Fingerprint Indexing Based on Local Arrangements of Minutiae Neighborhoods

Akhil Vij and Anoop Namboodiri
International Institute of Information Technology
Hyderabad, 500032, India
akhil.vij@research.iiit.ac.in, anoop@iiit.ac.in

Abstract

This paper proposes a hash-based indexing method to speed up fingerprint identification in large databases. For each minutia, its local neighborhood information is computed with features defined based on the geometric arrangements of its neighboring minutiae points. The features used are provably invariant to translation, rotation, scale and shear. These features are used to create an affine invariant local descriptor, called an arrangement vector, for each minutia. To account for missing and spurious minutiae, we consider subsets of the neighboring minutiae and hashes of these structures are used in the indexing process. The primary goal of the work is to explore the effectiveness of affine invariant features for representing local minutiae structures. Experiments on FVC 2002 databases show that representation is quite effective even though the technique performs slightly below the state-of-the-art methods. One could use the representation in combination with other techniques to improve the overall performance.

1. Introduction

Fingerprints have long been used as a reliable biometric for identifying a person. Fingerprints are popular because of their ease of capture, distinctiveness, and persistence over time [5] and are commonly employed in both verification and identification systems.

Although state-of-the-art fingerprint matching algorithms are fast and highly accurate [5], identification over a large database is an open problem and poses many challenges. The size of the database can be over a hundred million in some forensic and civilian applications and there can be significant distortions between different impressions of the same finger. Due to the scale of these databases, performing a sequence of one-to-one verifications to solve an identification problem is not a feasible approach. We need to use efficient filtering techniques to narrow down the portion of database to be searched. The most common solutions are fingerprint classification, indexing, and filtering. Classification involves labeling each fingerprint image into one of a few known global patterns and restricting the matching of query to sample of the same class from the database. Filtering uses a set of light weight comparisons done on all samples in the dataset to narrow down the set of potential candidates to match.

The third approach, that of fingerprint indexing is a generalization of the classification approach, where the database is automatically divided into a large number of possibly overlapping subsets. The indexing function predicts the subsets that need to be searched for each query image. The approach is quite promising and several representation has been proposed recently in this direction. In each case, the goal is to find a mapping (or feature representation), that maps similar fingerprints to close points in a multi-dimensional space. Retrieval is performed by matching the input fingerprint with those in the database whose corresponding vectors are close to the searched one. Based upon the features used, these techniques can be classified as:

Global Representations: Global features like average ridge-line frequency, orientation flow around core points, Poincare index etc. represent the global pattern of ridges with uniform model. The algorithms used in [15] and [16] belong to this category. However, these features are more suited for classification purposes and are not particularly good at handling distortions including translation, rotation, shear, scale, occlusion, clutter and other non-linear distortions. These techniques often require prior alignment of fingerprint images in the database and use the location of singular points. In many low quality images, it is tough to locate the singular points reliably and thus, such images are rejected in this case. These features are usually used in conjunction with more discriminative features to further narrow down the search [16].

Orientation Flow Features: Features such as local ridge line orientations [10, 9] and local ridge-line frequencies [13] fall under this category. However, one disadvantage of using features obtained from orientation image is that these features are not present in the ISO standard minu-

978-1-4673-1612-5/12/$31.00 ©2012 IEEE

tiae templates and have to be computed separately starting with the original image.

Minutiae-based Features: Most minutiae based indexing techniques [12, 18, 6, 4], derive geometric features from sets of minutiae points that are robust in presence of rotations and translations variations and use hashing techniques for searching. Some techniques like [17], and [11] form complex structures from minutiae representations and use them for indexing purposes. *Minutiae Cylinder Codes* or MCC [11] was proposed recently and have been demonstrated to be a highly effective method for representing a minutiae neighborhood for the purposes of fingerprint matching as well as indexing. While the MCC representation of a minutiae neighborhood is not provably invariant to affine deformations, the regularizations performed during the computation make them very robust. In this work, we try to ensure affine-invariance of the minutiae neighborhood features and explore their effectiveness for the purposes of indexing.

Other Features: Features such as Fingercode [2] and SIFT-based features [19] use wavelet responses to encode local textures. Some techniques also try to combine different types of features to improve the results [3]. There are also techniques which are based on match scores [1] and hash functions [14].

As mentioned above we develop a minutiae based feature representation that is provably invariant to affine deformations and hence make it applicable directly to minutiae based templates. The representation does not require detection of singular points or prior alignment of the templates. Unlike MCC, we have even avoided the use of minutiae orientations to make the method applicable to the widest variety of existing templates. We also propose a way of handling missing or spurious minutiae points.

2. Arrangement Vector Representation

The atomic unit of our representation is a fixed-length descriptor for a minutia that captures its distinctive neighborhood pattern in an affine-invariant fashion. This distinctive representation of each minutiae allows us to compare two minutiae points and determine their similarity irrespective of the global alignment.

The process of calculating the arrangement vector for a minutia (p5), shown in Figure 1, is as follows:

- We calculate the nearest n neighbors of minutia p5. In Figure 1, let $n = 7$, and the nearest minutiae are p1,p2,p3,p4,p6, and p7. We then enumerate all combinations of m points of the above n ($\binom{n}{m}$ combinations).
- For each combination, we arrange the m points in clockwise order. Now, we describe the local geometry of these m points around the minutia p5. As shown in Figure 1, let m=6, and let p3, p4, p2, p1, p7 and p6 be the m minutiae arranged in clockwise order. With four points denoted as A, B, C, D, we calculate the following invariant features for indexing :

  Ratio of Areas The first feature φ is the ratio of the areas of the triangles formed by minutiae triplets A, B, C and A, B, D.

  Ratio of Lengths of Largest Side The second feature λ is the ratio of the lengths of the largest side of the triangles formed by minutiae triplets A, B, C and A, C, D.

  Ratio of median and minimum angles The third and fourth features α_1 and α_2 are the ratios of the median and minimum angles of the triangles formed by minutiae triplets A, B, C and A, C, D.

- These features are invariant to affine transformations [4] and remain unchanged even when the fingerprint is translated, rotated, scaled or sheared. A weighted combination of these four features is computed to get one final invariant value that describes the local arrangement of these m points. By sliding the points to regard A, B, C and D in clockwise rotation, m such invariants are calculated (*i.e.,*a,b,c,d,e and f in Figure 1). Thus, *abcdef* represents an arrangement vector for minutia p5.

2.1. Enrolling a Fingerprint

The above vector depends on the initial choice of A, B, C and D points and is not rotation invariant. To achieve rotation invariance, we use cyclic permutations of this vector. Cyclic permutation of these m invariants give us m vectors (*i.e.,*abcdef, bcdefa, cdefab, defabc, efabcd, fabcde). Each vector is considered for hashing and a hash value is calculated from it by Equation 1. The minutia ID, fingerprint ID, along with the arrangement vector is stored in the corresponding hash bin. Separate Chaining technique is applied to resolve collisions that occur when two vectors map to the same hash bin. Summary of the offline Enrollment stage is shown in Algorithm 1.

$$H_{index} = \left(\sum_{i=1}^{m} v[i] \cdot k^i \right) mod\ H_{size} \qquad (1)$$

2.2. Querying the Index

For each minutia p' in a query image and for each combination of m points around that minutia, we calculate its arrangement vector v'' as described earlier. The hash value of v'' is computed, and the corresponding list of fingerprints that contain a similar minutiae neighborhood is obtained from the hash table. Each minutia in the query fingerprint

Algorithm 1 Enrollment Algorithm

INPUT → Entire Fingerprint Database db, n, m, k
OUTPUT → Model Hash Table
for all fingerprint image *fp* in *db* **do**
  for all minutia p in *fp* **do**
    N → nearest n neighbors of minutia p
    L → list of all possible combinations of m points
    for all combination of m points in L **do**
      find arrangement vector *v*
      C → list of all cyclic permutations of *v*
      for all vector v' in list C **do**
        calculate H_{index} from *v'* using eq.1
        register item (Fingerprint ID of *fp*, Minutia ID of p, Arrangement vector *v'*) using H_{index}
      end for
    end for
  end for
end for

casts a vote for each fingerprint in its candidate list. Finally, a list of top N fingerprints with the maximum votes is returned as the output of the indexing algorithm. Summary of the on-line indexing stage is given in Algorithm 2.

Algorithm 2 Indexing Algorithm

INPUT → Query image im, n, m, k, N
OUTPUT → List of top N fingerprints sorted by number of votes received
for all minutia p' in *im* **do**
  N → nearest *n* neighbors of minutia p'
  L → list of all possible combinations of m points
  for all combination of m points in L **do**
    find arrangement vector *v"*
    calculate H_{index} from *v"* using eq.1
    lookup Hash Table with H_{index} and retrieve the corresponding list
    for all $item$ in the retrieved list **do**
      if v''==$item$.Arrangement vector **then**
        Increment vote count for FingerprintID corresponding to $item$
      end if
    end for
  end for
end for
Sort all fingerprints according to vote counts in descending order
Output list of top N as the indexing result

3. Experiments and Discussions

The experiments were conducted on the four *FVC 2002* databases: DB1, DB2, DB3 and DB4. Each database contains 800 fingerprints from 100 users (8 impressions per user). For each user, the first 4 impressions were placed in the gallery to build the hash table while the remaining 4 impressions were used as probes. Experiments were conducted with different values of n, m and k. The best results were observed for n=6, m=5 and k=28 and $H_{size} = 1000000$. Accuracy and efficiency are two main indicators of the retrieval performance. In the experiments, the accuracy is denoted by *Correct Index Power (CIP)* where CIP = (N_{ci}/N_d), N_{ci} is the number of correctly indexed probe images, and N_d is the number of images in the database. The retrieval efficiency is indicated by the *Penetration Rate*, which is the average percentage of database probed over all test fingerprints. Ideally, we would want a high CIP and a low Penetration Rate.

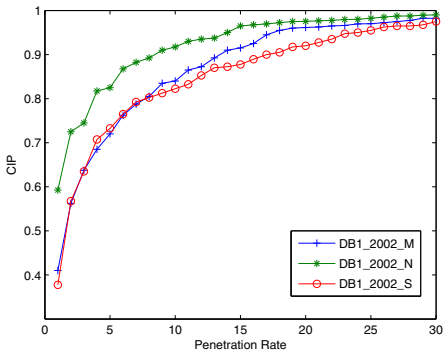

Figure 2. Performance on FVC 2002 DB1 database in case of 20% missing minutiae data(M), original minutiae (N) and 20% spurious minutiae (S).

Dealing with missing or spurious minutiae

Handling the case of missing or spurious minutiae is a major challenge for minutiae based indexing techniques [12], [18], [6], [4]. We deal with this problem by first

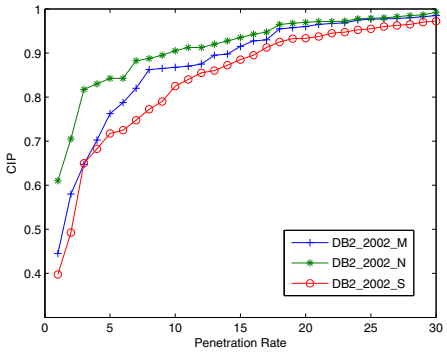

Figure 3. Performance on FVC 2002 DB2 database in case of 20% missing minutiae data(M), original minutiae (N) and 20% spurious minutiae (S).

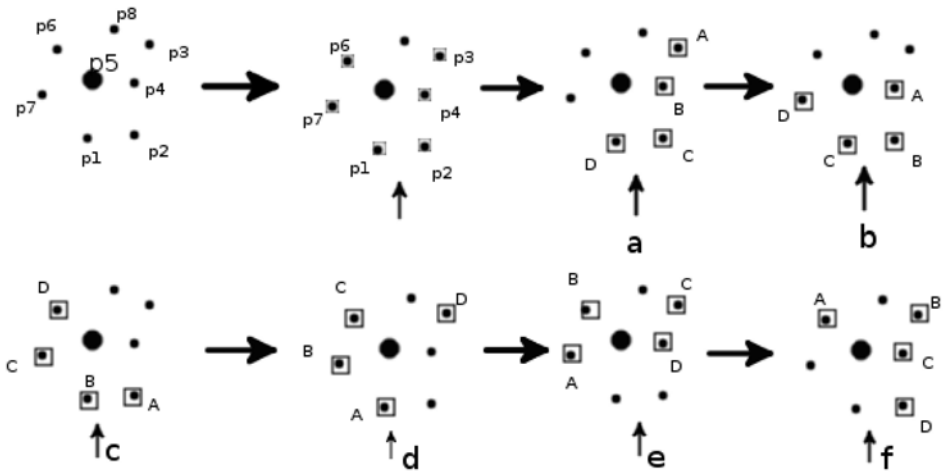

Figure 1. This figure describes the process of creating the Arrangement Vector for minutia p5. In step1, we find the nearest n minutiae of p5 (n=7). In step2, we take all combinations of m points out of those n points. In subsequent steps, we take four points A,B,C,D and calculate invariants a,b,c,d,e,f (see Section 2). *a*bcdef is the required vector that describes the arrangement of p3,p4,p2,p1,p7 and p6 around p5.

choosing nearest *n* neighbors for a minutia and then out of these *n*, we choose all possible combinations of *m* minutiae points. This rule of *choosing m out of n neighbors* helps us to deal with missing and spurious minutia. Experiments were done with datasets having 20% spurious minutiae and 20% missing minutiae. Minutiae were removed and added randomly to the database. For each experiment, the following three cases were considered: datasets in their original form; datasets with 20% spurious minutiae; and datasets with 20% missing minutiae. As the plots show (Figures 2,3,4 and 5), even removing or adding 20% extra minutiae did not affect the low penetration rates at the hit rate of greater than 97%. This shows that the scheme is able to handle low quality noisy images, where there are lots of missing or spurious minutiae points.

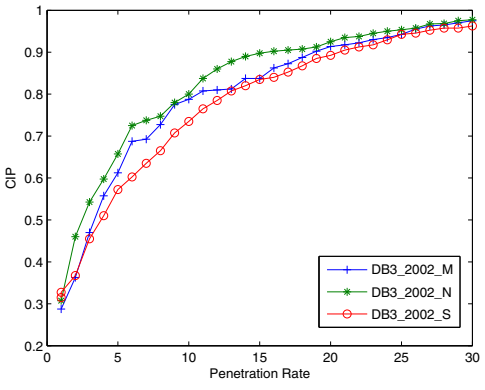

Figure 4. Performance on FVC 2002 DB3 database in case of 20% missing minutiae data(M), original minutiae (N) and 20% spurious minutiae (S).

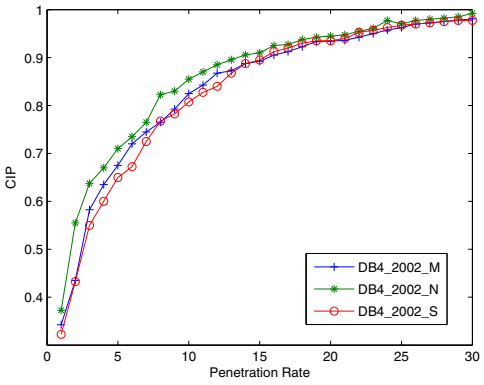

Figure 5. Performance on FVC 2002 DB4 database in case of 20% missing minutiae data(M), original minutiae (N) and 20% spurious minutiae (S).

Dealing with distortions

Handling the case of non-linear distortions and transformations is a major challenge for indexing algorithms that use global features [15] and [16]. We deal with this problem by using features, like ratio of sides, angles and areas, which are invariant to geometric transformations like rotation, translation, scaling and shear [4]. It is known that non-linear distortion happening in local minutiae structures is small enough to be ignored compared with the much larger global non-linear distortion [7]. The arrangement vector, the proposed local minutiae structure, can tolerate non-linear distortion as indicated by the experimental results.

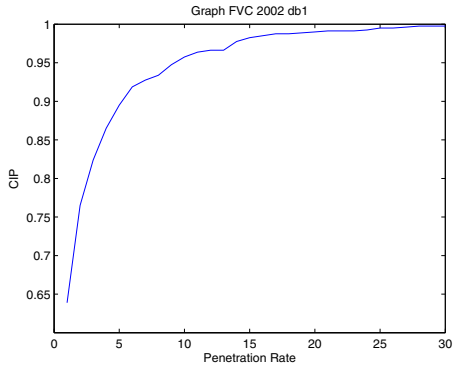

Figure 6. CIP vs. Penetration for database size of 799.

3.1. Time Analysis

The major benefit of using a fingerprint indexing algorithm is that it reduces the number of expensive one-to-one matchings, resulting in significant reduction in the overall time for identification. To find the time required for each identification test, each image in DB1 was used as a probe for identification against a gallery containing all the 799 other images. A test image is considered to be correctly classified, if at least one impression from its class is present in the list returned by indexing algorithm. Figure 6 presents the CIP vs. Penetration plot for this experiments. We now use this result to compute the expected time taken by a new query in a database of this size.

Consider a case in which, upon receiving a test image, the entire DB1 is searched exhaustively for a match. This resulted in 799 one-to-one matches and took a time of **7** seconds in the worst case when entire database was searched. Let *t* be the time taken for one one-to-one match.

$$t = 7/799 = 0.00876\ seconds \tag{2}$$

Now let us consider a second case, when the above indexing algorithm was applied first to get the list of top N candidates and then one-to-one matching was done only with these N candidates. Let t_1 be the time taken to calculate the list and let t_2 be the time taken to do one-to-one matchings with this retrieved list. t_1 is the time taken by indexing algorithm to come up with top N list for a query fingerprint and had a average value of **1.01** seconds over all queries. To calculate t_2, we need to find the expected value of the number of explicit one-to-one matches that we need to do. Let this number be N_e. Using the values of $(CIP, Penetration)$ pairs from graph in Figure 6, (CIP = 0.82 at a Penetration rate of 3, CIP of 0.895 at Penetration of 5, etc.) we can calculate the Expected value of number of matches (for 100% CIP) N_e, as:

$$E(N_e) = (3 * 0.82 + 5 * 0.075 + 10 * 0.0625 + 20 * 0.0325 + 30 * 0.01) * (N_d/100)$$

Therefore, the total time taken for indexing followed by the explicit comparisons, N_e, is given by:

$$t_1 + E(N_e) * t = 1.01 + 35.28 * 0.00876 = 1.32\ s$$

Hence, the total time taken in the second case when indexing algorithm was used came out to be **1.32** seconds, which is much quicker than the **7** seconds it took when the entire database was searched exhaustively. In real-time biometric applications, where time is a critical factor, indexing algorithms play a vital role in creating usable solutions. In other words, given a limit for the response time for an identification query, the use of an indexing algorithm will free up more time for explicit comparisons, and one could employ a more rigorous matching algorithm at this stage, resulting in improved accuracies. The implementation of the indexing algorithm is in Matlab, and hence one can expect to improve the indexing time significantly by an efficient implementation in C/C++.

3.2. Comparison of Results

The proposed algorithm is compared with the quadruplet based indexing algorithm in [8], which is also a minutiae based indexing algorithm. We have selected this work for comparison as, to the best of our knowledge, it has the highest accuracy among the affine invariant representations that have been proposed till date. The Minutiae Cylinder Code uses a richer representation of the minutiae neighborhoods and performs better in practice. However, the feature has only translation invariance and rotation invariance is achieved by orienting the cube using minutiae orientation. Other invariances are not considered. While explicit invariances are not necessary for practical systems (as indicated by MCC results), we prefer to use them as they provide a theoretically sound basis for future analysis of errors.

The evaluation protocol was based on [8], which uses Hit Rate as a measure of correct indexing. Hit Rate is defined as $CIP * 100\%$. As the results in Tables 1 and 2 show, the Hit Rate of the proposed algorithm is better than that reported in [8] at lower penetration rates. However, the quadruplet based algorithm performs better than ours for 100% Hit Rate. Figure 7 compares the Hit Rate vs Penetration graphs of both the algorithms. The tests were run on FVC 2002 DB1 database. The plot for the quadruplet based algorithm is reproduced from the graph given in [8].

In terms of storage, our algorithm requires $m^2.\binom{n}{m}$ numbers (150 in our experiments) to be stored per minutiae, while the quadruplet based indexing only requires around 50 numbers to be stored per fingerprint.

Hit Rate	Minutiae Quad. [8]	Proposed Algo.
60%	6.8	**1**
70%	8.68	**1.9**
80%	10.5	**3.9**
90%	15	**8.6**
95%	17.6	**14**
100%	**21.5**	57

Table 1. Average penetration rates using the proposed and quadruplets [8] approaches 480 at various Hit Rates on FVC 2002 DB1.

Hit Rate	Minutiae Quad. [8]	Proposed Algo.
60%	6.31	**2.8**
70%	8.15	**4.14**
80%	11.8	**8**
90%	17.89	**13.5**
95%	22.89	**17.5**
100%	**27.89**	60

Table 2. Average penetration rates using the proposed and quadruplets [8] approaches at various Hit Rates. The database used was FVC 2002 DB1 with 20% minutiae removed randomly.

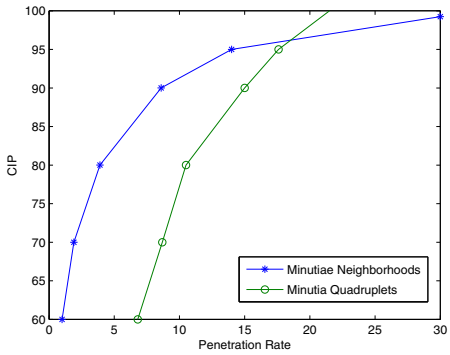

Figure 7. Comparison of the proposed and quadruplets [8] approaches on FVC 2002 DB1.

4. Conclusions and Future Work

Experimental results show that the algorithm efficiently narrows down the size of the database to be searched. It is also shown to be robust under missing or spurious minutiae. The proposed algorithm requires only the locations of the minutiae points and no other high level features such as orientation flow, directional field, etc. This makes the proposed approach applicable to a large variety of existing templates. It also avoids the need for prior alignment of fingerprints or calculation of singular points. We are currently investigating the use of minutiae orientations in the feature computation as well as the use of the representation for the purpose of fingerprint matching. Combination of the proposed feature vector with other existing feature representations such as MCC and Quadruplet features is also being explored for indexing purposes.

References

[1] A.Gyaourova and A.Ross. A novel coding scheme for indexing fingerprint patterns. *Proc. Intl. Workshop on Statistical Pattern Recognition*, Dec. 2008.

[2] A.K.Jain, S.Prabhakar, and L.Hong. A multichannel approach to fingerprint classification. *IEEE TPAMI*, 21(4):348–359, 1999.

[3] A.Ross and R.Mukherjee. Augumenting ridge curves with minutiae triplets for fingerprint indexing. *Proc. SPIE Conf. on Biometric Technology for Human Identification*, 2007.

[4] B.Bhanu and X.Tan. Fingerprint indexing based on novel features of minutiae triplets. *TPAMI*, 25(5):616–622, 2003.

[5] D.Maltoni, D.Maio, A.K.Jain, and S.Prabhakar. Handbook of fingerprint recognition, second ed. springer. 2009.

[6] J.Boer, A.Bazen, and S.Gerez. Indexing fingerprint databases based on multiple features. *Proc. Ann. Workshop Circuits, Systems and Signal Proc.*, pages 300–306, 2001.

[7] Z. Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE TPAMI*, 22(11):1266–1276, 2000.

[8] O.Iloanusi, A.Ross, and A.Gyaourova. Indexing fingerprints using minutiae quadruplets. *Proc. IEEE Computer Society Workshop on Biometrics at the Computer Vision and Pattern Recognition (CVPR) Conference*, pages 127–133, June 2011.

[9] R.Cappelli, A.Lumini, D.Maio, and D.Maltoni. Fingerprint classification by directional image partitioning. *IEEE TPAMI*, 21(5):402–421, 1999.

[10] R.Cappelli, D.Maio, and D.Maltoni. A multi-classifier approach to fingerprint classification. *Pattern Analysis and Applications*, 5:136–144, 2002.

[11] R.Cappelli, M.Ferrara, and D.Maltoni. Fingerprint indexing based on minutia cylinder-code. *IEEE TPAMI*, 33(5):1051–1057, 2011.

[12] R.Germain, A.Califano, and S.Colville. Fingerprint matching using transformation parameter clustering. *IEEE Computational Science and Eng.*, 4(4):42–49, 1997.

[13] S.Lee, Y.Kim, and G.Park. A feature map consisting of orientation and inter-ridge spacing for fingerprint retrieval. *Proc. of AVBPA*, 2005.

[14] S.Tulyakov, F.Farooq, and V.Govindaraju. Symmetric hash functions for fingerprint minutiae. *Proc. Intl. Conference on Pattern Recognition and Image Analysis*, 2, 2005.

[15] T.Lui, G.Zhu, P.Hao, and C.Zhang. Fingerprint indexing based on singular point correlation. *Proc. Int'l Conference on Image Processing*, 2005.

[16] X.Jiang, M.Liu, and A.Kot. Fingerprint retrieval for identification. *IEEE TIFS*, 1:532–542, 2006.

[17] X.Liang, A.Bishnu, and T.Asano. A robust fingerprint indexing scheme using minutia neighborhood structure and low-order delaunay triangles. *IEEE Trans. Information Forensics and Security*, 2(4):721–733, 2007.

[18] X.Liang, T.Asano, and A.Bishnu. Distorted fingerprint indexing using minutia detail and delaunay triangle. *Proc. Int'l Symp. Voronoi Diagrams in Science and Eng.*, 2006.

[19] X.Shuai, C.Zhang, and P.Hao. Fingerprint indexing based on composite set of reduced sift features. *Proc. Int'l Conf. Pattern Recognition*, pages 1–4, 2008.