

## Chapter 2

# Hierarchical Decomposition of Extended Triangulation for Fingerprint Indexing

**Abstract** In biometric identification systems, the identity corresponding to the query image is determined by comparing it against all images in the database. This exhaustive matching process increases the response time and the number of false positives of the system. This chapter presents an efficient indexing algorithm for fingerprint databases to improve the search speed and accuracy of identification. A variant of Delaunay triangulation called extended triangulation is used to make the system robust against distortions. Then the triangles are partitioned into groups such that the retrieval algorithm searches in reduced space of the database. Experiments are conducted on different fingerprint databases, and the results show that while maintaining high hit rate the proposed method achieves lower penetration rate than what existing methods achieve.

**Keywords** Fingerprint · Delaunay triangulation · Extended triangulation · Hierarchical decomposition

## 2.1 Introduction

Nowadays, the increasing use of biometrics for personal identification in various applications has lead to increase of some large-scale biometric databases in real time [1–9]. However, identifying a user on such huge databases using a linear matching process makes the system extremely slow. Boro et al. [10] extracted the minutiae points of the fingerprints and mapped them into a hash table using geometric hashing [11]. Similarly, Hunny et al. [12] extracted the key features of iris using scale invariant feature transform [13, 14] and mapped them with the help of geometric hashing. However, the above methods require high computational and memory costs as each feature is inserted multiple times into the hash table to handle the intra-class natural variations. The triplets have been successfully used to index biometric databases [15, 16]. The triplet-based techniques have proven more powerful than point-based techniques, as the uncertainty of feature points and intra-class natural variations do not affect the angles of a triangle. In this chapter, we propose an efficient triangulation-based approach for indexing fingerprint databases. This work enhances the Delaunay triangulation to make the system robust against biometric distortions.

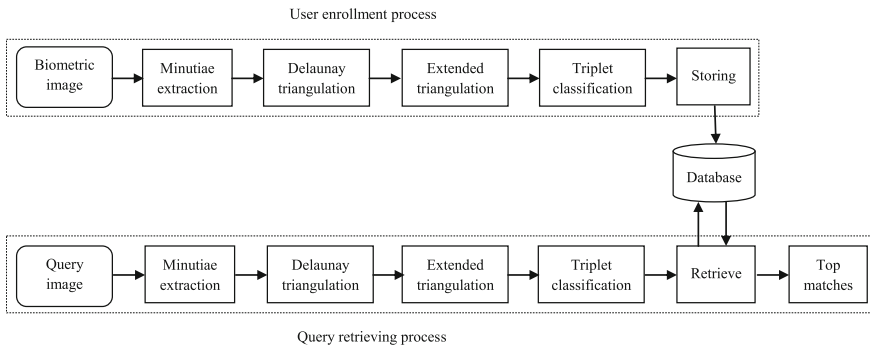
## 2.2 Indexing Framework

The proposed indexing approach follows these steps: extraction of minutiae features from fingerprint images. Then, for each fingerprint, its Delaunay triangulation is computed using the extracted minutiae features. Next, a robust representation is defined for the fingerprints known as extended triangulation which is an enhanced model of Delaunay triangulation [17]. Finally, the computed extended triangulation is classified such that the retrieval algorithm searches in reduced space of the database. The overview of the indexing framework is shown in Fig. 2.1. Various steps involved in the indexing process are discussed in the following:

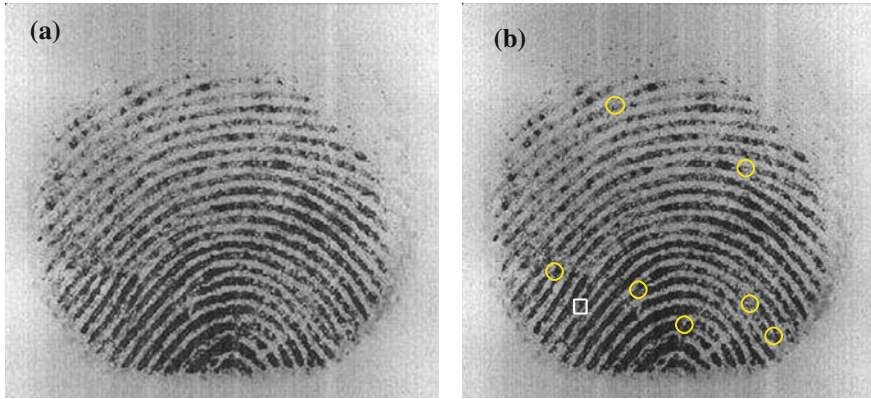
### 2.2.1 Minutiae Extraction

In this work, the minutiae are considered as the key features for the fingerprints. The minutiae points considered are (i) bifurcation points and (ii) end points. A bifurcation point is a point where the ridge forks or diverges into branch ridges. An end point is a point where a ridge ends abruptly. To extract the minutiae features from the fingerprint images, we used the Nuerotechnology VeriFinger SDK [18]. A sample fingerprint and its extracted minutiae are shown in Fig. 2.2 (bifurcation points are represented with circle symbol and end points are represented with square symbol).

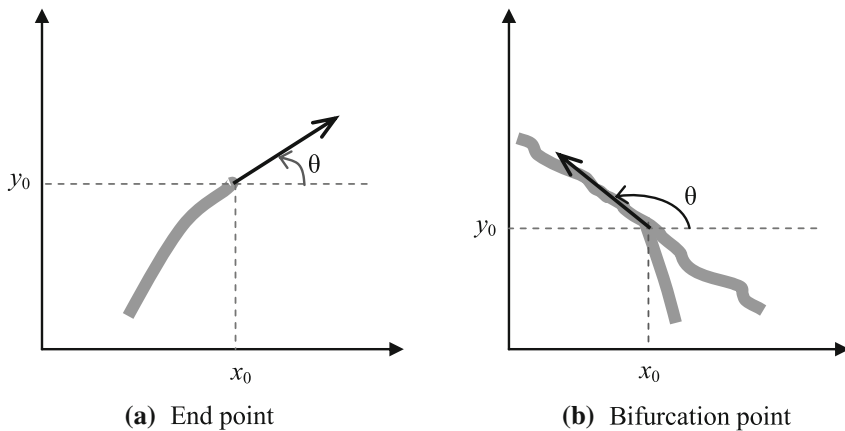
Let  $p = (x, y, t, \theta)$  be a minutiae feature point of a fingerprint, where  $(x, y)$  is its position in the fingerprint,  $t$  is its type (fork or end), and  $\theta$  is its orientation or angle with respect to  $x$ -axis. The process of computing the minutiae orientation is shown in Fig. 2.3. The orientation of a minutiae point (either bifurcation or end) is measured by calculating the angle between the tangent to the ridge line at the minutiae position and the  $x$ -axis.



**Fig. 2.1** Overview of the proposed approach



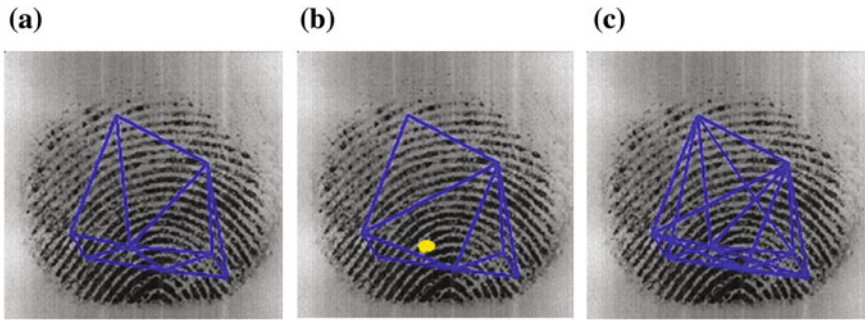
**Fig. 2.2** **a** A sample fingerprint **b** Retrieved minutiae (*circle*- bifurcation points; *square*- end points)



**Fig. 2.3** Minutiae position and its orientation extraction process

### 2.2.2 Computation of Delaunay Triangulation

Once the minutiae features have been extracted, their Delaunay triangulation is computed. Let  $P = \{p_1, p_2, \dots, p_n\}$  be the set of extracted minutiae points from a fingerprint. Then the Delaunay triangulation  $T$  of minutiae set  $P$  is a maximal planar subdivision in which no edge connecting two vertices can be included to it without destroying its planarity. Delaunay triangle contains no other point of  $P$  in its circum-circle. Figure 2.4 shows the Delaunay triangulation of the minutiae for one of the fingerprints. The motivation of using Delaunay triangulation in this work is that the Delaunay triplets possess certain unique properties compared to other topological structures [19–21], including the following:



**Fig. 2.4** Triangulation for a sample fingerprint [22]: **a** Delaunay triangulation; **b** Structure of the Delaunay triangulation after missing a minutiae point (missing minutiae is shown with star); **c** Extended triangulation

1. Delaunay triangulation partitions a whole region into many smaller pieces and exactly describes the closest neighbor structures of minutiae.
2. Insertion of a new point in a Delaunay triangulation affects only the triangles whose circum circles contain that point. As a result, noise affects the Delaunay triangulation only locally [20];
3. The Delaunay triplets are not skinny which is desirable as the skinny triangles lead to instabilities and errors.
4. The Delaunay triangulation creates only  $O(n)$  triangles when compared to the approaches in Germain et al. [15] and Bhanu et al. [16] which uses all possible triangles of minutiae set in the fingerprint, and therefore  $O(n^3)$  triangles have to be compared during indexing. Hence, computing cost greatly decreases using Delaunay triangulation.
5. Compared to other topological structures, the Delaunay triangulation is less sensitive to distortion.

However, the noise seriously affects the Delaunay triangulation structure which is a common problem in image processing. Figure 2.4a, b shows one such example with spurious minutiae for two fingerprints of the same user. So, to minimize this, the Delaunay triangulation is enhanced to form a new structure called extended triangulation [17].

### 2.2.3 Retrieval of Extended Triplet Set

Let  $T$  be the Delaunay triangulation of a fingerprint with minutiae set  $P = \{p_1, p_2, \dots, p_n\}$ , and  $G = \{P, E\}$  is its Delaunay graph; where  $E$  is its edge set. Before defining the extended triangulation of  $P$ , let us first explore the triangular hull of a point  $p_i \in P$ . Let  $N_i = \{p_j | (p_i, p_j) \in E\}$  be the set of neighborhood minutiae of  $p_i$  in the Delaunay graph  $G$ . Then the triangular hull of  $p_i$  denoted by

$H_i$  is the Delaunay triangulation of  $N_i$ . Now, the extended Delaunay triangulation  $S = \{T \cup H_1 \cup H_2 \cup \dots \cup H_n\}$  [17]. The extended triangulation for a fingerprint is shown in Fig. 2.4c. It can be seen that the extended triangulation is more robust against distortions. For example, Fig. 2.4a shows the Delaunay triangulation of a fingerprint. Figure 2.4b shows the changes in structure of the triangulation due to missing of a minutiae point. This results in false rejection. However, the extended triangulation (Fig. 2.4c) contains the triangles of both Fig. 2.4a, b. In other words Fig. 2.4c shows  $T$  and the triangular hull  $H_x$  that is formed when feature  $x$  is missed, i.e.,  $T \cup H_x$ . From this example, we can say that, even when the system fails to extract feature  $x$  (i.e., missing minutiae) at the time of identification, the corresponding triangles can be found in the extended set and increases the accuracy. Note that this is not possible with Delaunay triangulation. Further, this is also true for fake minutiae [22].

The extended triangulation contains more triangles than Delaunay triangulation, i.e.,  $|S| \geq |T|$  (Fig. 2.4). But  $|S| \in O(n)$  like Delaunay triangulation [17]. The following theorem proves this.

**Theorem 1** *The number of triangles in  $S$  is  $O(n)$  [17].*

*Proof* The number of triangles in  $S$  can be given as follows:

$$|S| \leq |T| + \sum_{i=1}^n |H_i|. \quad (2.1)$$

The number of triangles in a Delaunay triangulation  $T$  of  $n$  points can be bounded by  $2n - 1$ , i.e.,  $|T| = 2n - 1$  [17, 19]. This is also true with the case of each triangular hull  $H_i$ . So,  $|H_i| = 2d_i - 1$ , where  $d_i$  is degree of the  $p_i$ . Hence, Eq. 2.1 can be transformed as follows:

$$\begin{aligned} |S| &\leq (2n - 1) + \sum_{i=1}^n (2d_i - 1), \\ &\leq 2n - 1 + 2 \sum_{i=1}^n d_i - n, \\ &\leq n - 1 + 2 \sum_{i=1}^n d_i. \end{aligned} \quad (2.2)$$

According to handshaking lemma of graph theory, the sum of the degrees of all the vertices of a graph is equal to twice the number of edges, i.e.,  $\sum_{i=1}^n d_i = 2|E|$  [23]. So, Eq. 2.2 can be transformed as follows:

$$|S| \leq n - 1 + 4|E|. \quad (2.3)$$

Further, according to Euler's formula, the number of edges ( $E$ ) for a planar graph with vertices ( $n$ )  $\geq 3$  is less than or equal to  $3n - 6$ , i.e.,  $|E| \leq 3n - 6$  [23]. Thus, Eq. 2.3 can be transformed to Eq. 2.4:

$$\begin{aligned} |S| &\leq n - 1 + 4(3n - 6) \\ &\leq 13n - 25. \end{aligned} \quad (2.4)$$

From Eq. 2.4, we have  $|S| \leq 13n - 25$ , proving that  $|S| \in O(n)$ .  $\square$

### 2.2.4 Hierarchical Decomposition of Extended Set

In the next step, the extended triangles of the fingerprint are classified based on the combination of type of minutiae at the vertices of each triangle. Figure 2.5 shows an example of an extended triangle. Let  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  are the minimal, medial, and maximal angles in the triangle, respectively. The vertices of the triangle are labeled as  $V_1$ ,  $V_2$ , and  $V_3$  corresponding to the angles  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$ . For example, the vertex with  $\alpha_1$  is labeled as  $V_1$ . The remaining vertices are labeled accordingly. Then, based on the combination of types of minutiae at the vertices  $V_1$ ,  $V_2$ , and  $V_3$  of the triangle, it is classified into one of eight types as depicted in Table 2.1.

### 2.2.5 Enrollment

This section explains the process of enrolling (or storing) a fingerprint into an index table. Note that the fingerprint is represented with an extended triangle set  $S$ . For each triangle in  $S$ , an index  $X$  and a feature vector  $f$  are computed as shown in Eq. 2.5,

**Table 2.1** Hierarchical decomposition of extended triangles

Triangle class	Minutiae type <sup>a</sup>		
$t_c$	$V_1$	$V_2$	$V_3$
1	$b$	$b$	$b$
2	$b$	$b$	$e$
3	$b$	$e$	$b$
4	$b$	$e$	$e$
5	$e$	$b$	$b$
6	$e$	$b$	$e$
7	$e$	$e$	$b$
8	$e$	$e$	$e$

<sup>a</sup> $b$ -bifurcation point,  $e$ -endpoint

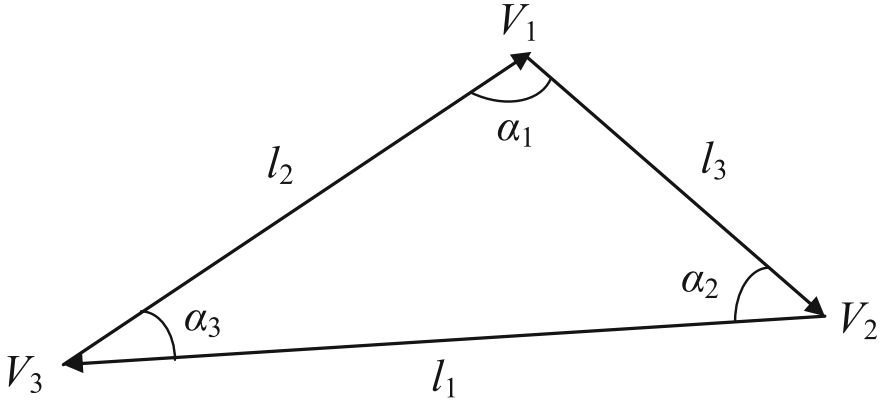


Fig. 2.5 Minutiae triangle

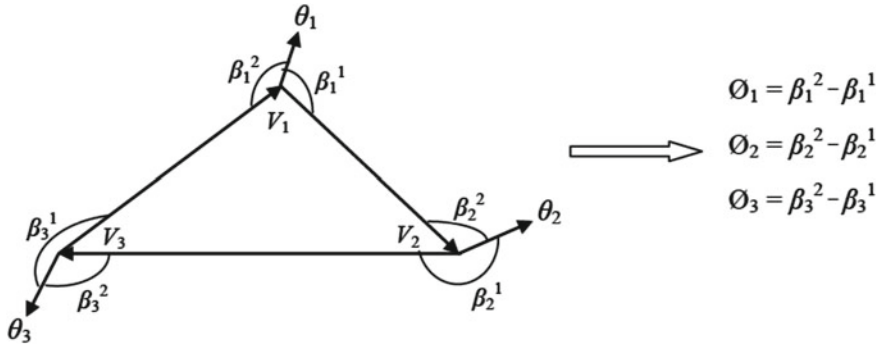


Fig. 2.6 Relative orientation of a minutiae [22]

where  $t_c$  is the triangle class,  $l_1, l_2, l_3$  are the lengths of each side of the triangle such that  $l_1 \geq l_2 \geq l_3$  and  $\phi_1, \phi_2, \phi_3$  are the relative orientations of minutiae points at vertices  $V_1, V_2$ , and  $V_3$ , respectively:

$$\begin{aligned} X &= (t_c, \alpha_1, \alpha_2) \\ f &= (l_1, l_2, l_3, \phi_1, \phi_2, \phi_3). \end{aligned} \quad (2.5)$$

The process of computing the relative orientation of a triplet minutiae at vertices  $(V_i, V_j, V_k)$  is shown in Fig. 2.6. The relative orientation  $\phi_i$  of minutiae at vertex  $V_i$  is defined in Eq. 2.6, where  $\beta_i^1$  and  $\beta_i^2$  are the angles that the orientation vector  $(\theta_i)$  of minutiae at vertex  $V_i$  makes with its incident edges, i.e.,  $V_i V_j$  and  $V_i V_k$ :

$$\phi_i = \beta_i^1 - \beta_i^2. \quad (2.6)$$

In the next step, the triplet is enrolled into a 3D index table (*ISPACE*) using its index  $X$ . The 3D index table is shown in Fig. 2.7a. The index table size is chosen as  $\max(t_c) \times \max(\alpha_1) \times \max(\alpha_2)$  bins, where  $1 \leq t_c \leq 8$ ,  $0 \leq \alpha_1 \leq 180$ ,  $0 \leq \alpha_2 \leq 180$ . Further, it can be seen that each bin has two lists:

- $I_{id}List$ —This list stores the fingerprint ids of the triplets which are mapped (i.e., indexed) to this particular bin;
- $FVList$ —This list stores the feature vectors of the mapped triplets.

Enrolling of a triangle into the *ISPACE* is shown in Eq. 2.7, where  $X$  is the index space location (i.e., bin) where the triplet to enroll;  $I_{id}$  represents the image identity to which the triplet belongs; and  $f$  is the feature vector of the triplet [22]. Note that the first dimension (say triplet class) partitions the 3D index space into eight classes (2D tables). This is shown in Fig. 2.7a:

$$\begin{aligned} ISPACE[X].I_{id}List &\leftarrow I_{id} \\ ISPACE[X].FVList &\leftarrow f. \end{aligned} \quad (2.7)$$

The process of enrolling a triplet into the *ISPACE* is illustrated with an example: Let  $X = (4, 50, 65)$  be the index of one of the triplets of an image  $x$  and  $f$  be its feature vector. Using  $X$ , the indexing algorithm access the (4,50,65)th bin of the *ISPACE* and places  $x$  and  $f$  at the  $I_{id}List$  and  $FVList$  of it respectively. In other words, the algorithm maps to the (50, 65)th location in the 4th partition of the *ISPACE* and store the triplet's feature vector  $f$  and its image identity  $x$  in the lists provided (Fig. 2.7b).

The remaining triangles in extended set are also enrolled into the *ISPACE* likewise. We repeat this process for other fingerprints in the database. Finally, note that more than one triangle may map to the same bin of *ISPACE* because different triangles may have same index. In other words, some bins of the *ISPACE* may receive multiple triangles. Hence, the insertion of image identity  $I_{id}$  along with the feature vector  $f$  into the *ISPACE* helps to eliminate the false matches. The indexing mechanism is given in Algorithm 2.1.

---

**Algorithm 2.1** Indexing: Fingerprint enrollment into the index space

---

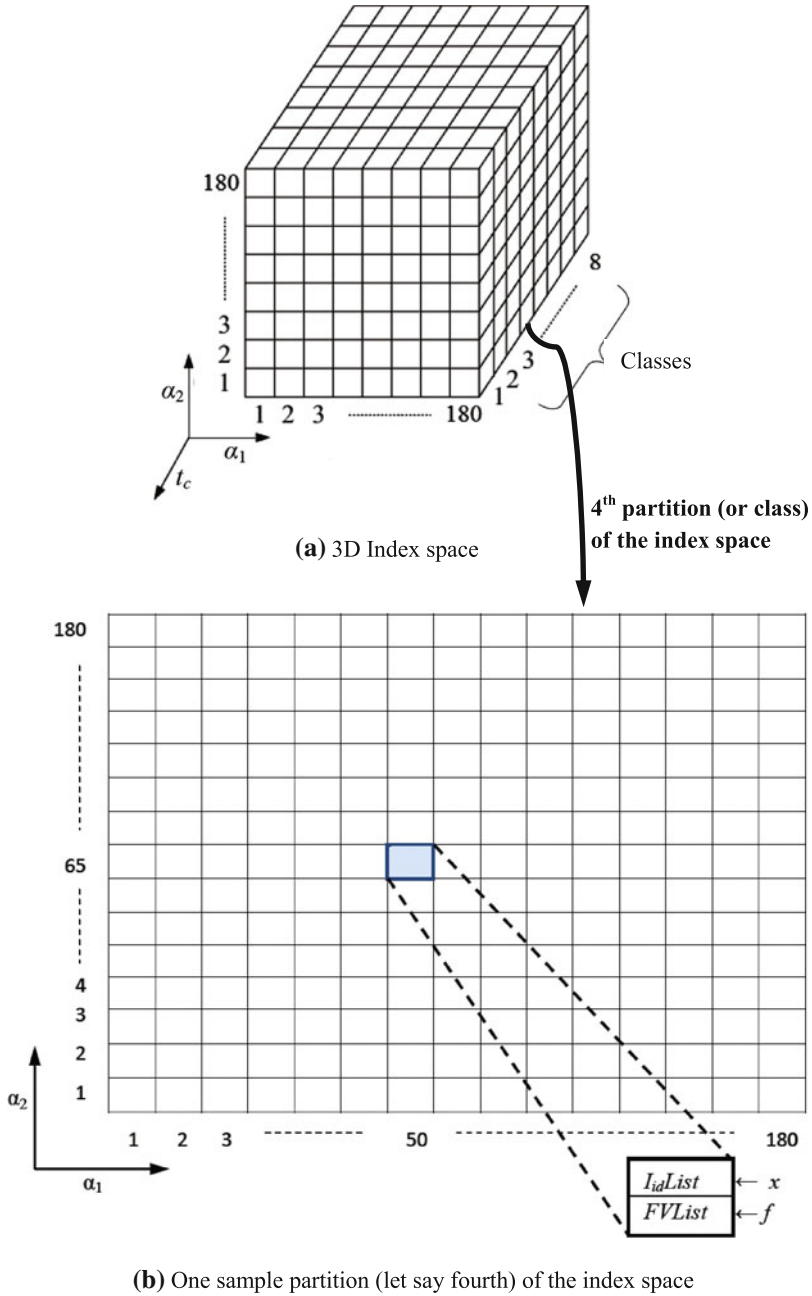
```

1: INPUT:  $x$ : Input fingerprint,  $S$ : Extended triplet set of  $x$ , and ISPACE: Index space.
2: OUTPUT:  $A$ : Updated ISPACE.
3: for each extended triplet of fingerprint  $x$  do
4:    $t_c \leftarrow i$ , where  $1 \leq i \leq 8$ .                                // Compute the triplet's class  $t_c$ 
5:    $f \leftarrow (l_1, l_2, l_3, \phi_1, \phi_2, \phi_3)$ .                  // Compute triplet's feature vector  $f$ 
6:    $X \leftarrow (t_c, \alpha_1, \alpha_2)$ .                             //  $X$  is triplet's index in ISPACE
7:    $ISPACE[X].I_{id}List \leftarrow x$ .
8:    $ISPACE[X].FVList \leftarrow f$ 
9: end for
10: RETURN Updated ISPACE

```

---





**Fig. 2.7** **a** Proposed 3D Index space (*ISPAC*E) structure, **b** Process of enrolling a triplet into the *ISPAC*E: A triplet with index (4, 50, 65) is stored into the (50, 65)th location (shown with color) of the 4th partition in the *ISPAC*E, where  $f$  is the feature vector of the triplet and  $x$  is its image identity

### 2.3 Query Identification

Query identification is the process of retrieving a small set of candidates  $C$  from the  $ISPACE$  which are most similar to it. To do this, first, the extended triangle set of the query is computed as discussed in Sect. 2.2.3. Then, the triangles in the retrieved extended set are classified as discussed in Sect. 2.2.4. Let a query fingerprint consist of  $n$  triplets in its extended set  $S$ . For each triplet in  $S$ , its index and feature vector are computed.

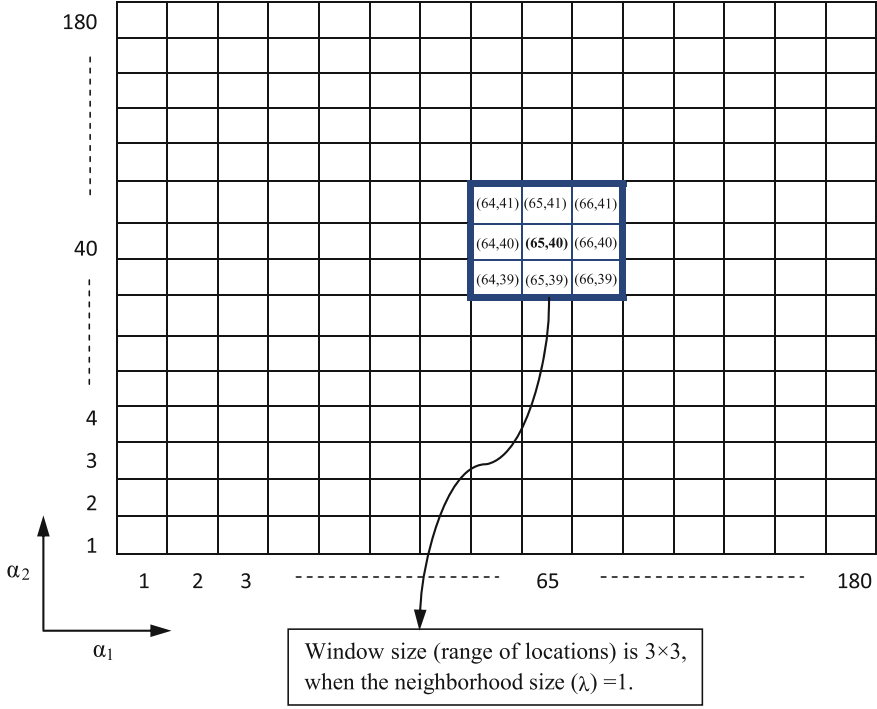
Let  $X = (t_c, \alpha_1, \alpha_2)$ ,  $f = (l_1, l_2, l_3, \phi_1, \phi_2, \phi_3)$ , and  $t_c$  be the index, feature vector, and class of a enrolled triplet, respectively. Let  $X' = (t'_c, \alpha'_1, \alpha'_2)$ ,  $f' = (l'_1, l'_2, l'_3, \phi'_1, \phi'_2, \phi'_3)$  and  $t'_c$  be the index, feature vector, and class of a enrolled triplet, respectively. The index  $X'$  of the query triplet is used to access the  $ISPACE$  and retrieve a set of fingerprints from the  $I_{id}List$  of that bin, whose triplet feature vectors satisfy the set of conditions given in Eq. 2.8 as successful correspondences to the query triplet, where  $T_l$  and  $T_\phi$  are predefined thresholds. In other words, two triangles are said to be matched, iff their feature vectors are similar. Let these retrieved image identities are stored into a temporary list  $L$ :

$$\begin{aligned} &|l_1 - l'_1|, |l_2 - l'_2|, |l_3 - l'_3| < T_l, \\ &\text{and } |\phi_1 - \phi'_1|, |\phi_2 - \phi'_2|, |\phi_3 - \phi'_3| < T_\phi. \end{aligned} \quad (2.8)$$

Note that the image acquisition and preprocessing is sensitive to noise and distortions, and the features of the two images of same user may be shifted or missed. Therefore, the retrieval systems need to consider the triplets not only from the mapped bin but also from its nearest bins. The image identities in the nearest bins (i.e., predefined neighborhood  $\lambda$ ) that satisfy the conditions given in Eq. 2.8 are retrieved and stored into  $L$ .

Illustration: Let  $X' = (6, 65, 40)$  be the index of a query triplet and  $f'$  be its feature vector. First, the retrieval algorithm maps to the (65,40)th location in the 6th partition of the  $ISPACE$ . Then, it compares the feature vector of the query triplet, i.e.,  $f'$  with each feature vector found in the  $FVList$  of the bin, and retrieves all the  $I_{ids}$  from  $I_{id}List$  whose triplet feature vectors are similar to query triplet. The retrieval algorithm also retrieves the  $I_{ids}$  from predefined  $\lambda$  which satisfy the conditions in Eq. 2.8. Let  $\lambda=1$ , i.e., window size is  $3 \times 3$  (shown in Fig. 2.8). Hence, the range of locations is from (64, 39) to (66, 41) (shown in Fig. 2.8). All these retrieved image identities ( $I_{ids}$ ) are stored into temporary list  $L$ .

Similarly, this process is repeated for each query triplet and the selected fingerprint ids  $I_{ids}$  are retrieved into temporary list  $L$ . In the next step, the number of occurrences (i.e.,  $Votes$ ) of each fingerprint identity, i.e.,  $I_{id}$  in  $L$ , is counted and forms the set as  $\{(I_{id}, Votes_{I_{id}})\}$ , where  $I_{id}$  is the image identity and  $Votes_{I_{id}}$  is the number of occurrences of  $I_{id}$  in  $L$ . Finally, the  $I_{ids}$  whose  $Votescore$  greater than a threshold ( $T$ ) are retrieved as similar fingerprints (i.e., candidate set  $C$ ) to the query. The vote



**Fig. 2.8** Range of locations considered in the *ISPACE*, to retrieve the similar triplets for a query triplet

score of an image identity represented as  $Votescore_{I_{id}}$  is defined in Eq. 2.9, where  $Votes_{I_{id}}$  is the number of corresponding matched triangles between  $q$  and  $I_{id}$ , and  $n$  is the number of query triangles. The retrieval method is given in Algorithm 2.2:

$$Votescore_{I_{id}} = \left( \frac{Votes_{I_{id}}}{n} \right) \times 100. \quad (2.9)$$

## 2.4 Experimental Results

To study the effectiveness of the proposed indexing approach, a number of experiments have been conducted on FVC fingerprint databases. This section describes the experiments carried out and the results observed.

**Algorithm 2.2** Fingerprint identification: Retrieving similar fingerprints for a query

---

```

1: INPUT:  $q$ : Query fingerprint,  $S$ : query's Extended triplet set,  $ISPACE$ : Index space,  $\lambda$ : pre-
   defined neighborhood,  $T$ : matching threshold
2: OUTPUT:  $C$ : Set of similar fingerprints.
3: for each extended triplet of  $q$  do
4:    $F \leftarrow \{\}, Im \leftarrow \{\}$ 
5:    $t_c \leftarrow i$ , where  $1 \leq i \leq 8$ .
6:    $f \leftarrow (l_1, l_2, l_3, \phi_1, \phi_2, \phi_3)$ .
7:    $X \leftarrow (t_c, \alpha_1, \alpha_2)$ . //  $X$  is triplet's index i.e., bin location in  $ISPACE$ 
   // Retrieve the identities from the mapped bin and its neighbors
8:   for  $j = \alpha_1 - \lambda$  to  $\alpha_1 + \lambda$  do
9:     for  $k = \alpha_2 - \lambda$  to  $\alpha_2 + \lambda$  do
10:       $F \leftarrow F \cup ISPACE[t_c, j, k].FVList$ 
11:       $Im \leftarrow Im \cup ISPACE[t_c, j, k].IdList$ 
12:     end for
13:   end for
   // Select the identities whose triplets are equal to query triplet
14:   for  $l=1$  to  $|F|$  do
15:      $f' \leftarrow F.l$ 
16:     if  $f' \approx f$  then
17:        $L \leftarrow L \cup Im.l$ .
18:     end if
19:   end for
20: end for
21: Select the  $I_{ids}$  in  $L$  whose  $Votescore \geq T$  as similar to  $q$  and retrieve them into  $C$ .
22: RETURN  $C$ 

```

---

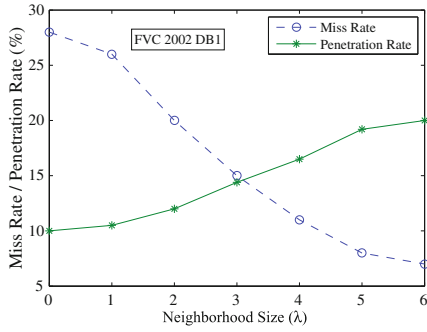
### 2.4.1 Parameter Selection

Selection of appropriate values for different parameters involved in the system is critical for achieving its best performance. One such important parameter is the selection of neighborhood size ( $\lambda$ ) (Sect. 3.3). By fixing the matching threshold  $T$ , an experiment is conducted with various  $\lambda$  sizes starting from 0 to 6, and the corresponding MR and PR are recorded for every  $\lambda$  (Table 2.2). The relationship between MR, PR, and  $\lambda$  is shown in Figs. 2.9 and 2.10 for FVC 2002 and FVC 2004 fingerprint databases, respectively. It is observed that the PR increases with the  $\lambda$  while MR decreases. But for a real-time application, both MR and PR should be low. Hence, an optimal value for the  $\lambda$  should be chosen such that the system achieves low MR as well as low PR. Therefore, the optimum  $\lambda$  value is chosen as where the two curves intersect, i.e.,  $MR = PR$ .

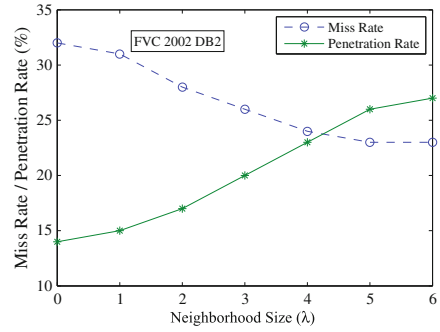
In the experiment, for FVC 2002 DB1 and FVC 2004 DB4 databases, the optimum  $\lambda$  size obtained is in and around 3 (i.e., window size is  $7 \times 7$ ). For FVC 2002 DB2, FVC 2002 DB3, FVC 2004 DB1, and FVC 2004 DB2, the optimum  $\lambda$  value is in and around 4 (i.e., window size is  $9 \times 9$ ). In case of FVC 2002 DB4, the optimum  $\lambda$  value is 5 (i.e., window size is  $11 \times 11$ ). Table 2.3 shows the optimum  $\lambda$  value obtained for different databases.

**Table 2.2** Effect of neighborhood size on the indexing performance

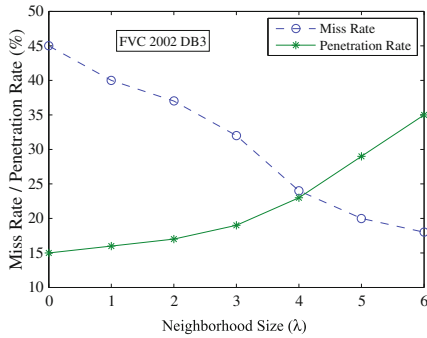
$\lambda$	2002DB1		2002DB2		2002DB3		2002DB4		2004DB1		2004DB2		2004DB4	
	MR	PR	MR	PR	MR	PR	MR	PR	MR	PR	MR	PR	MR	PR
0	28.45	10.23	32.14	14.23	45.42	15.93	45.56	9.81	42.54	12.69	32.17	8.86	26.65	9.89
1	26.12	10.54	31.05	15.23	40.52	16.26	39.42	12.65	40.15	14.32	29.47	11.26	24.42	9.51
2	20.08	12.75	28.26	17.45	37.19	17.64	34.18	15.26	35.48	16.54	24.20	12.13	17.26	10.87
3	<b>15.11</b>	<b>14.42</b>	26.11	20.92	32.15	19.05	32.45	20.32	28.65	19.09	19.87	15.04	<b>13.42</b>	<b>12.86</b>
4	11.32	16.55	<b>24.21</b>	<b>23.98</b>	<b>24.69</b>	<b>23.18</b>	28.96	24.58	<b>23.09</b>	<b>21.86</b>	<b>16.91</b>	<b>18.82</b>	10.68	15.26
5	8.26	19.36	23.65	26.09	20.54	29.04	<b>26.35</b>	<b>27.28</b>	17.08	26.40	16.08	20.15	9.86	19.04
6	5.32	20.89	23.03	27.68	18.23	35.45	25.24	28.86	12.75	30.24	15.32	21.86	9.12	21.31



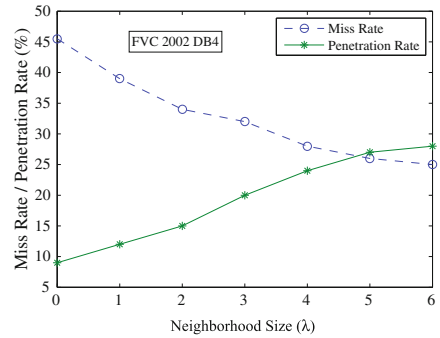
(a) FVC 2002 DB1



(b) FVC 2002 DB2



(c) FVC 2002 DB3



(d) FVC 2002 DB4

**Fig. 2.9** Effect of neighborhood size on the indexing performance for FVC 2002 databases

### 2.4.2 Results

Once the optimum  $\lambda$  value is chosen, an experiment was conducted to evaluate the performance of the proposed indexing technique for different databases. At various threshold ( $T$ ) values, we determine the MR and PR of the system. The relationship

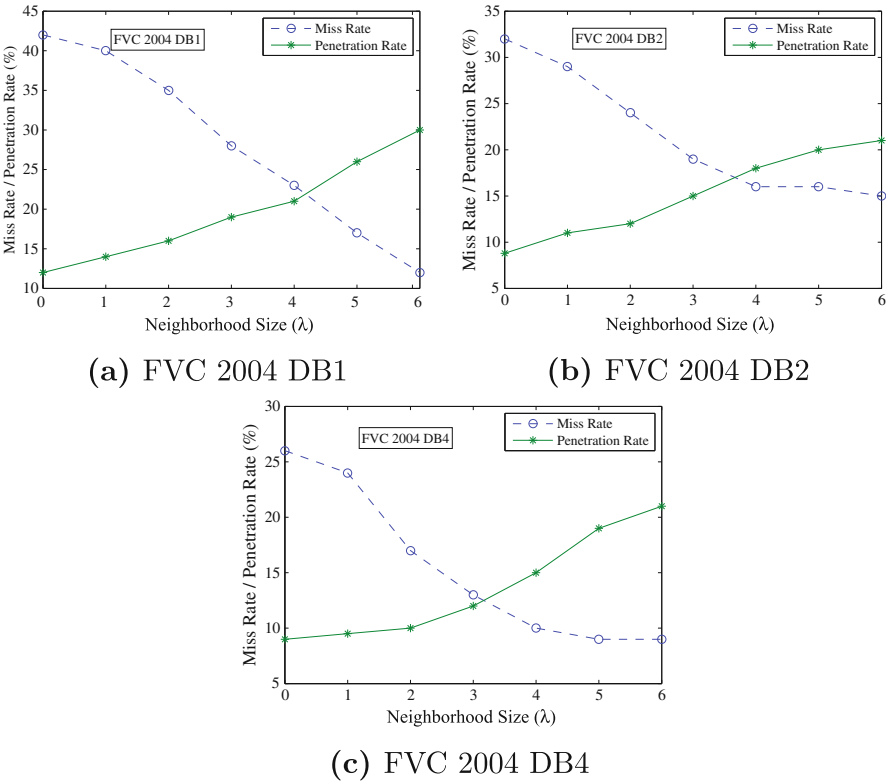
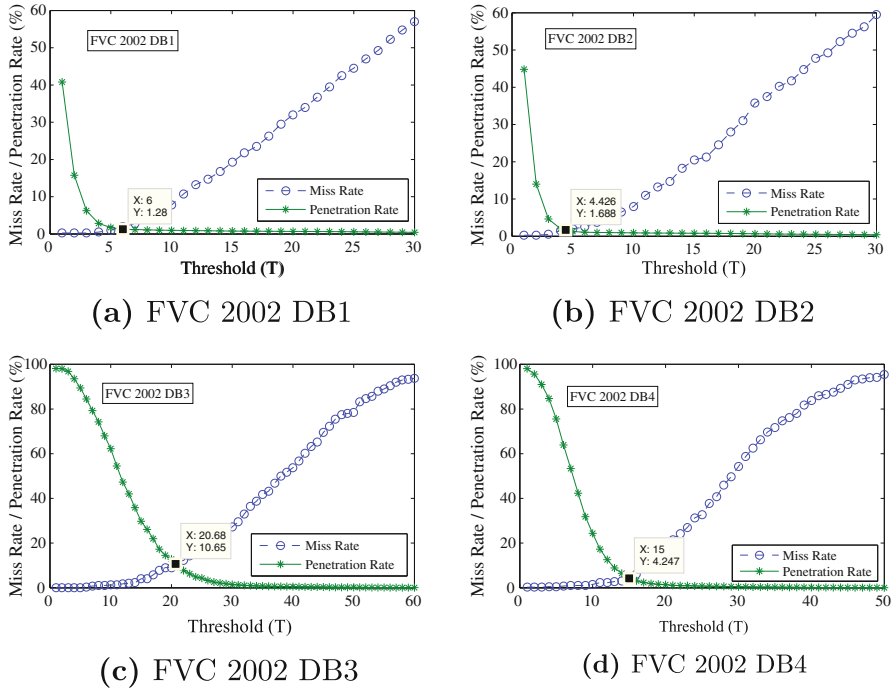


Fig. 2.10 Effect of neighborhood size on the indexing performance for FVC 2004 databases

Table 2.3 Optimum neighborhood size ( $\lambda$ ) obtained for different databases

Database	Optimum $\lambda$ value	Window size
FVC 2002 DB1	3	$7 \times 7$
FVC 2002 DB2	4	$9 \times 9$
FVC 2002 DB3	4	$9 \times 9$
FVC 2002 DB4	5	$11 \times 11$
FVC 2004 DB1	4	$9 \times 9$
FVC 2004 DB2	4	$9 \times 9$
FVC 2002 DB4	3	$7 \times 7$

between  $MR$ ,  $PR$ , and  $T$  is plotted in Figs. 2.11 and 2.12 for FVC 2002 and FVC 2004 fingerprint databases, respectively. It is observed that, for small values of  $T$ ,  $MR$  is low and  $PR$  is high which is not desirable. On the other hand, increasing the  $T$  value decreases the  $PR$  but it increases the  $MR$ , which is also not desirable. High  $PR$



**Fig. 2.11** Performance of the proposed indexing approach on FVC 2002 databases

results in more search time, while high MR results in a less secure system. However, as noted earlier, an effective identification system should have low MR as well as PR. Hence, the performance of the system at  $MR = PR$  is recorded (Figs. 2.11 and 2.12).

It is observed that, for FVC 2002 DB1, at  $MR = PR$ , the system achieves a PR and MR of 1.28%. In other words, the system searches only 1.28% of the database and genuine image is identified (i.e.,  $HR$ ) with an accuracy of 98.72% (i.e.,  $100 - 1.28\%$ ). Further, the system achieves a PR and MR of 1.68, 10.65, 4.24, 9.7, 6.55 and 9.47% for FVC 2002 DB2, FVC 2002 DB3, FVC 2002 DB4, FVC 2004 DB1, FVC 2004 DB2, and FVC 2004 DB4 databases, respectively. The PR and HR of the proposed system at  $MR = PR$  are shown in Table 2.4 for different databases.

### 2.4.3 Comparison with Other Related Approaches

In the next experiment, we compared the performance of the proposed approach with Delaunay triplets [20] and Extended triplet approaches [17]. Figures 2.13 and 2.14

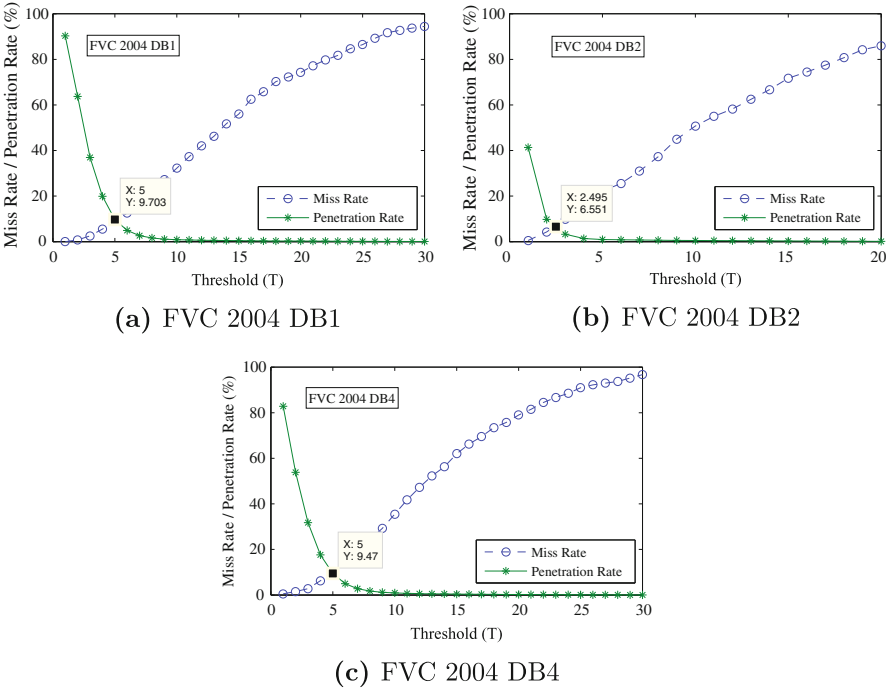


Fig. 2.12 Performance of the proposed indexing approach on FVC 2004 databases

Table 2.4 PR and HR (i.e.,  $HR = 100 - MR$ ) of the proposed system at  $MR = PR$  for different databases

Database	PR(%)	HR(%)
FVC 2002 DB1	1.28	98.72
FVC 2002 DB2	1.68	98.32
FVC 2002 DB3	10.65	89.35
FVC 2002 DB4	4.24	95.76
FVC 2004 DB1	9.7	90.3
FVC 2004 DB2	6.55	93.45
FVC 2002 DB4	9.47	90.53

show the results of different approaches on FVC 2002 and FVC 2004, respectively. The PR of the proposed approach is less compared to Delaunay and extended triangulation-based approaches for most of the datasets. This shows that the partitioning of the index space results in reducing the search space during identification.



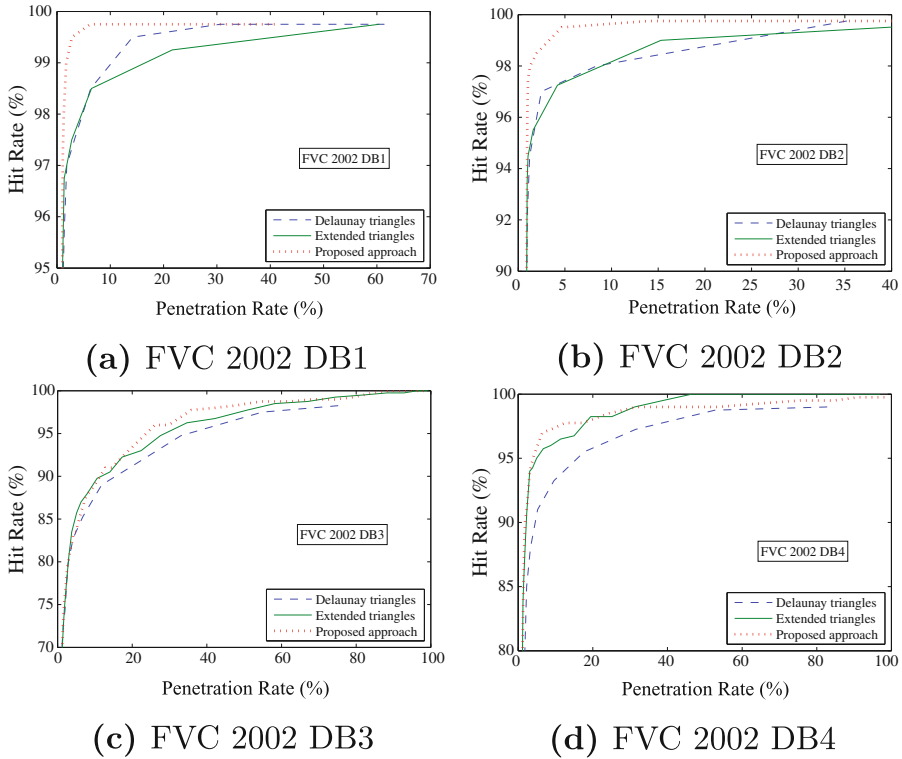
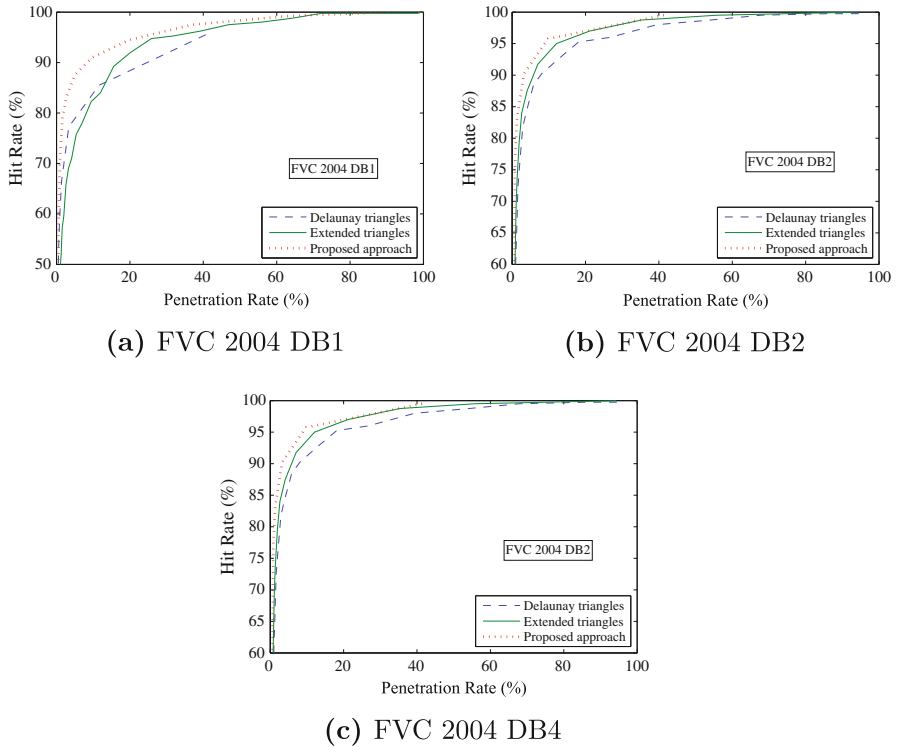


Fig. 2.13 Comparison of the proposed approach with other approaches over FVC 2002 Databases

### 2.4.4 Retrieval Time

We analyze the retrieval time of the proposed approach with big-O notation. Let  $n$  be the number of triplets in the extended set of query image  $q$ ; and  $N$  be the number of images in the database. Note that  $n \ll N$ , as seen from Algorithm 2.2, for a given query triplet  $t$ :

- First, we compute its class  $t_c$ , its feature vector  $f$ , and its index  $X$ . Note that each of these takes  $O(1)$  time.
- Next, we retrieve all the  $I_{id}$ s from  $I_{id}Lists$  corresponding to the bins from  $X - \lambda$  to  $X + \lambda$  into a set named “ $Im$ ”. This process takes  $O(1)$  time.
- Then, a set of  $I_{id}$ s are retrieved from the  $Im$  into a temporary list  $L$ , whose triplet feature vectors are similar to query triplet feature vector. Let  $m$  be the size of the  $Im$ . This process requires  $O(m)$  time. Note that  $m \ll N$ .



**Fig. 2.14** Comparison of the proposed approach with other approaches over FVC 2004 Databases

Hence, the total time required for each query triplet to retrieve the  $I_{id}$ s from the index space into the temporary list  $L$  is  $O(m)$  time. Note that there are  $n$  triplets in the extended set of the query image. So, this retrieval process takes  $O(nm)$  time.

Finally, we count the number of occurrences of each  $I_{id}$  in  $L$  and select top-ranked ones into a candidate set  $C$ . Let the size of  $L$  is  $p$ , where  $p \ll N$ . This process requires  $O(p)$  time. Hence, the total retrieval time for a query image can be approximated as  $O(nm) + O(p)$ .

## 2.5 Summary

In this chapter, an efficient indexing algorithm using hierarchical decomposition of extended triplets is proposed. It has been shown that the proposed algorithm performs better for the fingerprint databases. The decomposition of extended triplets provides better classification in the database, and further reduces search space. Without

increasing the computing cost, the extended triangulation reduces the search space and increases the response time as it produces only  $O(n)$  triplets. Further, this new representation is more robust against distortions compared to all other structures.

## References

1. AADHAAR. **Unique Identification Authority of India**. <http://uidai.gov.in/>, July 2014.
2. Singapore Immigration and Checkpoint authority. <http://www.ica.gov.sg/page.aspx?pageid=407>, september 2014.
3. CROSSING U.S. BORDERS. <http://www.dhs.gov/crossing-us-borders>, July 2014.
4. FIND BIOMETRICS. **Mobile Biometrics, PDAs & Laptop Fingerprint Readers**. <http://findbiometrics.com/applications/mobile-biometrics/>, september 2014.
5. PLANET BIOMETRICS. **Physical Access and Attendance**. <http://www.planetbiometrics.com/physical-access/>, september 2014.
6. **Iris Scans at Amsterdam Airport Schiphol**. <http://www.schiphol.nl/Travellers/AtSchiphol/Privium/Privium/IrisScans.htm>, september 2014.
7. **United Arab Emirates Deployment of Iris Recognition**. <http://www.cl.cam.ac.uk/jgd1000/deployments.html>, July 2014.
8. TSA. **Transportation Security Administration**. <http://www.tsa.gov/>, september 2014.
9. J. HAMMOND. **Biometric traveler screening introduced in Orlando, Denver next**. <http://www.examiner.com/article/biometric-traveler-screening-introduced-orlando-denver-next/>, september 2014.
10. R. BORO AND S.D. ROY. **Fast and Robust Projective Matching for Fingerprints Using Geometric Hashing**. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 681–688, 2004.
11. Y. LAMDAN AND H.J. WOLFSON. **Geometric hashing: A general and efficient model-based recognition scheme**. In *ICCV*, **88**, pages 238–249, 1988.
12. H. MEHROTRA, B. MAJHI, AND P. GUPTA. **Robust iris indexing scheme using geometric hashing of SIFT keypoints**. *Journal of Network and Computer Applications*, **33**(3):300–313, 2010.
13. D.G. LOWE. **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision*, **60**(2):91–110, 2004.
14. SIFT. **SIFT for matlab**. <http://www.vlfeat.org/vedaldi/code/sift.html>.
15. R.S. GERMAIN, A. CALIFANO, AND S. COLVILLE. **Fingerprint Matching Using Transformation Parameter Clustering**. *IEEE Computing in Science and Engineering*, **4**(4):42–49, 1997.
16. B. BHANU AND X. TAN. **Fingerprint indexing based on novel features of minutiae triplets**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(5):616–622, 2003.
17. A.G. ALONSO, J.H. PALANCAR, E.R. REINA, AND A.M. BRISEÑO. **Indexing and retrieving in fingerprint databases under structural distortions**. *Expert Systems with Applications*, **40**(8):2858–2871, 2013.
18. “NEUROTECHNOLOGY”. **VeriFinger SDK**. <http://www.neurotechnology.com/verifinger.html>, January 2014.
19. M.D. BERG, M.V. KREVELD, M. OVERMARS, AND O.C. SCHWARZKOPF. *Computational geometry*. Springer, 2000.
20. G. BEBIS, T. DEACONU, AND M. GEORGIPOULOS. **Fingerprint Identification Using Delaunay Triangulation**. In *International Conference on Information, Intelligence, and Systems*, pages 452–452, 1999.
21. ILAIAH KAVATI, MUNAGA VNK PRASAD, AND CHAKRAVARTHY BHAGVATI. **Vein Pattern Indexing Using Texture and Hierarchical Decomposition of Delaunay Triangulation**. *Security in Computing and Communications*, pages 213–222, 2013.

22. ILAIAH KAVATI, VAMSHIKRISHNA CHENNA, MUNAGA VNK PRASAD, AND CHAKRAVARTHY BHAGVATI. **Classification of extended delaunay triangulation for fingerprint indexing.** In *Modelling Symposium (AMS), 2014 8th Asia*, pages 153–158. IEEE, 2014.
23. N. BIGGS, E.K. LLOYD, AND R.J. WILSON. *Graph Theory, 1736-1936*. Clarendon Press, 1986.

Efficient Biometric Indexing and Retrieval Techniques  
for Large-Scale Systems

Kavati, I.; Prasad, M.V.N.K.; Bhagvati, C.

2017, XVII, 67 p. 29 illus., Softcover

ISBN: 978-3-319-57659-6