An Efficient Indexing Scheme Based on K-Plet Representation for Fingerprint Database

Chaochao Bai[1], Tong Zhao[2(✉)], Weiqiang Wang[1], and Min Wu[3]

[1] School of Computer and Control,
University of Chinese Academy of Sciences, Beijing, China
baichaochaol2@mails.ucas.ac.cn, wqwang@ucas.ac.cn
[2] School of Mathematical Sciences,
University of Chinese Academy of Sciences, Beijing, China
zhaotong@ucas.ac.cn
[3] Eastern Golden Finger Technology Co. Ltd, Beijing, China
wumin@egafis.com

Abstract. Fingerprints are now widely employed in the security fields. A typical police fingerprint database may contain millions of template fingerprints. Consequently, fingerprint indexing plays an essential role to improve the performance of matching such a huge database. In this paper, the efficient index tree based on k-plet local patterns of minutiae for fingerprint database is proposed. The proposed algorithm is of robustness since the k-plet is translation-invariant and rotation-invariant, moreover, the multipath indexing strategy is introduced at the stage of indexing. As well, it is quite fast and effective due to look-up operation instead of complex computation. The performance testing was conducted in the datasets of FVC2002 DB1, NIST DB4 and NIST DB14, which concluded that the proposed algorithm is advantageous for fingerprint indexing since it achieves a high correct index performance with a fairly low penetration rate.

Keywords: Fingerprint indexing · Index tree · K-plet local pattern

1 Introduction

Fingerprints, consisting of the ridges and furrows on human fingers, have been one of the most describable biometric traits which are largely employed for authentication and identification tasks in the fields of civil and forensic systems. A number of features could be captured from a fingerprint to identify a person through a fingerprint recognition system [1]. Generally, the fingerprint recognition system operates in either verification mode or identification mode. For the latter one, identification of an unknown template operating in a 1:N matching process over a huge database usually consumes significant time in order to provide a reliable conclusion. Although state-of-the-art fingerprint matching algorithms are fast and accurate, the size of the database can be over one hundred million (e.g., the police database of a country) and it poses much more challenges for the recognition accuracy and efficiency.

D.-S. Huang et al. (Eds.): ICIC 2015, Part I, LNCS 9225, pp. 247–257, 2015.
DOI: 10.1007/978-3-319-22180-9_25

To address the above challenges, fingerprint classification and fingerprint indexing are the most common solutions. In the terms of fingerprint classification, Henry distinguished fingerprints into 5 classes including left loop, right loop, whorl, arch, and tented arch [2]. However, the number of classes is small and fingerprints are unevenly distributed among them. On contrary, fingerprint indexing is a more efficient approach where fingerprints are specified with feature vectors. These feature vectors are generated through a similarity-preserving transformation and similar fingerprints are mapped into close points (vectors) in the multidimensional space. The indexing is performed by matching the query fingerprint against the template fingerprints in the database whose vectors are close to the query one. Subsequently, the top N most similar template fingerprints are returned. Since it is only necessary to match N candidates instead of every fingerprint of the database, the fingerprint indexing narrows the number of the large database effectively.

Fingerprint indexing is challenging and promising so that there have been lots of approaches to improve the related techniques, in which the involved features can be generally classified into minutiae and global features. In global feature methods, singular points or orientation fields represent the global information of ridges. In [3] and [4], the algorithms adopt singular points as features to index fingerprints. However, these techniques mostly require pre-alignment of fingerprints and it is difficult to extract reliable location of singular point from poor fingerprint images. In addition, feature vectors for fingerprints indexing are received by orientation fields [5, 6]. However, these features are still global characters and their discrimination is not as good as minutiae.

In minutia feature methods, the intrinsic idea is to establish a structure of minutiae that is of enough discrimination and robustness in presence of rotation and translation variations. Depending on these minutia features, there mainly exist two indexing techniques, namely hash-based indexing and Approximate Nearest Neighbor (ANN) indexing. The algorithm [7, 8] based on minutiae derives triangle or quadrangle geometric features and adopts simple hashing technique for searching. Unfortunately, these geometric features are more sensitive to noise and distortion. On the other, Minutiae Cylinder Codes (MCC) characterize a minutia neighborhood through encoding the neighborhood of each minutia into a fixed-length bit vector and the algorithm indexes by means of a typical kind of ANN indexing technique, Locality Sensitive Hashing (LSH) [9]. The representation of a minutia's neighborhood in MCC seems quite complicated for the reason that it has a very high-dimensional feature vector. Furthermore, Locality Sensitive Hashing (LSH) is an approximate searching method which is not quite applicable for the cases requiring high accuracy rate.

In this paper, to our knowledge, the k-plet local pattern of minutiae [10] as feature is the first time to be imported in the field of fingerprint indexing. The k-plet representation is invariant under translation and rotation since it is based on its own local coordinate system. In addition, an efficient tree based indexing scheme designed for the k-plet representation is proposed to accelerate retrieval. In order to make this indexing scheme more robust, the multipath indexing strategy is employed which means that brother bins located in a certain range of the hit bin are all browsed at the stage of indexing. Furthermore, this indexing scheme is quite fast since it just need to look up index trees avoiding the complex computation. The k-plet local pattern is enrolled in

k 3-level index trees, then a query only need to orderly index k trees and count the matched votes.

The rest of this paper is organized as follow: Sect. 2 introduces the k-plet local pattern representation while Sect. 3 describes the efficient indexing approach. In Sect. 4, experiments for testing the proposed algorithm are conducted on public datasets. Finally, Sect. 5 draws some conclusions.

2 K-Plet Local Pattern

In the field of fingerprint matching, an algorithm named as k-plet presents excellent performance, which consists a fixed-length vector by quantizing the distance and orientation difference between a central minutia and its neighboring minutiae, Fig. 1.

Local pattern for indexing fingerprints is characterized based on the k-plet to get the local structural information in fingerprints. The main idea of k-plet is to represent a central minutia by using the nearest k neighbors of the central minutia. To solve the problem that minutiae are clustered and to maintain high connectivity in fingerprint image, k/4 nearest neighbors are sequentially selected in each of the four quadrant in local coordinate system of minutia m_i. It is noteworthy that the resulting k neighbors need not necessarily be the k closest neighbors of minutia m_i. The k-plet representation is invariant under translation and rotation since it is defined with its own local coordinate system. The advantage of these k neighbors is that it still performs well in the situation where the fingerprints exist missing or spurious minutiae. In this study, we considered 8 neighbors empirically (*i.e.*, k = 8), Fig. 2(a).

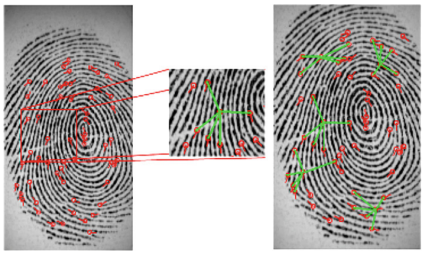

Fig. 1. K-plet local patterns of minutiae defined in a fingerprint [10]

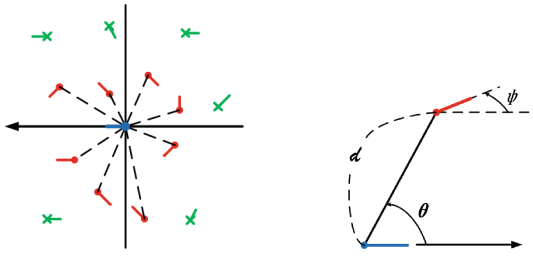

Fig. 2. (a). An example of k-plet **(b)**. IFV formed by a neighbor minutia and central minutia

The k-plet consists of a central minutia m_i and other k neighborhood minutiae defined as $\{m_1, m_2, m_3, \ldots, m_k\}$. Each k-plet owns its local coordinate in which the central minutia m_i is the original point and the direction of the central minutia m_i is chosen as the positive direction of the X axis. Each neighboring minutia $m_j (j = 1, 2, \ldots, k)$ is defined with its local radial coordinates $(d_{ij}, \varphi_{ij}, \theta_{ij})$, where d_{ij} represents the Euclidean distance between minutia m_i and neighboring minutia m_j; φ_{ij} is the relative orientation of minutia m_j with respect to the central minutia m_i; θ_{ij} represents the direction of the edge connecting the two minutiae and θ_{ij} is also measured relative to orientation of minutia m_i. The translation-invariant and rotation-invariant vector $(d_{ij}, \varphi_{ij}, \theta_{ij})$ formed by minutia m_i and minutia m_j is represented as $A_j = (d_{ij}, \varphi_{ij}, \theta_{ij})$, named as *IFV* (Invariant Feature Vector), Fig. 2(b). For each k-plet, the k neighboring minutiae $\{m_1, m_2, m_3, \ldots, m_k\}$ and the central minutia m_i form k invariant feature vectors $\{A_1, A_2, A_3, \ldots, A_k\}$. Thus, the k-plet local pattern is defined in terms of $LP = \{A_1, A_2, A_3, \ldots, A_k\}$.

3 Indexing Approach

In the proposed approach, it consists of two stages, known as enrollment of template fingerprints $\boldsymbol{T}$ and indexing of query fingerprints $\boldsymbol{I}$. Figure 3 illustrates a general flow chart of the proposed approach.

At the stage of enrollment, k index trees are constructed for storing the k-plet local patterns orderly and per tree has 3 levels representing d, φ and θ respectively. Finally,

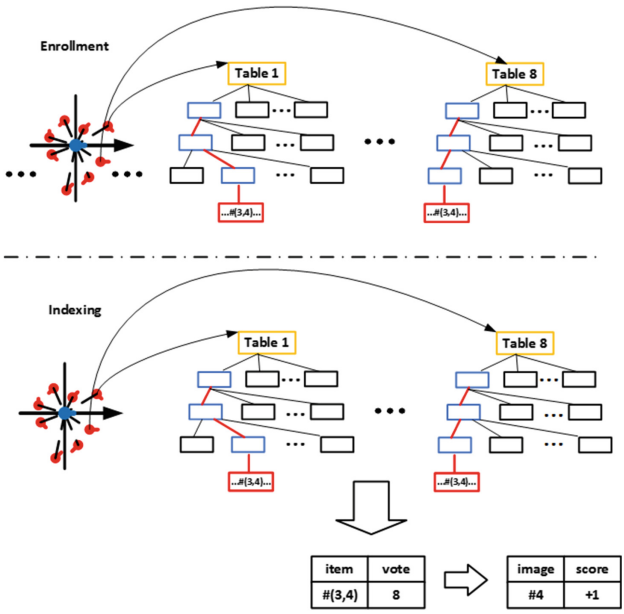

Fig. 3. The flow chart of the proposed indexing approach

the items # of *LP* ID and image ID is enrolled in the corresponding leaf bins. At the stage of indexing, the corresponding branches are selected. In the leaf bins, the items which are hit will be cast one vote. If votes are more than a certain threshold, the two *LP*s are regarded as matched successfully. According to the number of matched *LP*s, the similarity scores S between input fingerprint $\boldsymbol{I}$ and template fingerprint $\boldsymbol{T}$ can be acquired by Eq. (1).

$$S = \frac{\sum_{m} N_m}{N_i + N_t} \tag{1}$$

Where N_i is the number of *LP* in input fingerprint $\boldsymbol{I}$, N_t is the number of LP in template fingerprint $\boldsymbol{T}$, N_m is the number of matched *LP*s between input fingerprint $\boldsymbol{I}$ and template fingerprint $\boldsymbol{T}$. Finally, the output result can be achieved by sorting all template fingerprints with the scores in descending order through the whole database.

3.1 Enrollment of Template Fingerprints

An off-line enrollment is made with the purpose of filling all *LPt* of template fingerprints into k index trees before indexing query fingerprints. Firstly, k duplicate index trees are constructed respectively. Each tree consists of 3 levels, which are presented by: d,φ,θ and there are numbers of fixed-length bins in every level. A $LP_t = (A_1^t, A_2^t, A_3^t, \ldots, A_k^t)$ need to be filled into the $j_{th} (j = 1, 2, \ldots, k)$ index tree depending on the j^{th} invariant feature vector $A_j^t = (d_j^t, \varphi_j^t, \theta_j^t)$. That is, a *LPt* need to be stored totally k times. In the j_{th} index tree, the hash function (2), (3) and (4) at the corresponding level is used to obtain the sequence number n_j^d,n_j^φ and n_j^θ of the hit bin respectively.

$$n_j^d = H_d(d_j^t) = d_j^t / L_d + 1 \tag{2}$$

$$n_j^\varphi = H_\varphi(\varphi_j^t) = \varphi_j^t / L_\varphi + 1 \tag{3}$$

$$n_j^\theta = H_\theta(\theta_j^t) = \theta_j^t / L_\theta + 1 \tag{4}$$

Here, L_d,L_φ and L_θ is the length of one bin at d,φ and θ level and this operator / represents the quotient of the integer division. After the three levels browsed, along with the route in all levels, the item # of *LP* ID as well as fingerprint ID is registered into the last hit bin at the last level.

For example, suppose that we are going to fill the 3rd *LPt* in the 4th template fingerprint in database into the first index tree, which is instantiated as $LP_{3,4} = (A_1^t, A_2^t, A_3^t, \ldots, A_8^t)$, where the 1st invariant feature vector $A_1^t = (5, 8, 15)$. In addition, we assume that $L_d = L_\varphi = L_\theta = 10$. The enrollment process seems to be regarded as tracing the branches of the 3 levels in the 1st index tree. As $d = 5$, then $n_1^d = 5/10 + 1 = 1$, it need to be fell into the first bin of the first level, and all the rest branches should be traced along with $A_1^t = (5, 8, 15)$. At the end, the item $\#(3, 4)$ of *LPt* ID and fingerprint ID is enrolled into the second bin of the last level, Fig. 4.

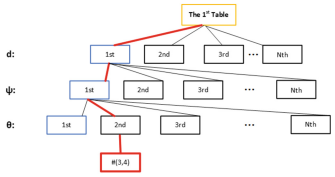

Fig. 4. An example of enrolling an *LPt* into the 1st hash table

Similarly, depending on the following $A_j^t = (j = 2, 3, \ldots, 8)$, the item $\#(3, 4)$ should be enrolled into the rest index trees by order (*i.e.*, the 2nd, 3rd,…,8th index tree).

Summary of the off-line enrollment process is given in Algorithm 1.

```
Algorithm 1: Enrollment Algorithm
Input: the whole fingerprint database, DB
Output: K index trees
for all template fingerprints T in DB do
  for all minutiae m_t in T do
    Construct k-plet local patterns, get LPt
    for K neighboring minutiae in LPt do
      Select corresponding index tree by order
      for K index trees do
        Calculate sequence number n_d , n_φ , n_θ respectively
        Choose corresponding bin in different levels
        Register item # of LPt ID and image ID in leaf
      end for
    end for
  end for
end for
```

3.2 Indexing of a Query Fingerprint

Indexing of a query fingerprint aims to obtain the similar template fingerprints by looking up k established index trees. It is similar to the process of enrollment. A $LP_i = (A_1^i, A_2^i, A_3^i, \ldots, A_k^i)$ is need to be browsed into $j_{th}(j = 1, 2, \ldots, k)$ index tree depending on the j^{th} invariant feature vector $A_j^i = (d_j^i, \varphi_j^i, \theta_j^i)$. Within the j^{th} index tree, each sequence number n_j^d,n_j^φ,n_j^θ will be got by the hash function (2), (3), (4) separately. One bin in the last level will be searched after three levels' routing. All the template items filled in this bin will be recorded into a vote box, moreover the number of votes of these items will plus one. After searching all k index trees, the template items whose the number of votes is more than T will be treated as successfully matched with this *LPi*. According to the number of matched LPs, the similarity scores S between input fingerprint $\boldsymbol{I}$ and template fingerprint $\boldsymbol{T}$ can be calculated by Eq. (1). Finally, the output result can be achieved by sorting all template fingerprints with the scores in descending order through the whole database.

Summary of the on-line indexing process is shown in Algorithm 2.

```
Algorithm 2: Indexing Algorithm
Input: query image, I
Output: list of top N template fingerprints
for all minutiae m_i in I do
  Construct k-plet local patterns, get LPi
  for K neighboring minutiae in LPi do
    Select corresponding index tree by order
    for K index trees do
      Calculate sequence number n_d , n_φ , n_θ respectively
      Select corresponding bin in different levels
      Vote item # of LPi ID and image ID in leaf
    end for
  end for
  if votes of one LPi>T, then
    label LPi with LPt pairs
    for all labeled pairs do
      Calculate scores of query I and T by equation (1)
    end for
  end if
end for
Rearrange all templates in descending order with scores
Output list of top N candidates
```

3.3 Multipath Indexing Strategy

In practice, there are some inherent variations in the fingerprints, including translation, rotation, distortion and other noise. They get reflected in the indexing procedure and thus reduce the overall accuracy of the system. To account for errors that might be caused due to such variations, we implement the multipath techniques in our indexing scheme.

At the stage of indexing, we select the hit bin and also its brother bins according to (5), (6) and (7) at different levels.

$$|n_d - n_{d0}| \leq T_{nd} \tag{5}$$

$$|n_\varphi - n_{\varphi 0}| \leq T_{n\varphi} \tag{6}$$

$$|n_\theta - n_{\theta 0}| \leq T_{n\theta} \tag{7}$$

Where T_{nd}, $T_{n\varphi}$ and $T_{n\theta}$ are some certain thresholds. That is to say, if bin n_{d0} is hit at level d, all its brother bins between $(n_{d0} - T_{nd}, n_{d0} + T_{nd})$ are also selected. Finally, in this way, a *LPi* will search certain numbers of leaf bins so that it increasing the probability of indexing the correct item. Figure 5 gives a general example, where $T_{nd} = T_{n\varphi} = T_{n\theta} = 1$.

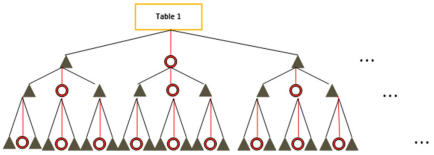

Fig. 5. An example of multipath indexing scheme. Circles are hit bins and triangles are their brother bins.

4 Experimental Results

This section describes experiments carried out to evaluate the proposed algorithm (*LP* + *Idx*) and to compare it with the algorithm (*OrigLP*) of original k-plet local pattern without indexing. As well, the time analysis of the proposed algorithm are given at the end.

4.1 Datasets

The experimental verification of *LP* + *Idx* and its comparison with *OrigLP* are performed based on three standard databases:

- FVC2002 DB1 database [11]: it contains 800 fingerprints, eight impressions for each of 100 distinct fingers, with the images of 388*374 pixels.
- NIST DB4 database [12]: it contains 4000 fingerprints, two different fingerprint instances (F and S) for each of 2000 distinct fingers, with the images of 512*512 pixels.
- NIST DB14 database [13]: we choose the first 2000 fingerprints, two impressions for each of 1000 distinct fingers, with the images of 832*768 pixels.

4.2 Evaluation Indicators and Setup

Typically, Correct Index Power (*CIP*) and Penetration Rate are adopted to evaluate the accuracy and efficiency. Here $CIP = N_{ci}/N_d$, where N_{ci} is the number of correctly retrieved input query fingerprints, N_d is the number of all input query fingerprints. Obtaining a higher *CIP* and a lower Penetration Rate is the common target of fingerprint indexing algorithm.

In the experiments, each database is equally divided into two parts: input fingerprints and template fingerprints. In FVC2002 DB1 database, the first 4 impressions of each individual finger are selected as the input fingerprints while the rest 4 impressions are regarded as the template fingerprints to build the index trees. Similarly, the first and second impressions of each individual finger in NIST DB4 and DB14 database are respectively chosen as input and template fingerprints.

For the feature extraction, an open investigation named as FM3 [14] is adopted to provide the reliable manual-marked minutiae in FVC2002 DB1 database. Meanwhile, a

commercial automatic fingerprint identification system GAFIS [15] is used for the minutiae extraction in NIST DB4 and DB14, which is widely employed in Chinese criminal investigation departments.

4.3 Results and Discussions

Figures 6 and 7 show the trade-off between Penetration Rate and *CIP* in FVC 2002 DB1, NIST DB4 and NIST DB14 where k = 4 and 8, respectively. Table 1 compares the algorithm *LP* + *Idx* with *OrigLP* at some certain *CIP* in FVC 2002 DB1. In addition, Table 2 lists the time factors in FVC2002 DB1, NIST DB4 and NIST DB14.

From Figs. 6 and 7, it is noteworthy that the indexing performance in FVC2002 DB1, NIST DB4 and DB14 is quite outstanding. Moreover, it indicates the performance in FVC2002 DB1 looks much better than the one in NIST DB4 and DB14. It is mainly induced by two factors: (1) the image in database of FVC2002DB1 is more

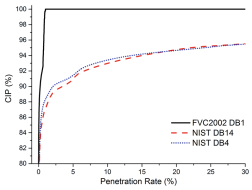

Fig. 6. Indexing performance in FVC 2002 DB1, NIST DB4 and NIST DB14, where k = 4

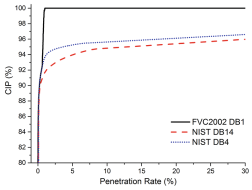

Fig. 7. Indexing performance in FVC 2002 DB1, NIST DB4 and NIST DB14, where k = 8

Table 1. Indexing performance of *LP* + *Idx* and *OrigLP* at certain *CIP* in FVC 2002 DB1

Algorithms \ Penetration Rate \ CIP	*99%*	*97%*	*95%*	*90%*
OrigLP	*0.84%*	*0.77%*	*0.71%*	*0.58%*
LP+Idx(k=8)	*0.85%*	*0.78%*	*0.71%*	*0.58%*
LP+Idx(k=4)	*0.86%*	*0.78%*	*0.71%*	*0.58%*

Table 2. Indexing time in FVC2002 DB1, NIST DB4 and NIST DB14

Algorithms / Time / Database	*FVC02DB1*	*NIST DB4*	*NIST DB14*
OrigLP	*0.159ms*	*1.106ms*	*1.540ms*
LP+Idx(k=8)	*0.004ms*	*0.017ms*	*0.022ms*
LP+Idx(k=4)	*0.003ms*	*0.011ms*	*0.014ms*

clear and smaller than the one in NIST DB4 and DB14. (2) Comparing with the only one template fingerprint for each individual input fingerprint in the database of NIST DB4 and DB14, there are four template fingerprints for each one in the database of FVC2002 DB1, which largely increases the probability of matching one input query fingerprint successfully.

In addition, with the increasing of the number of neighboring minutiae in *LP*, the discrimination of *LP* becomes stronger than before. Indeed, it achieves higher correct index power with a lower penetration rate. However, it will slow down the runtime and consume more memory resource. Consequently, we consider 8 neighbors empirically to achieve a balance between the two aspects (i.e., k = 8).

Furthermore, Table 1 compares the algorithm *LP + Idx* with *OrigLP* at some certain *CIP* in FVC 2002 DB1. The comparison demonstrates that the performance of *LP + Idx* is almost the same to *OrigLP* and verifies the robustness of *LP + Idx*.

For real-time fingerprint recognition system, time is another critical factor related to the efficiency. Consequently, the time factors in FVC2002 DB1, NIST DB4 and NIST DB14 are investigated and listed in Table 2. The time tests running on 2.40 GHz Intel Core CPU are implemented in C++ without particular optimizations. It is obvious to reveal that proposed algorithm *LP+ Idx* is very fast and efficient since these time factors are much faster than *OrigLP*.

5 Conclusion

In summary, we creatively adopt the k-plet local pattern for fingerprint indexing, which is translation-invariant and rotation-invariant. Then, according to these local patterns, k efficient index trees are constructed based on enrollment and indexing stages. The multipath indexing strategy ensures the accuracy and robustness while simple look-up operation makes it quite fast and effective. The performance testing proves that the proposed algorithm is beneficial for fingerprint indexing as it achieves high correct index power with a low penetration rate.

As our future focus, a completely hierarchical fingerprint indexing with other effective features, is expected to develop for large scale fingerprint database.

Acknowledgments. This work was funded by the Chinese National Natural Science Foundation (11331012, 71271204, 11101420)

References

1. Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S.: Handbook of Fingerprint Recognition. Springer, New York (2009)
2. Henry, E.R.: Classification and Uses of Finger Prints. HM Stationery Office, London (1905)
3. Liu, T., Zhu, G., Zhang, C., Hao, P.: Fingerprint indexing based on singular point correlation. In: IEEE International Conference on Image Processing, vol. 3, pp. II-293–6 (2005)
4. Liu, M., Jiang, X., Kot, A.: Fingerprint retrieval by complex filter responses, In: 18th IEEE International Conference on Pattern Recognition, vol. 1, p. 1042 (2006)
5. Wang, Y., Hu, J., Phillips, D.: A fingerprint orientation model based on 2D fourier expansion (FOMFE) and its application to singular-point detection and fingerprint indexing. IEEE Trans. Pattern Anal. Mach. Intell. **29**, 573–585 (2007)
6. Liu, M., Jiang, X., Kot, A.C.: Efficient fingerprint search based on database clustering. Pattern Recogn. **40**(6), 1793–1803 (2007)
7. Bhanu, B., Tan, X.: Fingerprint indexing based on novel features of minutiae triplets. IEEE Trans. Pattern Anal. Mach. Intell. **25**, 616–622 (2003)
8. Iloanusi, O., Gyaourova, A., Ross, A : Indexing fingerprints using minutiae quadruplets. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 127–133 (2011)
9. Cappelli, R., Ferrara, M., Maltoni, D.: Fingerprint indexing based on minutia cylinder-code. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 1051–1057 (2011)
10. Chikkerur, S., Cartwright, A.N., Govindaraju, V.: K-plet and coupled BFS: a graph based fingerprint representation and matching algorithm. In: Zhang, D., Jain, A.K. (eds.) ICB 2005. LNCS, vol. 3832, pp. 309–315. Springer, Heidelberg (2005)
11. Maio, D., Maltoni, D., Cappelli, R., Wayman, J.L., Jain, A.K.: FVC2002: second fingerprint verification competition. IEEE Int. Conf. Pattern Recogn. **3**, 811–814 (2002)
12. Watson, C., Wilson, C.: Nist Special Database 4. Fingerprint Database, National Institute of Standards and Technology (1992)
13. Watson, C.: Nist special database 14. Fingerprint Database, National Institute of Standards and Technology (1993)
14. Kayaoglu, M., Topcu, B., Uludag, U.: Standard fingerprint databases: Manual minutiae labeling and matcher performance analyses, CoRR, vol.3, abs/1305.1443 (2013)
15. GAFIS7.0, 2012. http://www.etgoldenfinger.com/