for fingerprint indexing. The

Fig. 6 Miss rate and penetration rate vs matching score for FVC 2002 databases

Fig. 7 Miss rate and penetration rate vs rank for FVC 2002 databases

Fig. 8 Miss rate and penetration rate vs matching score for FVC 2004 databases

Fig. 9 Miss rate and penetration rate vs rank for FVC 2004 databases

Table 11 Hit rate and penetration rate for different values of T for FVC 2002 DB2

HR	PR	T
95.65	3.2	50
100	3.8	70
100	4.8	100
99.27	5.1	150
97.82	5.1	200
97.10	5.3	300
95.65	5.4	500
94.92	5.3	1000

Table 12 Hit rate and penetration rate for different values of D for FVC 2002 DB2

HR	PR	D
98.50	3.7	15
98.50	3.8	16
100	3.8	17
99.27	3.9	18
99.27	4.0	20

Table 13 Hit rate and penetration rate for different values of m for FVC 2002 DB2

HR	PR	m
98.55	3.7	3
97.82	3.8	4
100	3.8	5
99.27	3.9	6
98.55	3.9	7
98.55	4.0	10

first approach uses information of the neighboring minutiae lying inside the L rectangles to construct the feature vector while the second one involves construction of triangle spiral for each fingerprint. Our technique thus has the added advantage of lower complexity as opposed to the ones considering

Table 14 Hit rate and penetration rate for different values of t for FVC 2002 DB2

HR	PR	t
99.27	3.8	1
100	3.8	2
100	3.9	3
98.50	3.9	5
98.50	3.9	7
98.50	3.9	10
98.50	3.9	15

Table 15 Hit rate and penetration rate for different values of d_0 for FVC 2002 DB2

HR	PR	d_0
97.10	3.7	5
100	3.8	10
94.92	3.9	15
94.92	3.9	20
97.10	4.0	30

Table 16 Hit rate and penetration rate for different values of N.N. I, N.N. II and N.N. III for FVC 2002 DB2

HR	PR	NN I	NN II	NN III
64.49	0.6	70	–	–
97.82	6.4	70	70	–
100	12.0	70	70	70
100	11.1	70	70	20
100	9.3	70	70	10
100	9.0	70	30	10
100	9.0	50	30	10
100	7.9	50	20	10
100	7.4	30	20	10
100	6.8	25	20	10
100	5.7	25	15	10
100	4.8	25	15	7
99.27	4.4	25	15	6
100	4.5	25	14	7
99.27	4.2	25	13	7
100	4.0	22	14	7
100	3.8	21	14	7
99.27	3.6	20	14	7

Table 17 Values of the user defined parameters used in the experiments

	HR	PR	s	T	NN I	NN II	NN III	D	t	m	d_0
2002 DB1	100	5.9	60	100	23	7	7	20	5	5	10
2002 DB2	100	3.8	70	70	21	14	7	17	2	5	10
2002 DB3	100	4.5	70	200	19	3	4	14	2	5	10
2002 DB4	100	5.3	70	100	25	11	2	19	3	5	10
2004 DB1	100	7.2	60	100	23	14	3	16	3	5	10
2004 DB2	100	8.0	60	100	25	12	6	19	4	5	10
2004 DB4	100	7.8	70	50	22	15	–	18	–	5	10

Table 18 Hit rate and penetration rate at various matching score values for FVC 2002 databases

Score	DB1		DB2		DB3		DB4	
	HR	PR	HR	PR	HR	PR	HR	PR
0	100	16.6	100	16.1	100	16.6	100	16.6
10	100	16.4	100	16.0	100	16.5	100	16.2
20	100	15.4	100	15.5	100	16.0	100	15.3
30	100	13.7	100	14.0	100	12.5	100	13.7
40	100	11.0	100	11.5	100	12.5	100	11.6
50	100	8.1	100	8.5	100	9.5	100	9.3
60	100	5.9	100	5.9	100	6.9	100	7.1
70	97.87	4.0	100	3.8	100	4.5	100	5.3
80	95.35	2.4	94.20	2.4	94.40	2.7	95.65	3.6
90	92.25	1.6	89.86	1.4	87.50	1.6	93.90	2.4

Table 19 Hit rate and penetration rate at various ranks for FVC 2002 databases

Rank	DB1		DB2		DB3		DB4	
	HR	PR	HR	PR	HR	PR	HR	PR
1	72.10	0.2	65.22	0.2	66.90	0.3	55.65	0.1
2	79.39	0.5	76.08	0.5	81.9	0.8	68.7	0.4
3	81.13	0.8	84.05	0.8	88.90	1.5	80.87	0.8
4	88.69	1.2	87.68	1.1	91.70	2.2	88.70	1.2
5	92.25	1.6	89.13	1.3	97.20	3.1	88.70	1.6
6	94.33	2.0	92.03	1.9	98.60	4.1	91.30	2.1
7	96.12	2.5	95.65	2.3	98.60	5.1	93.90	2.7
8	97.87	3.2	98.55	2.8	98.60	6.2	93.90	3.2
9	98.55	3.9	98.55	3.3	100	7.4	95.60	3.8
10	99.20	4.4	100	3.9	100	8.8	97.40	4.6
11	99.20	5.1	100	4.5			99.10	5.3
12	99.20	5.9	100	5.2			100	6.1
13	99.20	6.6					100	6.8
14	100	7.5					100	7.6

Table 20 Hit rate and penetration rate at various matching score values for FVC 2004 databases

Score	DB1		DB2		DB4	
	HR	PR	HR	PR	HR	PR
0	100	16.6	100	16.6	100	16.6
10	100	16.6	100	16.6	100	16.5
20	100	16.3	100	16.2	100	15.9
30	100	15.2	100	15.1	100	15.0
40	100	13.1	100	13.1	100	13.5
50	100	10.2	100	10.4	100	11.5
60	100	7.2	100	8.0	100	9.4
70	91.90	4.7	96.12	5.5	100	7.8
80	87.10	2.8	87.60	3.2	93.75	5.7
90	75.80	1.7	75.20	1.8	88.75	4.1
100	70.96	0.9	66.67	0.9	80.0	2.6

Table 21 Hit rate and penetration rate at various ranks for FVC 2004 databases

Rank	DB1		DB2		DB4	
	HR	PR	HR	PR	HR	PR
1	49.20	0.2	37.98	0.1	32.5	0.1
2	64.51	0.6	52.71	0.3	50	0.4
3	70.16	0.9	58.46	0.9	60	1.4
4	72.58	1.3	66.15	1.1	68.75	1.7
5	77.42	1.7	72.86	1.2	73.75	1.8
6	83.80	2.1	78.46	2.1	82.50	3.1
7	87.09	2.6	80.76	2.6	88.75	3.3
8	88.7	3.1	83.84	3.2	90	4.3
9	90.30	3.8	86.92	3.8	90	5.0
10	91.13	4.4	91.47	4.0	95	5.4
11	95.96	5.0	92.30	5.1	96.25	6.2
12	95.96	5.8	92.30	5.8	100	7.1
13	96.70	6.5	93.84	6.6	100	7.7
14	98.37	7.3	96.15	7.3		
15	99.20	8.1	97.67	8.0		
16			97.67	8.8		
17			99.20	9.6		
18			100	10.4		

Fig. 10 Miss rate and penetration rate vs matching score for FVC 2002 databases

all possible minutiae triplets and quadruplets while extraction of features. Furthermore, dynamic clustering helps to automatically divide the data into an optimal number of clusters unlike k-means or other clustering algorithms where the number of clusters has to be provided beforehand. The dynamic nature of the biometric database has also been taken into consideration in our proposed approach as insertion and deletion is only to be carried out in the respective clusters without disturbing the remaining ones. The results also con-

Fig. 11 Miss rate and penetration rate vs rank for FVC 2002 databases **Fig. 12** Miss rate and penetration rate vs matching score for FVC 2004 databases **Fig. 13** Miss rate and penetration rate vs rank for FVC 2004 databases

Table 22 Comparison with other indexing techniques

	2002 DB1		2002 DB2		2002 DB3		2002 DB4		2004 DB1		2004 DB2		2004 DB4	
	HR	PR	HR	PR	HR	PR	HR	PR	HR	PR	HR	PR	HR	PR
Ross et al. [14]	100	47.3	100	47	100	50.2	100	45.3	100	51.4	100	52.0	100	51.4
Vij et al. [22]	100	57	-	-	-	-	-	-	-	-	-	-	-	-
Iloanusi et al. [20]	100	11.8	-	-	-	-	-	-	100	12	-	-	-	-
Indrawan et al. [24]	99	5	-	-	-	-	-	-	-	-	-	-	-	-
Iqbal et al. [25]	-	-	99	26	-	-	-	-	-	-	-	-	-	-
Jayaraman et al. [17]	-	-	-	-	-	-	-	-	96	2.61	95	2.3	-	-
Zhou et al. [26]	100	3.51	-	-	-	-	-	-	-	-	-	-	-	-
Guoqiang et al. [27]	-	-	-	-	-	-	-	-	95	8.9	95	9.75	-	-
Triangle Spiral	100	5.9	100	3.8	100	4.5	100	5.3	100	7.2	100	8.0	100	7.8
Neighboring Relation	100	2.1	100	3.1	100	3.1	100	6.9	100	5.0	100	7.4	100	6.3

firm that our proposed methodology works better than the existing ones. The method of feature extraction using multi-line neighboring relation achieves a hit rate and penetration rate of 100 % and 2.1 %, 100 % and 3.1 %, 100 % and 3.1 %, and 100 % and 6.9 % for FVC 2002 DB1, DB2, DB3 and DB4, respectively. For FVC 2004 databases, the two rates achieved are 100 % and 5.0 %, 100 % and 7.4 %, and 100 % and 6.3 % for DB1, DB2 and DB4 respectively. On the other hand, if feature extraction is done by the construction of triangle spiral for each fingerprint, the results obtained are 100 % and 5.9 %, 100 % and 3.8 %, 100 % and 4.5 %, and 100 % and 5.3 % for FVC 2002 DB1, DB2, DB3 and DB4, respectively. The hit rate and penetration rate are 100 % and 7.2 %, 100 % and 8.0 %, and 100 % and 7.8 % for FVC 2004 DB1, DB2 and DB4 respectively. Further, the dynamic clustering used in this paper narrows down the search space of a fingerprint database.

References

1. Biometrics-related activities of the Federal government. <http://biometrics.gov>. Accessed 25 July 2015
2. Jain AK, Maltoni D (2003) Handbook of fingerprint recognition. Springer, Berlin
3. Aadhaar. Unique Identification Authority of India. <http://uidai.gov.in>. Accessed 25 July 2015
4. Henry (2003) Henry classification system international biometric group. <http://www.biometricgroup.com/HenryFingerprintClassification.pdf>. Accessed 25 July 2015
5. Dey S, Samanta D (2014) Unimodal And Multimodal Biometric Data Indexing. Walter de Gruyter, Hawthorne
6. Liu T, Zhu G, Zhang C, Hao P (2005) Fingerprint indexing based on singular points. International Conference on Image Processing, pp 293–296
7. Lee SO, Kim YG, Park GT (2005) A feature map consisting of orientation and inter ridge spacing for fingerprint retrieval. International Conference on Audio and Video Based Biometric Person Authentication, pp 184–190
8. Maeda, Matsushita M, Sasakawa K (2004) Characteristics of the identification algorithm using a matching score matrix. In: First International Conference on Biometric Authentication, pp 330–336
9. Shuai X, Zhang C, Hao P (2008) Fingerprint indexing based on composite set of reduced SIFT features. In: International Conference on Pattern Recognition, pp. 1–4
10. Germain RS, Califano A, Colville S (1997) Fingerprint matching using transformation parameter clustering. IEEE Comput Sci Eng 4(4):42–49
11. Bhanu B, Tan X (2003) Fingerprint indexing based on novel features of minutiae triplets. IEEE Trans Pattern Anal Mach Intell 25(5):616–622
12. Bebis G, Deaconu T, Georgiopoulos M (1999) Fingerprint identification using delaunay triangulation. In: International Conference on Information Intelligence and Systems, pp 452–459
13. Liang X, Bishnu A, Asano T (2007) A robust fingerprint indexing scheme using minutia neighborhood structure and low-order delaunay triangles. IEEE Trans Inf Forensics Secur 2(4):721–733
14. Ross A, Mukherjee R (2007) Augmenting ridge curves with minutiae triplets for fingerprint indexing. Proceedings of SPIE

15. Cappelli R, Ferrara M, Maltoni D (2010) Minutia cylinder-code: a new representation and matching technique for fingerprint recognition. *IEEE Trans Pattern Anal Mach Intell* 32(12):2128–2141
16. Wang Y, Wang L, Yuen P, Cheung YM (2015) Learning compact binary codes for hash-based fingerprint hashing. *IEEE Trans Inf Forensics Secur* 10(8):1603–1616
17. Jayaraman U, Gupta AK, Gupta P (2014) An efficient minutiae based geometric hashing for fingerprint database. *Neurocomputing* 137:15–126
18. Brisen A, Alonso A, Palancar J (2014) Using reference point as feature for fingerprint indexing. *Progress in pattern recognition, image analysis, computer vision, and applications. Lect Notes Comput Sci* 8827:367–374
19. Moujahdi C, Bebis G, Ghouzali S, Rziza M (2014) Fingerprint shell: secure representation of fingerprint template. *Pattern Recognit Lett* 45:189–196
20. Iloanusi O, Gyaourova A, Ross A (2011) Indexing fingerprints using minutiae quadruplets. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp 127–133
21. Linde Y, Buzo A, Gray RM (1980) An algorithm for vector quantizer design. *IEEE Trans Commun* 28(1):84–95
22. Vij A, Nambodiri A (2012) Fingerprint indexing based on local arrangements of minutiae neighborhoods. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp 71–76
23. Neurotechnology Verifinger SDK. <http://www.neurotechnology.com/verifinger.html>. Accessed 25 July 2015
24. Indrawan G, Akbar S, Sitohang B (2014) Fingerprint direct-access strategy using local-star-structure-based discriminator features: a comparison study. *Int J Electr Comput Eng (IJECE)* 4(5):817–830
25. Iqbal A, Nambodiri AM (2011) Cascaded filtering for biometric identification using random projections. In: *National Conference on Communications (NCC)*, pp 1–5
26. Zhou W, Hu J, Wang S, Petersen I, Bennamoun M (2014) Fingerprint indexing based on combination of novel minutiae triplet features. *Netw Syst Secur, Lect Notes Comput Sci* 8792:377–388
27. Guoqiang L, Bian Y, Busch C (2014) A score-level fusion fingerprint indexing approach based on minutiae vicinity and minutia cylinder-code. In: *International Workshop on Biometrics and Forensics (IWBF)*, pp. 1–6
28. Prasad MVNK, Kumar CS (2014) Fingerprint template protection using multiline neighboring relation. *Expert Syst Appl* 41:6114–6122. <http://www.sciencedirect.com/science/article/pii/S0957417414002176>
29. Yijing Su, Feng Jianjiang, Zhou Jie CS (2016) Fingerprint indexing with pose constraint. *Pattern Recognit* 54:1–13
30. Li G, Busch C (2015) A fingerprint indexing scheme with robustness against sample translation and rotation. In: *International Conference of the Biometrics Special Interest Group (BIOSIG)*, pp 1–8
31. Tiwari K, Gupta P (2015) Indexing fingerprint database with minutiae based coaxial Gaussian track code and quantized lookup table. In: *International Conference on Image Processing (ICIP)*, pp 4773–4777