Pattern Recognition 54 (2016) 1–13

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Fingerprint indexing with pose constraint

Yijing Su, Jianjiang Feng*, Jie Zhou

State Key Laboratory on Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 22 August 2015
Received in revised form
2 December 2015
Accepted 3 January 2016
Available online 16 January 2016

Keywords:
Fingerprint indexing
Pose estimation
Locality Sensitive Hashing

ABSTRACT

Fingerprint indexing is critical for identifying fingerprints efficiently in large-scaled fingerprint databases. The state-of-the-art indexing accuracies are achieved by minutiae-based indexing approaches. A major difference of these approaches is how they deal with fingerprint registration problem. Some of the approaches do not use registration, while others perform relative/pairwise registration (e.g., based on mated minutiae between two fingerprints). However, the former approach is not accurate, since without geometric constraints, many false matches can be found even for non-mated fingerprints. The latter is not efficient for large databases since the relative registration step has to be performed for each pair of fingerprints. It is desirable to develop an absolute registration approach, which can register any single fingerprint into a common coordinate system, and therefore can address efficiency and accuracy problems simultaneously. In this paper, we proposed a fingerprint pose estimation algorithm which can register fingerprints into a common finger coordinate system. Fingerprint pose estimation problem is viewed as a two-class classification problem and approached by sliding window classifiers trained on labeled data. Ridge orientation information is used as features instead of singularities which are often affected by noise. With the pose estimation, we are able to refine the matched minutiae with global spatial constraint. By combining the proposed pose estimation algorithm with an improved Locality Sensitive Hashing algorithm for MCC descriptor, our indexing system outperformed previous state-of-the-art on widely used databases and its scalability was tested on a large database with one million fingerprints.

1. Introduction

Fingerprint is easy to be measured, and believed to be unique among individuals. These intrinsic characteristics and the great advance in fingerprint recognition technologies make it the most widely used biometrics. Nowadays, large scale fingerprint recognition systems can have more than millions of fingerprints. For the task of identifying a fingerprint in large databases, fingerprint indexing algorithm is usually first used to quickly select a subset of candidates and then accurate but slower matching algorithm is employed to determine the final result [1,2].

Fingerprint indexing approaches can be roughly classified into two categories: level-1 indexing approaches and level-2 indexing approaches. Level-1 approaches use level-1 features, namely, ridge orientation map and ridge frequency map [3–6]. Fingerprint classification can be viewed as a special case of level-1 indexing approach, where the dimensionality of the feature vector is one. Level-2 approaches are based on level-2 features, namely, minutiae. Level-1 indexing approaches are commonly faster than level-2 indexing approaches, as level-1 features can be represented as compact feature vectors. On the other side, since minutiae contain more discriminative information of fingerprints than ridge orientation and frequency maps, level-2 indexing approaches should be more accurate if implemented properly. Some indexing approaches combine both features to make full use of them and get better performance [7,8]. The proposed approach belongs to level-2 indexing approach.

Minutiae-based indexing approaches generally extract a set of invariant features based on minutiae. Examples of invariant features include descriptors of minutiae [9], features of minutia triplets [10–12], and features of minutia quadruplets [13,14]. Most of the level-2 indexing approaches are based on the inverted index [15], which is a popular data structure in information retrieval. The basic framework of these indexing algorithms is briefly described as follows. A fingerprint is represented as a set of invariant features based on minutiae. In the offline stage, inverted index is used to store all the invariant features of all gallery fingerprints. In the online stage, given a query fingerprint, for each of its invariant features, corresponding invariant features in the inverted index

* Corresponding author.
E-mail addresses: suyj10@mails.tsinghua.edu.cn (Y. Su), jfeng@tsinghua.edu.cn (J. Feng), jzhou@tsinghua.edu.cn (J. Zhou).

http://dx.doi.org/10.1016/j.patcog.2016.01.006
0031-3203/

are found and a vote is cast for each corresponding gallery fingerprint. Finally, gallery fingerprints are ranked according to their received votes.

Although most level-2 indexing algorithms adopt the above basic framework, they may have some important differences. Two key differences are whether a global spatial transformation constraint is enforced or not and how the global constraint is enforced. The global constraint requires that the transformation between each matched minutiae pair should be close to a same rigid transformation. Without a global constraint, an invariant feature in the query fingerprint may have many false matches in non-mated gallery fingerprints. By enforcing global constraint correctly, the number of false matches in non-mated gallery fingerprints can be significantly reduced while the mated gallery fingerprint is less affected. See Fig. 1 for an example.

If two fingerprints are registered in the same coordinate system, global constraint can be simply enforced by requiring the matched minutiae to be close in both location and direction. Hence the key of enforcing global constraint is to register fingerprints. There are two different methods for fingerprint registration: absolute registration and pairwise registration. The most common method for absolute registration is to use singular points (loop/delta) for fingerprint registration. However, singular points are not present in some fingerprint images (fingerprints of plain arch type and incomplete fingerprints) and are often affected by noise which makes them not suitable for the use in absolute registration. That is why existing minutiae-based indexing approaches either enforce it by pairwise registration [10] or simply ignore it [9], since pairwise registration is not efficient for large databases.

To efficiently utilize global constraint in fingerprint indexing, we propose a pose estimation algorithm to register fingerprints into a unified coordinate system. We learn classifiers with manually marked samples to detect well positioned fingerprint in an image. We use ridge orientation as dominant features in the algorithm instead of singularities, which makes our algorithm less sensitive to noise. Pose determined by the well positioned fingerprint is shown in Fig. 1. We use the pose as global spatial constraint to refine the matched minutiae. As we can see from the figure, false correspondences are reduced with such constraint, especially for non-mated fingerprints. By combining the proposed pose estimation algorithm with an improved Locality Sensitive Hashing algorithm for MCC descriptor, our indexing system achieved state-of-the-art performance on several public domain databases as shown in experiments.

The rest of this paper is organized as follows: in Section 2, published algorithms for fingerprint indexing and fingerprint pose estimation are reviewed. Our pose estimation algorithm is introduced in Section 3. Details of how we retrieval fingerprints with estimated pose is explained in Section 4. The improved Locality Sensitive Hashing algorithm is also presented in this section. Then,

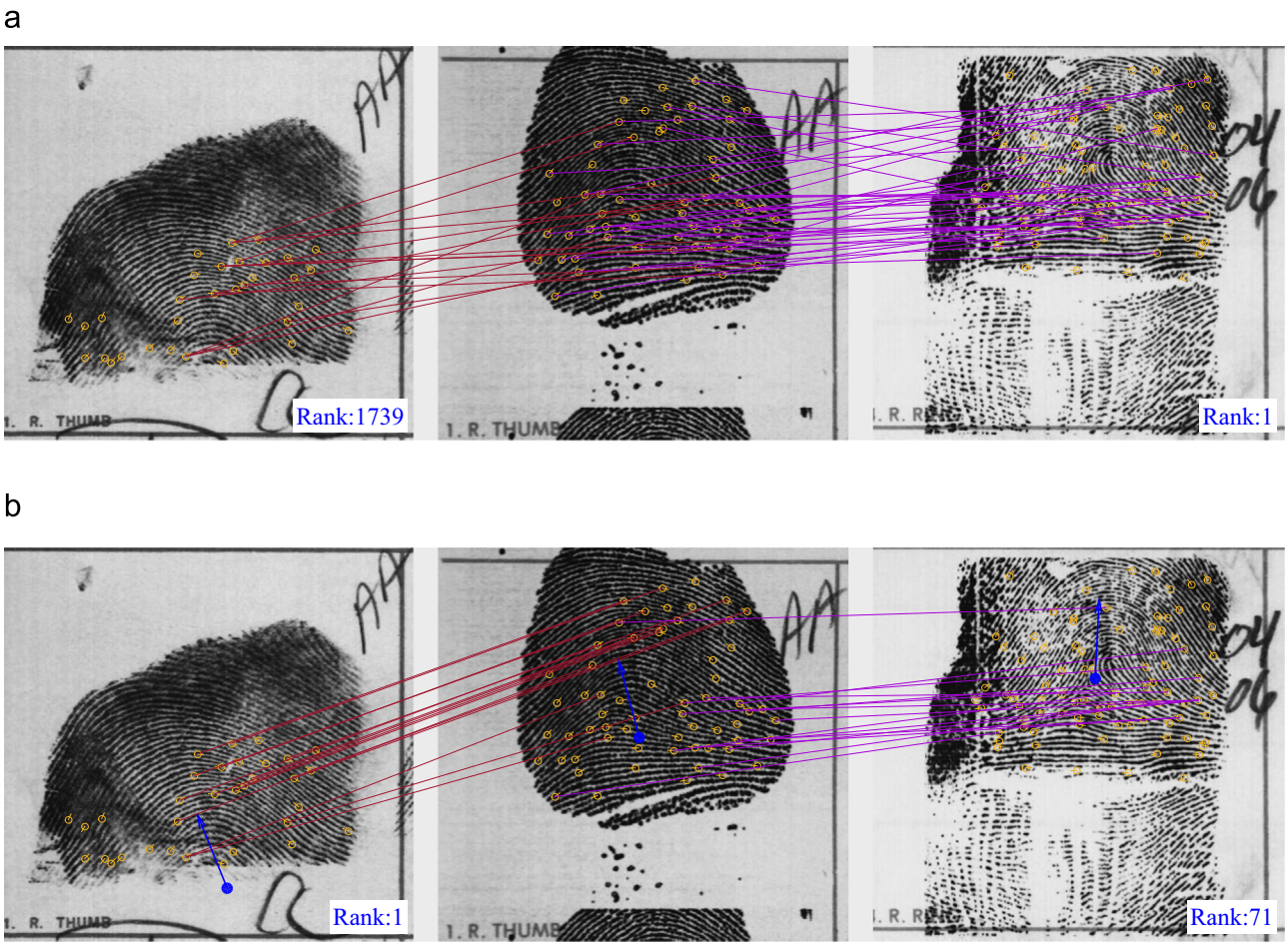

Fig. 1. Indexing results of MCC-LSH based fingerprint indexing algorithm [9] without/with pose constraint. (a) Indexing result with a loose constraint for translation and rotation (256 pixels and 45°, just the same with that in [9]). (b) Indexing result with a tight constraint defined by finger poses estimated by the proposed algorithm. The fingerprint in the middle column is query fingerprint, the left one is its mated gallery fingerprint, and the right one is the gallery fingerprint with highest score in the case of (a). Center point and direction (namely pose) of fingerprints estimated by the proposed algorithm are marked by circle and arrow, respectively. Matched minutiae are connected by lines. The ranks of gallery fingerprints are annotated on the bottom right corner of the images.

in Section 5, the experimental results are presented and analyzed. Finally, we finish with conclusions in Section 6.

2. Related work

2.1. Fingerprint indexing

Table 1 compares the proposed algorithm with representative minutiae-based fingerprint indexing algorithms in terms of minutiae descriptor, index generator, spatial constraint and performance. Germain et al. [10] proposed to use minutiae triplets as descriptor. Geometric features of triplet includes length of each side, the angles and the ridge count between each pair of minutiae. As minutiae triplet is not very discriminating, false correspondences are likely to happen. To reduce false correspondences, pairwise registration is carried out between each gallery fingerprint and query fingerprint using transformation parameter clustering, which is very time consuming.

Bhanu and Tan [11] improved the algorithm in [10] by using new feature of minutiae triplets, and imposing ad hoc geometric constraints on hypothesis matched triplets instead of pairwise registration. They reported higher matching accuracy and faster speed than the original algorithm.

The algorithm proposed in [9] uses binary minutia cylinder code (MCC) as minutia descriptor which encodes locations and directions of neighbor minutiae around each minutia into a fixed-length bit vector. MCC is more accurate and robust than minutia triplets, but its dimensionality is higher. Locality Sensitive Hashing (LSH) is used to efficiently find out hypothesis correspondences of MCC instead of traditional quantization scheme. Since the error rates of other indexing algorithms are larger than Cappelli et al. [9], we do not include them in the table.

2.2. Absolute fingerprint registration

Absolute registration or pre-alignment [1] refers to aligning fingerprint images in a unified manner. With absolute registration, tight constraint on location and direction of minutiae can be applied to reduce false correspondences. Existing absolute registration algorithms can be classified into two categories: special point based and Hough transform based.

Singularities were used for registration in [16]. Despite lots of effort in singularity detection, it is still very sensitive to noise [17]. Besides, fingerprints of arch type do not even have singularities. Other special points are also proposed, such as maximum curvature point on the convex ridge [18], focal point [19], convex core point [20]. However, similar to the detection of singularities, they are sensitive to noise, too.

Yang et al. [21] proposed a Hough transform based fingerprint pose estimation algorithm, which uses a whole fingerprint to predict the pose. They first learn a statistical model of fingerprint orientation field distributions in off-line training stage. Given an input fingerprint, the location of center point is estimated as follows: estimate the orientation field, divide the orientation image into patches, then each patch votes for the potential location of center point and direction of fingerprint according to the learned distribution of the corresponding group. Votes are accumulated and the one with highest votes is regarded as result. Although this algorithm is suitable for its original purpose: orientation field estimation, the estimated poses are not consistent in different impressions of a same fingerprint, which makes it not suitable for fingerprint indexing.

3. Pose estimation

The problem of fingerprint pose estimation is equal to searching for a subwindow containing a centered and upright fingerprint. Rotation of finger has to be considered since many fingerprints are not upright. This problem can be approached using two types of methods: (1) rotate image to all possible directions, and then use a single classifier to scan over the image; (2) scan the original image using classifiers of all possible directions. We adopt the second method since our experiment shows that it is faster.

Specifically, the pose estimation procedure is as follows (See Fig. 2):

1. Input fingerprint image is scanned at all possible locations.
2. Classifiers of various directions are used to determine the possibility that there is a fingerprint of certain direction at each location.
3. Direction and location with highest score (possibility) is chosen as estimated result.

In the following subsections, we present details of our algorithm.

3.1. Fingerprint pose definition

The direction of fingerprint is defined to be perpendicular to finger joint. But it is not easy to define finger center, as evidenced by the fact that there is no consensus on the definition of fingerprint center. We provide a definition suitable for our case:

- For fingerprints with at least one loop and one delta, the loop closer to finger joint and the delta farther away from the loop are chosen as reference points. The center is then defined as the middle point between loop and projective point of the delta on the line parallel with fingerprint direction and crossing the loop.
- Otherwise, as the definition in [21], the midpoint of the maximum curvature point and the northernmost straight ridge perpendicular to the fingerprint direction is defined as the center.

Examples are presented in Fig. 3 to illustrate the definition.

Table 1
Representative minutiae-based fingerprint indexing algorithms.

Algorithm	Feature	Index generator	Registration	Error rate (Penetration rate = 10%)				
				FVC2000 DB2	FVC2000 DB3	FVC2002 DB1	NIST SD4	NIST SD14
Germain et al. [10]	Feature of minutiae triplets	Quantization	Pairwise	–	–	–	16.5%[a]	-
Bhanu and Tan [11]	Feature of minutiae triplets	Quantization	No	–	–	–	14.5%	-
Cappelli et al. [9]	MCC	LSH	No	2.2%	6%	1%	4%	5%
Proposed[b]	MCC	Improved LSH	Absolute	1.49%	5.86%	0.714%	2.97%	2.33%

[a] According to the implementation by Bhanu and Tan [11].
[b] Registration is not used in FVC databases.

3.2. Training examples

According to the above definition, we manually marked the poses of a set of training fingerprint images. A supporting tool is developed to make the process easier. Based on the marked poses, we generate training samples for classifier of each direction. Take 0° as an example. Each positive example is required to be centered at the center point of fingerprint and its upright direction is required to be parallel with the direction of fingerprint. Negative examples are generated in the same way except that the reference poses (both centers and directions) are randomly produced. A constraint has been enforced on the pose of negative examples to guarantee that (1) the pose is sufficiently different from the true pose; and (2) negative examples should cover sufficient fingerprint foreground region. Fig. 4 gives positive and negative examples for classifiers of three directions. Both of positive and negative examples are of fixed size (640×576 pixels).

3.3. Feature extraction

As fingerprint pose can be inferred from ridge orientation field, we extract features as follows (See Fig. 5):

1. Divide the input image into small overlapping blocks of fixed size.
2. For each block, accumulate a local 1-D normalized histogram of ridge orientations over the block. Ridge orientation are computed with Sobel gradient operator.
3. To compute the feature vector of a sliding window, we combine all the histogram entries of the blocks in the window to form the feature vector.

Window size is of size 640×576 pixels and divided into 10×9 square blocks, with 16 pixels overlap between blocks. In each block, a 16 bins fuzzy orientation histogram is calculated as follows: each pixel votes to two adjacent bins, and the voting values

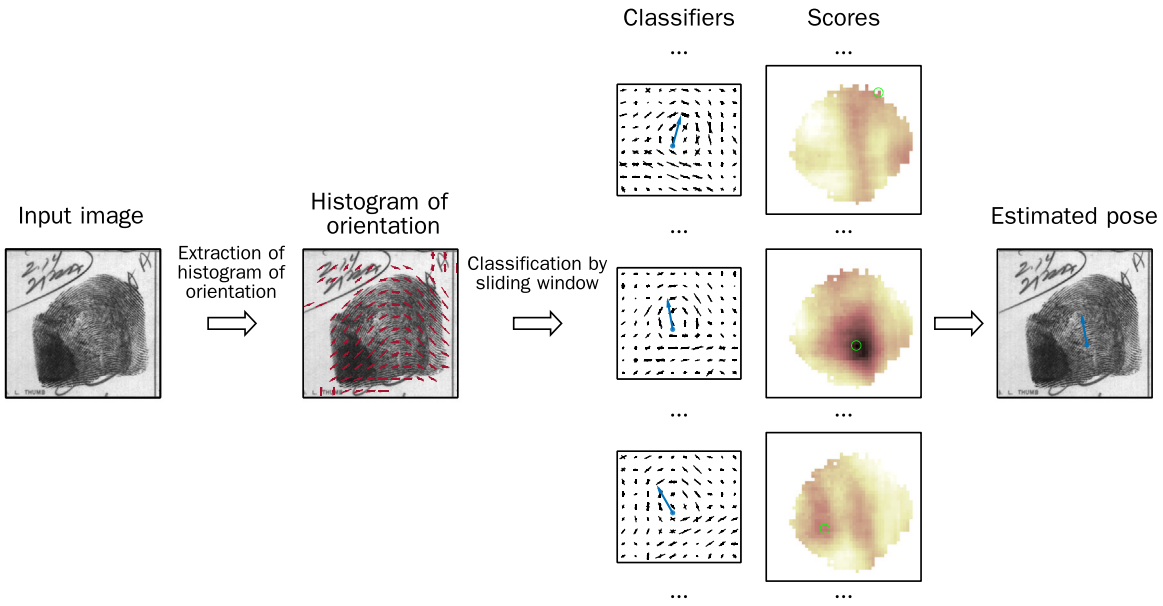

Fig. 2. Fingerprint pose estimation flowchart. After extracting histogram of orientation of each block in a fingerprint, a detection window is slid over the input fingerprint image. A set of 37 classifiers (in direction $[-90^\circ, -85^\circ, \ldots, 0^\circ, \ldots, 85^\circ, 90^\circ]$, only positive components are shown in figure) are used to estimate the possibility that there is a centered fingerprint with corresponding direction. Decision values of classifiers are shown as 37 images, where the pixel intensity is proportional to score (low intensity for high score). The center and direction of window with global highest score (the extrema of all classifiers in all locations) is regarded as the center and direction of fingerprint.

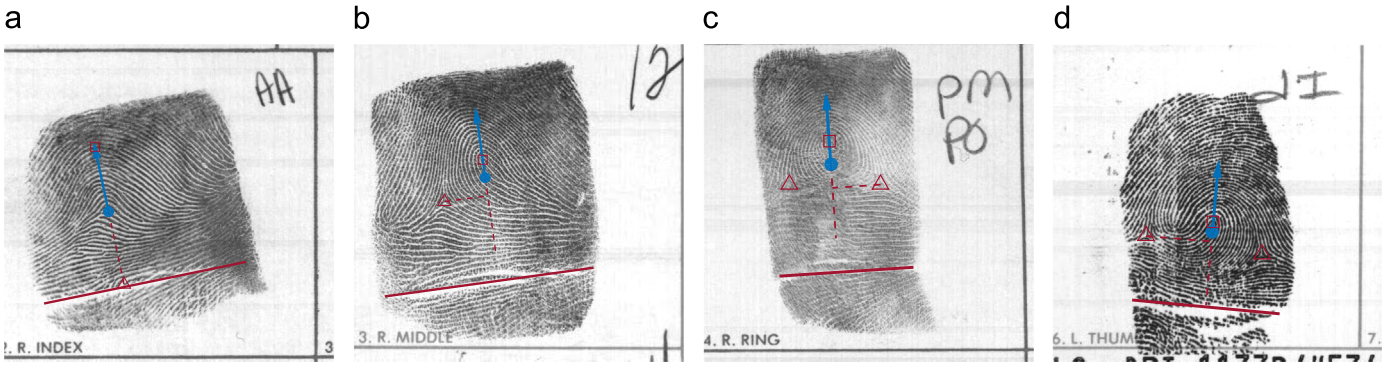

Fig. 3. Examples of manually marked pose, where the circle and arrow line specify center and direction, respectively. The square and triangle sign label the reference loop (or maximum curvature point) and delta (or projective point of maximum curvature point on the northernmost straight ridge), respectively. The red solid line labels the position of finger joint. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

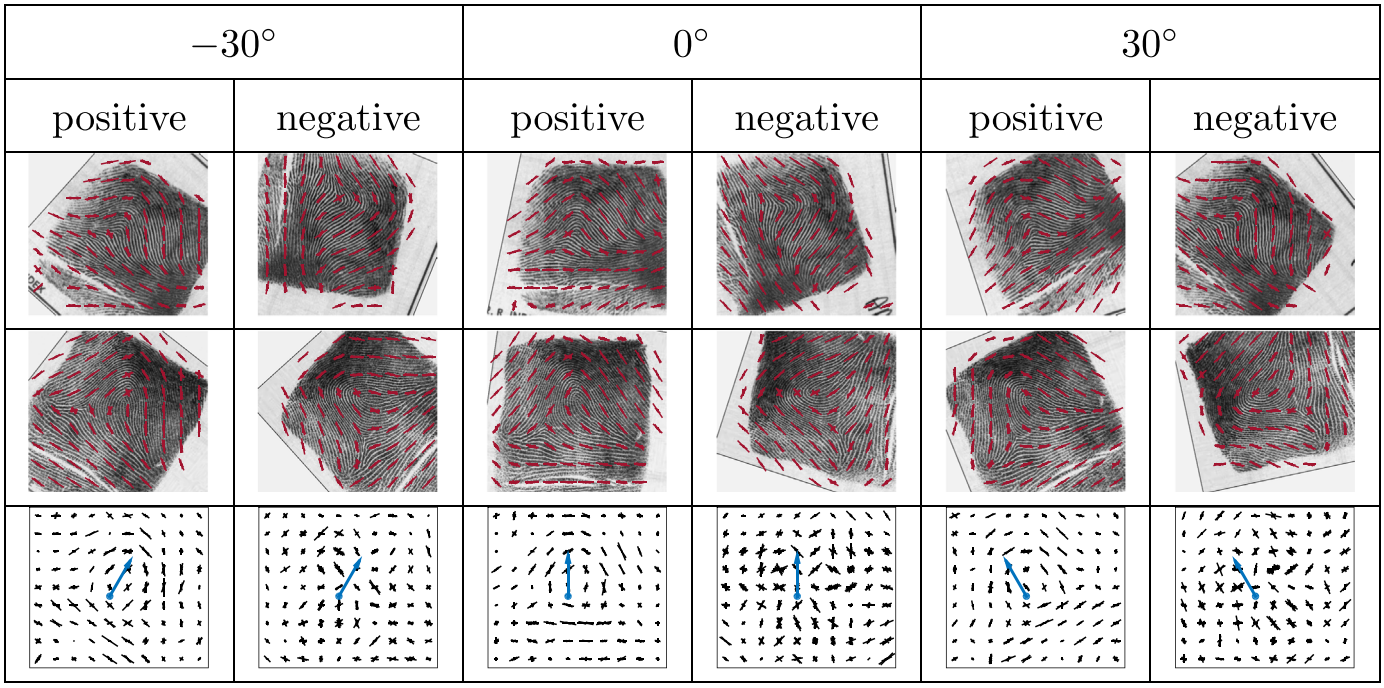

Fig. 4. Positive and negative training samples (only two are shown) for pose estimation and learned classifiers of three directions ($-30°$, $0°$, and $30°$). Computed features labeled by variable-length line segments are drawn over the samples. Positive and negative coefficients in weighted sum of SVM support vectors are separately drawn on the third row.

are inversely proportional to the angle between orientation of the pixel and the two bins, respectively.

Our feature vector is histograms of orientation instead of orientation field, since it is a richer representation. Analogous descriptors that capture edge feature with histogram have been proved to be effective in computer vision problems [22,23]. The computed feature vectors are shown in Figs. 2 and 4 as variable-length line segments centered on each block. The longer a line segment is, the more concentrated gradients in the block are on the direction perpendicular to it.

3.4. Classification

Features of training samples are fed to SVM classifiers of different directions. The classifiers are shown in Figs. 2 and 4. In Fig. 2, positive components of SVM weight vectors are rearranged into block feature format. In Fig. 4, positive and negative SVM weights are separately drawn. We can see that positive SVM weights are similar to feature of a whorl with one core and two delta, and negative SVM weights are perpendicular to positive ones.

Fig. 6 gives estimated poses of 12 fingerprints by two different algorithms: the algorithm of Yang et al. [21] and the proposed algorithm. These fingerprints are from 6 different fingerprints. Poses are expected to be consistent between different images of the same fingerprint. The algorithm by Yang et al. [21] estimates poses based on votes from every small ridge pattern block and thus it fails to capture global constraint. Our classifiers are learned from both positive and negative samples. In this way, we are able to learn a global pattern of fingerprints and separate positive and negative samples better.

4. Indexing

Performance of minutiae-based indexing approaches highly depend on mapping and matching among minutiae in fingerprints. Similarities among minutia descriptors are used to find out mated minutiae. Examples of descriptors include MCC [9], features of minutia triplets [10–12], and features of minutia quadruplets [13]. Although these descriptors are invariant to translation and rotation, descriptors of mated minutiae still would vary due to various kinds of noise.

MCC encodes neighborhood of a minutia by projecting its neighbor minutiae to a three dimensional space based on their relative locations and directions with center minutia. A discretization of the space into cells is then carried out to convert these projection values into a fixed length vector. Neighborhood of minutia includes more information of minutia than the other descriptors, making it comparatively easy to distinguish among minutiae. Thus, we use MCC as descriptor in the paper. Besides, the projection value is calculated by smoothing functions making the representation insensitive to small distortion and noise. However, the dimension of MCC is too high to be directly used in indexing which requires real-time operations.

Locality Sensitive Hashing (LSH) is effective and efficient in finding approximate nearest neighbor of high-dimensional vector [24,25]. In the basic LSH scheme, a family of functions $H = h_1, h_2, \ldots, h_l$ are used to project vectors into buckets. Near vectors are more likely to be projected into the same buckets by these functions. Finding approximate nearest neighbors could then be simplified to be finding vectors projected into the same buckets with the query one. In this scheme, construction of hash functions H is the most important part. For binary vectors, picking out a few bits from the original vectors is often used as the projection method. In Cappelli et al. [9], hash functions are randomly generated. However, spurious or missing minutiae are quite common especially in low quality or small fingerprints. For minutiae that are near low quality area, or around fingerprint edge, some bits in MCC features may be invalid or very noisy. Hash functions that are robust to noise and occlusion are thus desired.

4.1. Construction of hash functions

We use a group of hash functions that cover area of different size on neighborhood of minutia instead of randomly selection. Fig. 7 gives a few examples of such hash functions. Disks of cylinder are equally divided into 4 small regions. Hash function

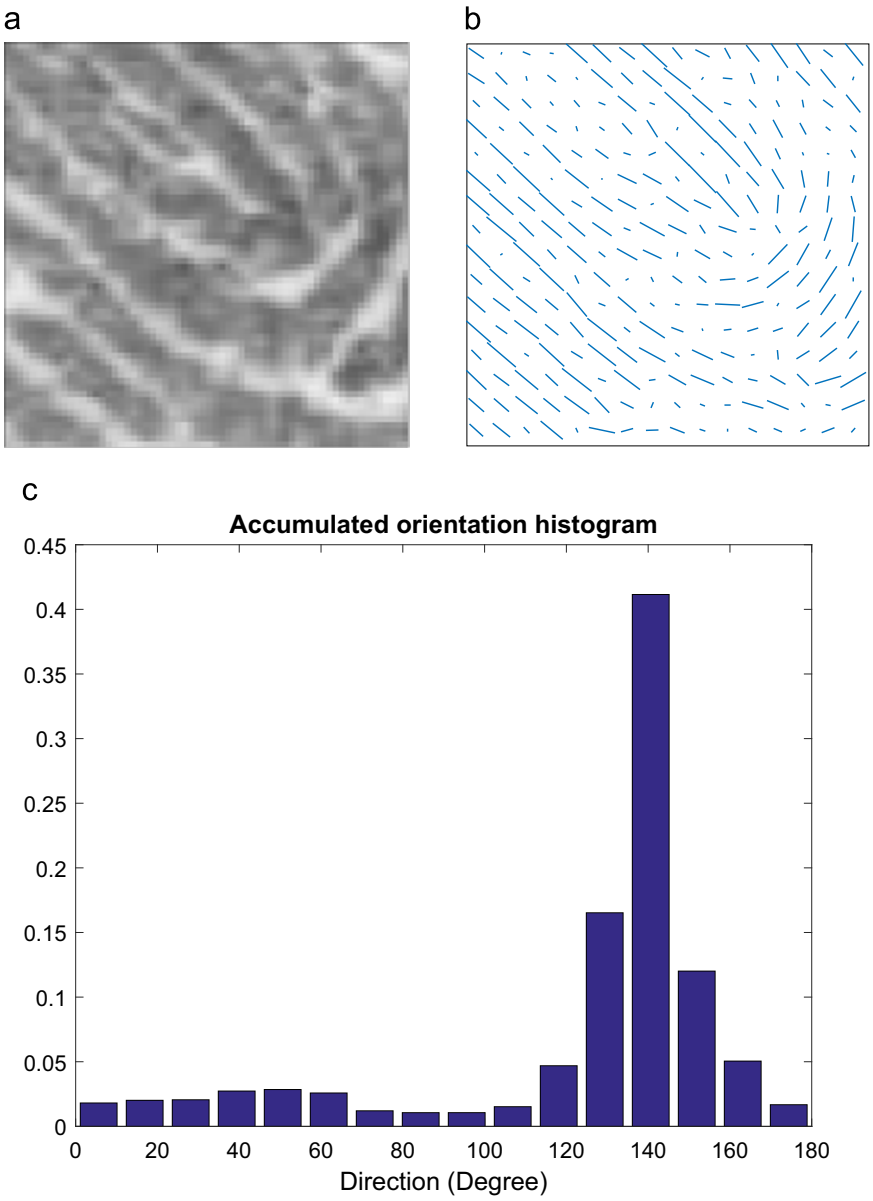

Fig. 5. Feature extraction procedure. (a) A block on input fingerprint. (b) Ridge orientations. Length of line segments are proportional to magnitude of local gradient. (c) Accumulated orientation histogram.

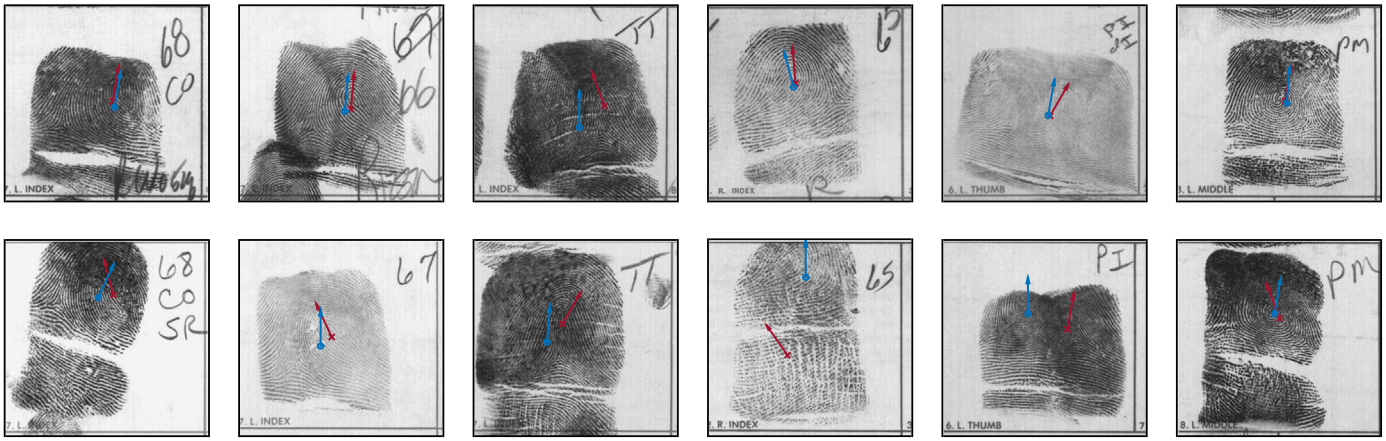

Fig. 6. Estimated poses by two algorithms for six pairs of fingerprints in NIST SD14. Impressions in each column are two different images of the same fingerprint. Poses estimated by the proposed algorithm are marked with blue circle and arrow, and poses estimated by Yang et al. [21] are marked with red cross and arrow. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

(a) covers the upper-left region, and selects bits only in this area. Hash function (b) covers two regions in the right side, so it selects bits randomly in this area. Let $H_{1/4}$ be the set of functions that cover a quarter of the disk, $H_{1/2}$ and H be the sets of functions that cover half of the disk and the whole disk, respectively. Each function randomly selects $N_D \cdot n$ bits, where N_D means the number of disks in a MCC.

Intuitively, when using the first hash function shown in Fig. 7, minutiae projected into the same bucket have similar neighborhood in the upper-left area. When using the last hash function, minutiae projected into the same bucket are roughly similar to each other in global area. Combination of such functions makes our algorithm more robust in dealing with various noise. See Fig. 8 for an example. Due to noise, there are one spurious minutia and one shifted minutia in the lower left part (Fig. 8(d)) in the query fingerprint, making much noise in the corresponding area in MCC. In this case, hash functions that select bits only in local area are more robust to noise than those that randomly select bits in the whole disk.

4.2. Searching

Inverted index table is a popular data structure in information retrieval which can significantly speed up the indexing process. In the offline stage, inverted index table is used to store all the invariant features of all gallery fingerprints. In the online stage, given a query fingerprint, for each of its invariant features, corresponding buckets in the inverted index tables are found and a vote is cast for each gallery fingerprint in the buckets. Finally, gallery fingerprints are ranked according to their received votes.

Our indexing algorithm is also built within this framework. The major steps of making inverted index table are:

1. Estimate pose of gallery fingerprint images.
2. Extract MCC features from gallery fingerprint image and use hash functions to generate index terms.
3. Construct inverted index table for each hash function.

With the proposed global registration strategy, when finding mated minutiae for query minutia, we only consider those minutiae that are stored under the same index terms as the query one, and are matchable with the query one. We call two minutiae matchable, if they are close to each other (less than e_l in location and e_θ in direction) after transformed into the coordinate system defined by estimated pose. To efficiently filter unmatchable minutiae, location and direction of minutia are stored in inverted index table together with its ID. Note that, these thresholds (e_l and e_θ) are set based on the accuracy of pose estimation algorithm. Besides, index terms with number of 1 bits small than $\min_{pc}$ are discarded when constructing inverted index tables as they are less able to discriminate minutiae.

The major steps of searching a fingerprint in inverted index table are described as follows:

1. Estimate pose of query fingerprint images.
2. Extract MCC features, transform them into index terms with hash functions.
3. For each minutia m_i in query fingerprint f_q, find mated minutiae $m_{g,j}$ in every gallery fingerprint f_g, and vote for corresponding gallery fingerprints according to their similarity $s_{i,g,j}$.
4. Score each gallery fingerprint by averaging over the votes from its minutia.

$$S_{q,g} = 1/n_g \times \sum_{i=1}^{n_q} \sum_{j=1}^{n_g} s_{i,g,j}, \tag{1}$$

where n_q, n_g mean the number of minutiae of query fingerprint f_q and gallery fingerprint f_g, respectively.

As we use a group of hash functions, some minutiae may collide with each other in multiple hash functions. We calculate similarity

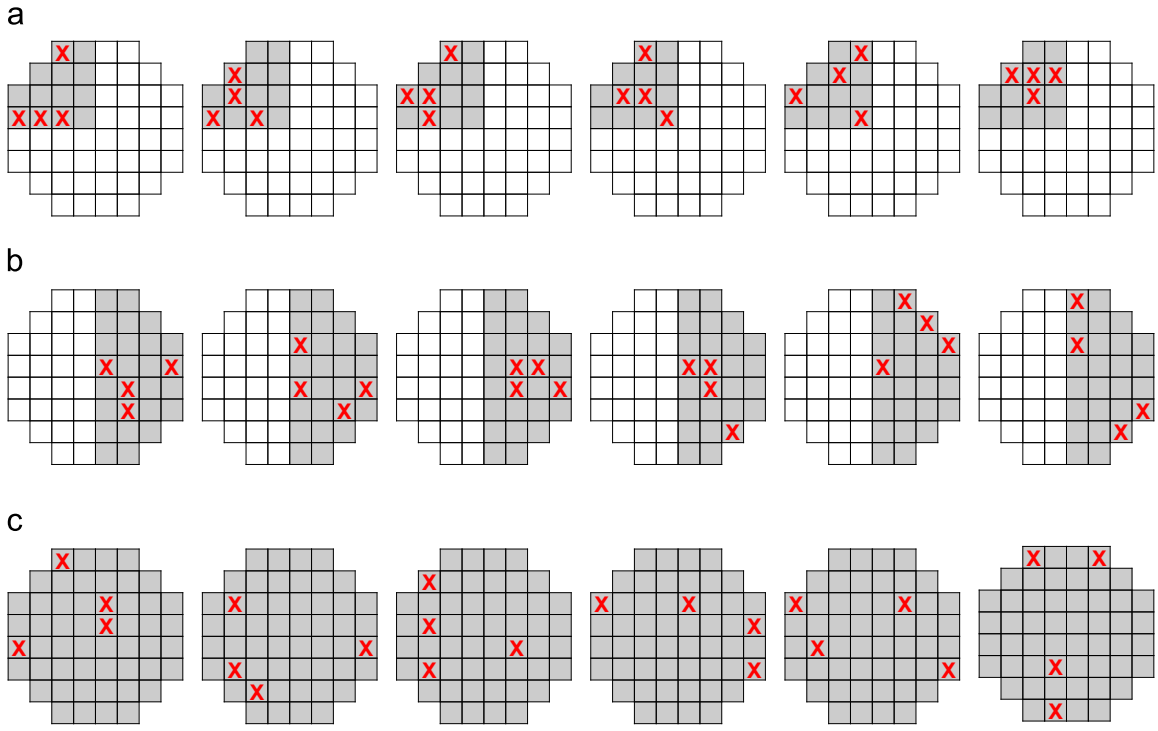

Fig. 7. Examples of selected bits of hash functions in a MCC code. (a) This hash function selects only bits in the upper-left area. (b) This hash function selects only bits in the right side. (c) This hash function selects bits randomly in the whole disk.

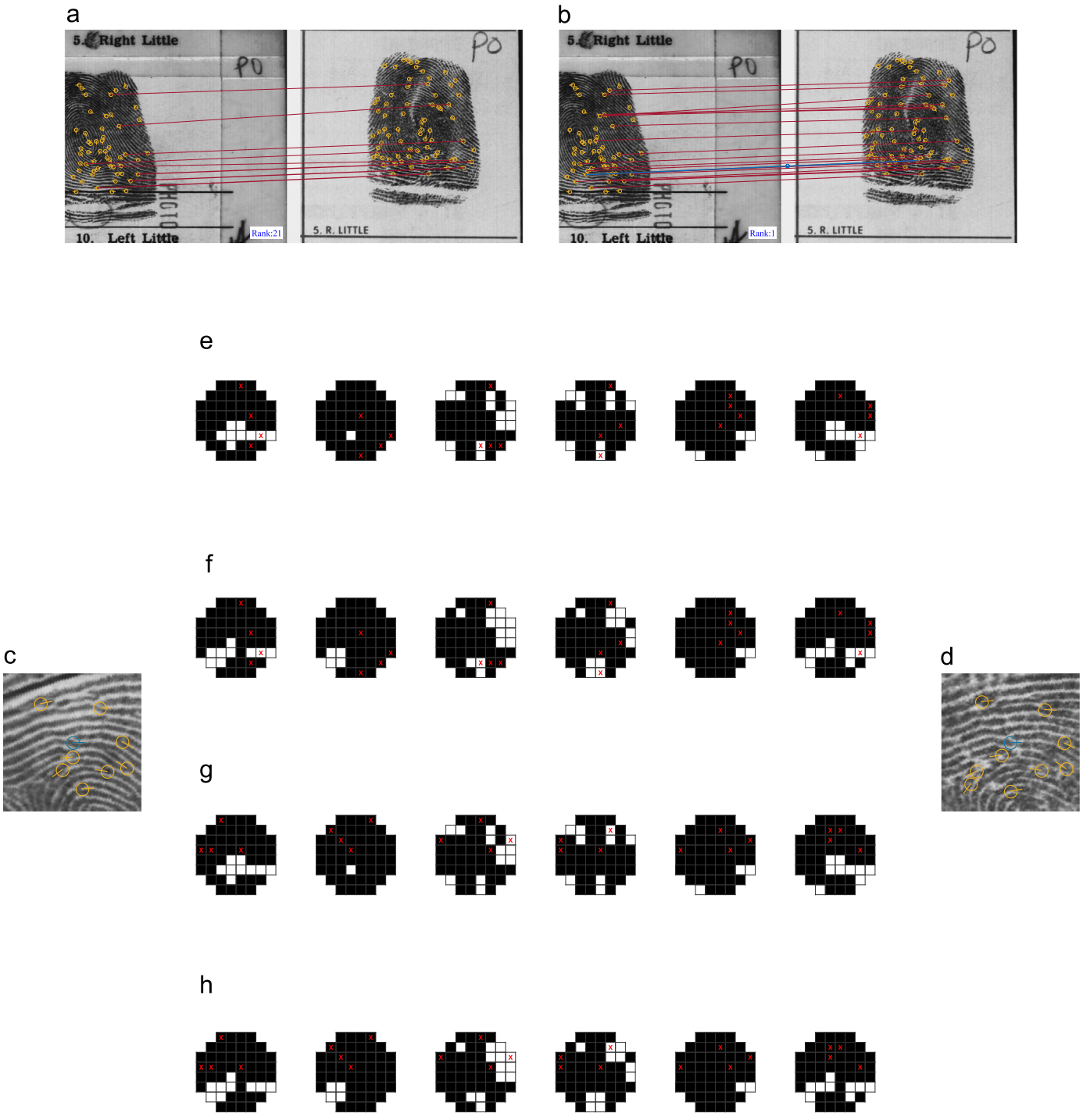

Fig. 8. Matched minutiae found by indexing algorithm with different hash functions. On the top row, matched minutiae are connected with lines. Width of line connecting matched minutiae is proportional to the times they projected into the same bucket. (a) Using a group of hash functions that randomly select bits from the whole disk. (b) Using a group of hash functions that cover area of different size. (c) and (d) show the local structures of minutiae connected by line with circle mark in (b), respectively. (e), (g) show MCC of the left minutia, and (f), (h) show MCC of the right minutia. Note that they are mapped into the same buckets by hash functions which selecting bits only in the right side area ((e), (f)) and the upper area ((g), (h)).

between two minutiae $s_{i,g,j}$ based on the collision times $t_{i,g,j}$:

$$s_{i,g,j} = t_{i,g,j}^{p/h}, \tag{2}$$

in which, h means the number of hash functions, and p is a parameter controlling the similarity function shape.

5. Experiment

5.1. Databases

To compare the performance of our algorithm with published work, several public domain fingerprint databases used in previous studies are used, including:

- FVC2000 DB2 Set A [26]: Containing 100 fingers and 8 impressions per finger (800 in total). Each fingerprint is captured by capacitive sensor "TouchChip" as plain impression.
- FVC2000 DB3 Set A [26]: Containing 100 fingers and 8 impressions per finger (800 in total). Each fingerprint is captured by optical sensor "DF-90" as plain impression.
- FVC2002 DB1 Set A [27]: containing 100 fingers and 8 impressions per finger (800 in total). Each fingerprint is captured by optical sensor "TouchView II" as plain impression.
- NIST Special Database 4 [28]: Containing 2000 fingers and 2 ("F" and "S") impressions per finger (4000 in total). Each fingerprint is rolled against a paper card and then scanned into digital form. The database is evenly distributed over different fingerprint pattern types (arch, left loop, right loop, tented arch, and whorl). The size of images is 512×480 pixels.
- NIST Special Database 14 [29]: Containing 27,000 fingers and 2 ("F" and "S") impressions per finger (54,000 in total). Each fingerprint is rolled against a paper card and then scanned into digital form. The size of images is 800×768 pixels.

In addition, a database of 1,000,000 rolled fingerprints from law enforcement agencies is used as the background database in our indexing experiments to make the experiments more realistic.

5.2. Performance indicators

5.2.1. Pose estimation

As we hope the estimated poses are consistent in different impressions of one fingerprint, we provide a method to measure how consistent they are. Since fingerprint center cannot be accurately marked, evaluating fingerprint pose based on manual marking is not very meaningful. Hence we estimate the pose deviation between different impressions of same fingerprint as follows:

1. Get mated minutia pairs between two fingerprints using a minutiae matcher,[1] as shown in Fig. 9(a).
2. Transform the coordinates of minutiae of each fingerprint into the coordinate system defined by its estimated fingerprint center and direction. The fingerprint center is defined as origin, the Y-axis is determined by the direction, and the X-axis is determined according to the right-hand rule. The transformed minutiae of two fingerprints are plotted in Fig. 9(b).
3. Calculate the mean direction deviation and the mean location deviation between mated minutia pairs. The location and direction deviation of the example illustrated in Fig. 9(b) is 46 pixels and 6.2°, respectively.

5.2.2. Indexing

Error rate and penetration rate are commonly used to measure the accuracy of fingerprint indexing approaches. The penetration rate is the proportion of retrieved fingerprints in the whole database. The error rate is defined as the proportion of query fingerprints whose mated fingerprints are not included in the retrieved fingerprints. An error-penetration curve is typically plotted by the trade-off between error rate and penetration rate as shown in Fig. 11. Moreover, execution time and memory requirement are another two important performance indicators for a fingerprint indexing algorithm.

[1] Cappelli et al. [30] described four matching algorithms to select potentially mated minutia pairs and then calculate similarity score of two fingerprints. We get the mated minutia pairs by taking the union of minutia pairs found by the third and fourth matching algorithms.

5.3. Evaluation of pose estimation

5.3.1. Evaluation setup and parameters

The first 1500 pairs of fingerprints in NIST SD14 are used to train fingerprint classifiers. For each direction, one positive and one negative image are cropped from every training fingerprint image.

5.3.2. Results

Fig. 10 shows empirical cumulative distribution function of location and direction deviation of mated minutiae on the last 2700 pairs of NIST SD14 before and after registration (with core points, our pose estimation algorithm and algorithm proposed by Yang et al. [21]). We use VeriFinger SDK 6.2 [31] to extract location and direction of core points. Among the last 2700 pairs of NIST SD14, 1915 pairs are able to be aligned with the core points (at least one core point are detect in both of the mated fingerprints). The deviations of minutiae on the other fingerprint pairs are directly evaluated without registration.

It can be seen that core point based registration even increases the direction deviation compared to without registration. That is because the direction of core point is usually determined by local orientation field which, however, changes rapidly around core point. Affected by the large direction deviation, location deviation is also very large. It is clear that both location and direction deviations become smaller after registration with our algorithm. Especially, location deviation is significantly reduced after registration. In contrast, when using algorithm proposed by Yang et al. [21], location deviation is reduced, but direction deviation becomes larger. The left region (small deviation) of the two plots in Fig. 10 is more important when we use tight pose constraints for minutiae matching, since the pose constraints should not affect most genuine matches. This indicates that we can use tighter constraints for minutiae matching when registered by our algorithm compared with by the algorithm of Yang et al. [21]. Tight spatial constraints are beneficial for both speed and accuracy of minutiae based indexing algorithms.

5.4. Evaluation of indexing

5.4.1. Evaluation setup and parameters

Location uncertainty and direction uncertainty are set as $e_l = 80$, $e_\theta = 20°$. We use $h = 32$ local hash functions with each selecting 24 ($n = 4, N_D = 6$) bits from MCC cylinder in indexing, including

- 8 quarter hash functions $H_{1/4}$, namely, two independent $H_{1/4}$ functions for each quarter of the cylinder disk;
- 12 half hash functions $H_{1/2}$, namely, 3 independent hash functions for each half disk on upper, below, left and right side; and
- 12 hash functions H covering the whole disk.

VeriFinger SDK 6.2 [31] is used to extract minutiae. Parameters for MCC creation and searching are the same as Cappelli et al. [9].

5.4.2. Results

On NIST SD14, the "F" impressions of the last 2700 pairs of fingerprints are used as gallery fingerprints, and the "S" impressions are used as query fingerprints. The trade-off curves between error rate and penetration rate for different indexing algorithms (Cappelli et al. [9] and proposed one) are shown in Fig. 11(a). We use MCC SDK (version 1.4) available online [32], and the same indexing parameters as [9].

For comparison, results of improving Cappelli et al. [9] by adding different pose constraints are also shown in Fig. 11(a). We transform minutiae into coordinate system defined by the estimated pose by two algorithms: the proposed algorithm and Yang et al. [21], and use two pose constraints separately. For loose

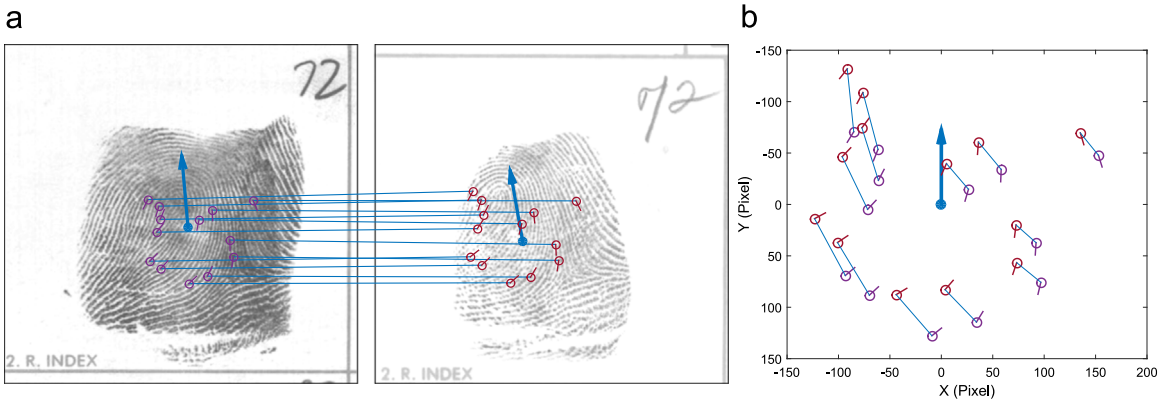

Fig. 9. An example showing how to measure pose deviation. (a) The estimated poses and mated minutia pairs (found by a minutiae matcher without using pose constraint) in two impressions of one fingerprint. (b) Mated minutia pairs transformed into the coordinate system defined by the estimated poses. The average location and direction deviation between these minutia pairs is about 46 pixels and 6.2°, respectively.

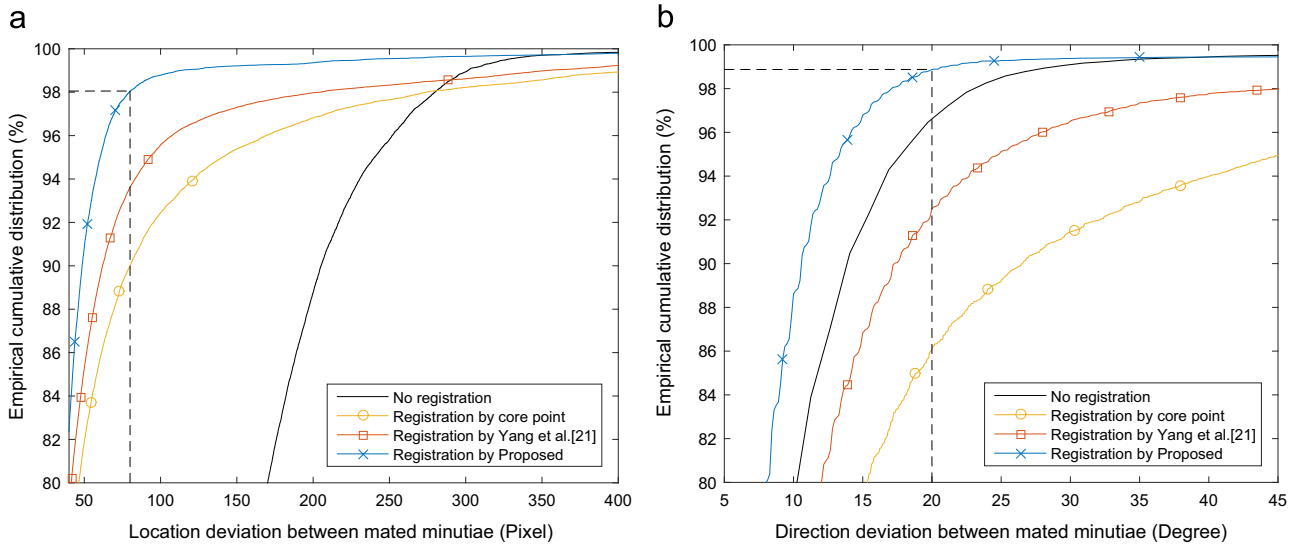

Fig. 10. Empirical cumulative distribution function of location (a) and direction (b) deviations between mated minutiae before and after registration. Three estimation approaches (core points, Hough transform based fingerprint pose estimation algorithm proposed by Yang et al. [21] and the proposed estimation algorithm with step of 16 pixels) are evaluated on the last 2700 pairs of fingerprints in NIST SD14. The left region (low deviation) is more relevant because tight spatial constraints are beneficial for both speed and accuracy of minutiae based indexing algorithms.

constraint we use $e_l = 256, e_\theta = 45°$, just the same as that in [9], and for tight constraint we use $e_l = 80, e_\theta = 20°$.

We can see that with poses estimated by Yang et al. [21] and loose constraint, indexing performance is better than without any registration. However, when applying tight constraint, the error rates are higher than our algorithm with loose constraint on high penetration rate. The reason is that poses estimated by Yang et al. [21] can remove some false mated minutiae, but since they are not that consistent, when applying tight constraint, some genuine mated minutiae are rejected, too. The proposed pose estimation algorithm outperforms previous ones markedly with both loose and tight constraints.

We also test the algorithm on NIST SD4 (Fig. 11(b)). The 2000 "F" impressions are used as gallery fingerprints, and the "S" impressions as query. The advantage of our algorithm is evident from the figure. The error rate is significantly smaller than Cappelli et al. [9] in both low and high values of penetration rate.

To test algorithm performance on larger database, we use a database of 1,000,000 rolled fingerprints as the background database. The result in Fig. 11(c) confirms the scalability of our algorithm. Note that size of fingerprint images in the background database are 640×640 pixels, different from that of images in NIST SD14. It makes no difference to our algorithm, as minutiae are all transformed to a united coordinate system when indexing. But for algorithm by Cappelli et al. [9], the constraints on mated minutiae may affect indexing performance, as no registration has been performed. So we augmented fingerprint images in background database with blank border to make them as big as fingerprint images in NIST SD14 when testing algorithm by Cappelli et al. [9].

As for plain fingerprints, small foreground area and large out-of-plane rotation makes it far more challenging to keep estimated pose consistent among different impressions (as shown in Fig. 12). Thus, the proposed pose estimation algorithm wasn't used in experiments on plain fingerprints. While, with the proposed local hash functions, indexing performances on FVC2000 DB2 (Fig. 11(d)), FVC2000 DB3 (Fig. 11(e)) and FVC2002 DB1 (Fig. 11(f)) are better than previous state of the art [9]. In these experiments, the first impressions are selected as gallery fingerprints, and the remaining 7 impressions are used as query fingerprints, following the evaluation protocol in [9].

The proposed indexing algorithm is highly dependent on accuracy of estimated fingerprint pose. Fig. 13 shows an example where the estimated poses of query fingerprint and gallery fingerprint are inconsistent due to poor image quality around the central area. Most of the mated minutiae are determined as unmatchable after being

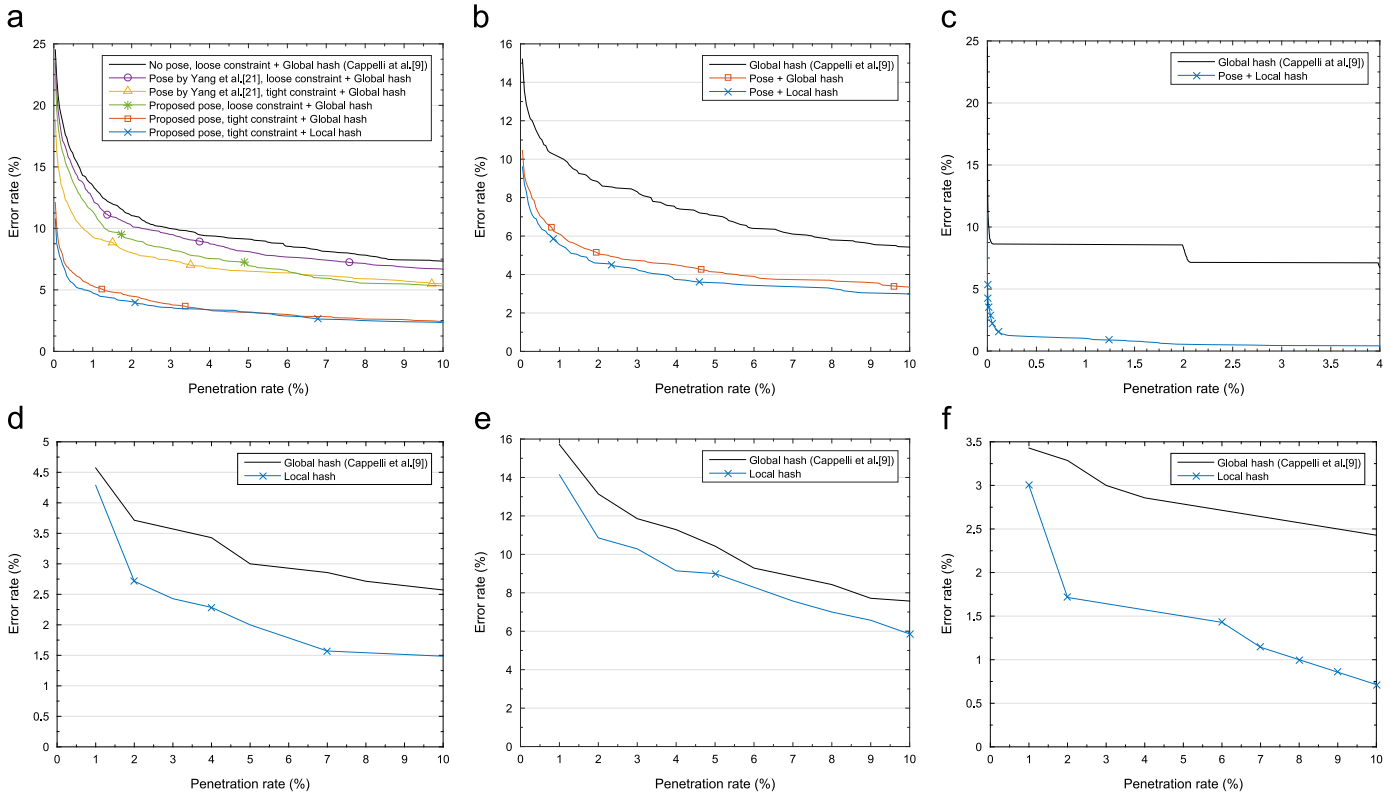

Fig. 11. Indexing performance on several fingerprint databases. (a) Last 2700 pairs of NIST SD14. (b) NIST SD4. (c) Last 2700 pairs of NIST SD14 and large background database (containing 1,000,000 rolled fingerprints). (d) FVC2000 DB2A. (e) FVC2000 DB3A. (f) FVC2002 DB1A. "Global hash" means hash functions covering the whole disk in MCC. "Local hash" means to use the proposed local hash functions in our paper.

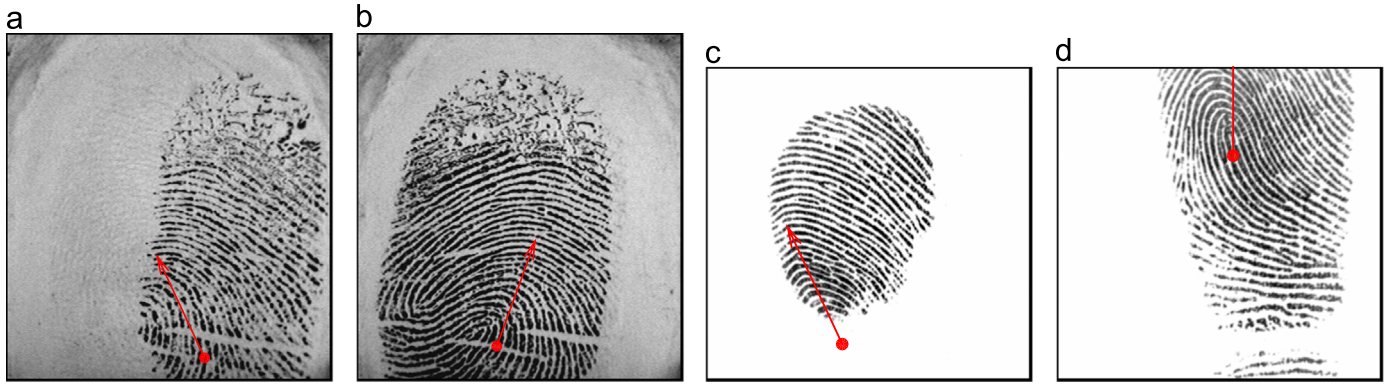

Fig. 12. Challenges for consistent pose estimation in plain fingerprints with small common area. (a) and (b) are from one fingerprint. (c) and (d) are from another fingerprint.

transformed into the coordinate system defined by estimated poses. As a result, the matching score of the mated gallery fingerprint is very low.

Table 2 shows the single threaded execution times and memory consumptions of proposed algorithm and Cappelli et al. [9]. The program is tested on a 2.5 GHz Intel Xeon CPU. Implemented with Matlab, the pose estimation program takes 754 ms to process one fingerprint image of size 800×768 pixels with step of 16 pixels (in direction $[-90^\circ, -85^\circ, \ldots, 0^\circ, \ldots, 85^\circ, 90^\circ]$). The indexing program is implemented with C++ without special optimization. For each fingerprint image, pose estimation algorithm needs to be performed only once. With GPU and proper optimization, the time required will be further reduced. With the proposed pose estimation algorithm, we use tight constraints in selecting mated minutiae, which dramatically reduces the number of false mated minutiae, and thus greatly improves the speed of the indexing program. Besides, coordinates of minutiae are scaled to be stored in 1 Byte (i.e. costs 2 Bytes to store X and Y values), which helps to reduce memory consumptions.

6. Conclusion

In this paper, we proposed a learning-based fingerprint pose estimation algorithm. The estimated poses are quite consistent among different impressions of roll fingerprint which makes it possible to align fingerprints in an efficient way. With estimated pose, we proposed a minutiae-based indexing algorithm which refine the matched minutiae with global spatial constraint made by poses. The experimental results on public domain fingerprint databases show

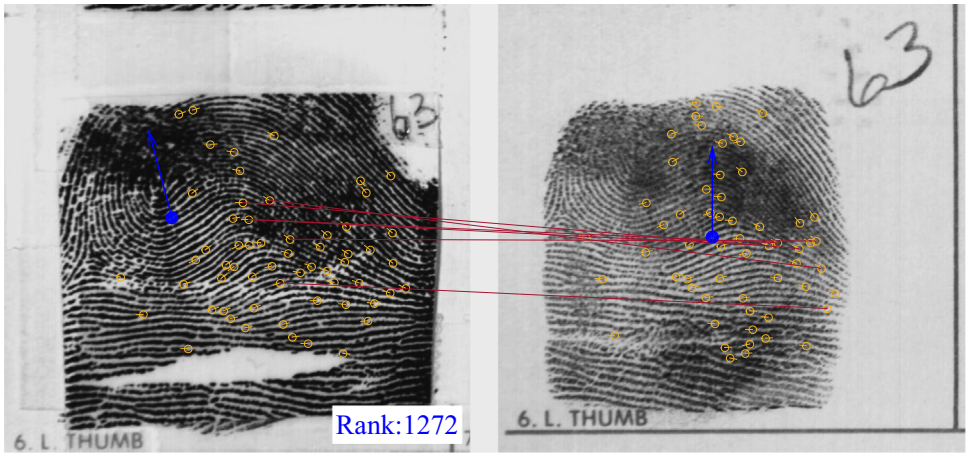

Fig. 13. A failure example of the proposed algorithm due to inconsistent poses. The fingerprint in the right side is query fingerprint, the left one is its mated gallery fingerprint. A tight constraint (80 pixels and 20°) is posed on minutiae to justify whether they are matchable. Center point and direction of fingerprint are marked by circle and arrow, respectively. Matched minutiae are connected by lines. The ranks of gallery fingerprints are annotated on the bottom right corner of the image.

Table 2
Execution time and memory of the proposed algorithm on NIST SD14 without/with background fingerprints.

Algorithm[a]	Adding a template (ms)	Searching (ms)	Memory cost (Byte)
Cappelli et al. [9][b]	34.9/36.8	$141/5.49 \times 10^4$	400M/46.1G
Proposed	3.51/8.69	$13.9/4.43 \times 10^3$	300M/40.8G

[a] The database sizes of without/with background fingerprints are 2700 and 1,002,700 respectively.

[b] According to our experiments using MCC SDK (of version 1.4) available online [32].

that the proposed approach performs better than the state-of-the-art algorithm. Besides, we proposed an improved version of Locality Sensitive Hashing algorithm for MCC descriptor which is shown to be useful for improving accuracy further. Experiment in large scale database containing one million rolled fingerprints from real applications also confirmed the scalability of our algorithm.

However, the proposed algorithm can be still improved by: (1) speeding up the pose estimation algorithm with GPU; (2) extending the pose estimation algorithm for plain fingerprints especially those with small foreground area and large out-of-plane rotations; (3) developing a quality evaluation algorithm that can determine uncertainty of estimated pose, which can be used to adaptively adjust tolerance of location and direction deviation between minutia pairs in fingerprint indexing.

Conflict of interest

No conflict of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants 61225008, 61572271, 61527808, 61373074 and 61373090, the National Basic Research Program of China under Grant 2014CB349304, the Ministry of Education of China under Grant 20120002110033, and the Tsinghua University Initiative Scientific Research Program.

References

[1] D. Maltoni, D. Maio, A.K. Jain, S. Prabhakar, Handbook of Fingerprint Recognition, Springer, London, UK, 2009.
[2] D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J. Benitez, Fast fingerprint identification for large databases, Pattern Recognit. 47 (2) (2014) 588–602.
[3] M. Liu, X. Jiang, A.C. Kot, Efficient fingerprint search based on database clustering, Pattern Recognit. 40 (6) (2007) 1793–1803.
[4] X. Jiang, M. Liu, A.C. Kot, Fingerprint retrieval for identification, IEEE Trans. Inf. Forensics Secur. 1 (4) (2007) 532–542.
[5] Y. Wang, J. Hu, D. Phillips, 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 (4) (2007) 573–585.
[6] M. Liu, P.T. Yap, Invariant representation of orientation fields for fingerprint indexing, Pattern Recognit. 45 (7) (2012) 2532–2542.
[7] R. Cappelli, M. Ferrara, A fingerprint retrieval system based on level-1 and level-2 features, Expert Syst. Appl. 39 (12) (2012) 10465–10478.
[8] A.A. Paulino, E. Liu, K. Cao, A.K. Jain, Latent fingerprint indexing: fusion of level 1 and level 2 features, in: IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS), IEEE, Arlington, VA, USA, 2013, pp. 1–8.
[9] R. Cappelli, M. Ferrara, D. Maltoni, Fingerprint indexing based on minutia cylinder-code, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2011) 1051–1057.
[10] R.S. Germain, A. Califano, S. Colville, Fingerprint matching using transformation parameter clustering, IEEE Comput. Sci. Eng. Mag. 4 (4) (1997) 42–49.
[11] B. Bhanu, X. Tan, Fingerprint indexing based on novel features of minutiae triplets, IEEE Trans. Pattern Anal. Mach. Intell. 25 (5) (2003) 616–622.
[12] X. Liang, A. Bishnu, T. Asano, A robust fingerprint indexing scheme using minutia neighborhood structure and low-order Delaunay triangles, IEEE Trans. Inf. Forensics Secur. 2 (4) (2007) 721–733.
[13] O. Iloanusi, A. Gyaourova, A. Ross, Indexing fingerprints using minutiae quadruplets, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop on Biometrics, IEEE, Colorado Springs, CO, US, 2011, pp. 127–133.
[14] O.N. Iloanusi, Fusion of finger types for fingerprint indexing using minutiae quadruplets, Pattern Recognit. Lett. 38 (2014) 8–14.
[15] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, vol. 1, Cambridge University Press, Cambridge, UK, 2008.
[16] J.H. Wegstein, An Automated Fingerprint Identification System, US Department of Commerce, National Bureau of Standards, 1982.
[17] S. Yoon, J. Feng, A.K. Jain, On latent fingerprint enhancement, in: SPIE Defense, Security, and Sensing, International Society for Optics and Photonics, 7767 (2010) 395–407.
[18] M. Liu, X. Jiang, A.C. Kot, Fingerprint reference-point detection, EURASIP J. Adv. Signal Process. 2005 (4) (2005) 498–509.
[19] K. Rerkrai, V. Areekul, A new reference point for fingerprint recognition, in: International Conference on Image Processing, vol. 2, IEEE, Vancouver, BC, CA, 2000, pp. 499–502.
[20] T.H. Le, H.T. Van, Fingerprint reference point detection for image retrieval based on symmetry and variation, Pattern Recognit. 45 (9) (2012) 3360–3372.
[21] X. Yang, J. Feng, J. Zhou, Localized dictionaries based orientation field estimation for latent fingerprints, IEEE Trans. Pattern Anal. Mach. Intell. 36 (5) (2014) 955–969.
[22] D.G. Lowe, Object recognition from local scale-invariant features, in: IEEE International Conference on Computer Vision, vol. 2, IEEE, Kerkyra, Greece, 1999, pp. 1150–1157.
[23] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, IEEE, San Diego, CA, US, 2005, pp. 886–893.

[24] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA, 1998, pp. 604–613.
[25] A. Gionis, P. Indyk, R. Motwani, et al., Similarity search in high dimensions via hashing, in: VLDB, vol. 99, 1999, pp. 518–529.
[26] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, A.K. Jain, FVC2000: fingerprint verification competition, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 402–412.
[27] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, A.K. Jain, FVC2002: second fingerprint verification competition, Int. Conf. Pattern Recognit. 3 (2002) 811–814.
[28] C. Watson, C. Wilson, NIST Special Database 4, Fingerprint Database, vol. 17, National Institute of Standards and Technology. ⟨http://www.nist.gov/srd/nistsd4.cfm⟩.
[29] C. Watson, NIST Special Database 14, Fingerprint Database, National Institute of Standards and Technology. ⟨http://www.nist.gov/srd/nistsd14.cfm⟩.
[30] R. Cappelli, M. Ferrara, D. Maltoni, Minutia cylinder-code: a new representation and matching technique for fingerprint recognition, IEEE Trans. Pattern Anal. Mach. Intell. 32 (12) (2010) 2128–2141.
[31] Neurotechnology Incorporated, VeriFinger SDK (2015). URL ⟨http://www.neurotechnology.com/verifinger.html⟩.
[32] R. Cappelli, M. Ferrara, D. Maltoni, Minutia cylinder-code SDK (2014), URL ⟨http://biolab.csr.unibo.it/⟩.

Yijing Su received the B.S. degree from Beijing University of Posts & Telecommunications, Beijing, China, in 2010. Since 2010, she has been working toward the Ph.D. degree in the Department of Automation, Tsinghua University. Her research interests include fingerprint indexing, pattern recognition and computer vision.

Jianjiang Feng is an associate professor in the Department of Automation at Tsinghua University, Beijing. He received the B.S. and Ph.D. degrees from the School of Telecommunication Engineering, Beijing University of Posts and Telecommunications, China, in 2000 and 2007, respectively. From 2008 to 2009, he was a post doctoral researcher in the PRIP lab at Michigan State University. He is an associate editor of Image and Vision Computing. His research interests include fingerprint recognition and computer vision.

Jie Zhou was born in November 1968. He received B.S. degree and M.S. degree both from Department of Mathematics, Nankai University, Tianjin, China, in 1990 and 1992, respectively. He received Ph.D. degree from Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology (HUST), Wuhan, China, in 1995. From then to 1997, he served as a postdoctoral fellow in Department of Automation, Tsinghua University, Beijing, China. From 2003, he has been a full professor in Department of Automation, Tsinghua University. His research area includes computer vision, pattern recognition and image processing. In recent years, he has authored more than 100 papers in peer-reviewed journals and conferences. Among them, more than 30 papers have been published in top journals and conferences such as PAMI, T-IP and CVPR. He is an associate editor for International Journal of Robotics and Automation, Acta Automatica and two other journals. Zhou is a recipient of the National Outstanding Youth Foundation of China.