Fingerprint Indexing and Matching: An Integrated Approach

Kai Cao and Anil K. Jain
Department of Computer Science and Engineering
Michigan State University, East Lansing, Michigan 48824
{kaicao,jain}@cse.msu.edu

Abstract

Large scale fingerprint recognition systems have been deployed worldwide not only in law enforcement but also in many civilian applications. Thus, it is of great value o identify a query fingerprint in a large background fingerprint database both effectively and efficiently based on indexing strategies. The published indexing algorithms do not meet the requirements, especially at low penetrate rates, because of the difficulty in extracting reliable minutiae and other features in low quality fingerprint images. We propose a Convolutional Neural Network (ConvNet) based fingerprint indexing algorithm. An orientation field dictionary is learned to align fingerprints in a unified coordinate system and a large longitudinal fingerprint database, where each finger has multiple impressions over time, is used to train the ConvNet. Experimental results on NIST SD4 and NIST SD14 show that the proposed approach outperforms state-of-the-art fingerprint indexing techniques reported in the literature. Further indexing results on an augmented gallery set of 250K rolled prints demonstrate the scalability of the proposed algorithm. At a penetrate rate of 1%, a score-level fusion of the proposed indexing and a state-of-the-art COTS SDK provides 97.8% rank-1 identification accuracy with a 100-fold reduction in the search space.

1. Introduction

Fingerprints are one of the most important biometric traits to identify individuals due to their perceived uniqueness and persistence of friction ridge patterns [13]. With decades of research and development, large scale fingerprint recognition systems have been deployed worldwide not only in law enforcement and forensic agencies but also in numerous civilian applications. For example, the FBI's Next Generation Identification (NGI) database, one of the world's largest law enforcement database, allows federal and state agencies to search more than 70 million civil fingerprints submitted for background checks alongside another 50 million or so

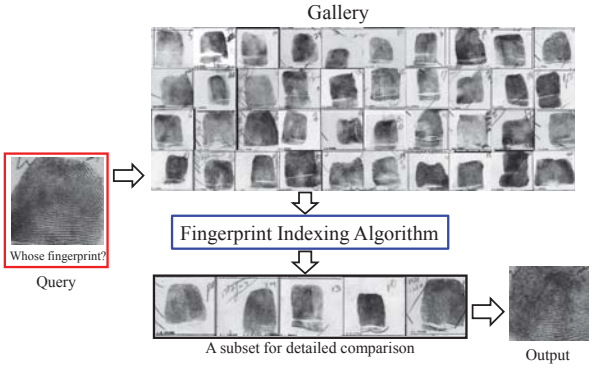


Figure 1: Fingerprint matching framework.

prints submitted for criminal investigations[1]. Representative examples of civilian applications include (i) the OBIM (formerly the US-VISIT) program by the Department of Homeland Security [1] and (ii) India's Aadhar project [2], which is now the largest biometrics deployment in the world with an enrollment that already exceeds 1.1 billion tenprints (along with corresponding irises and photos) of supposedly unique individuals[2]. For the task of identifying a fingerprint in such massive scale applications, both high identification accuracy and high search efficiency are critical. Fingerprint indexing is necessary to quickly locate a subset of candidates from the large background database, followed by a detailed comparison of the query with the subset of fingerprints to give the final output, as shown in Fig. 1. However, an efficient retrieval of a candidate list should ensure that the true mate of the query is indeed present in the candidate list. This challenging problem continues to be of significant interest to biometrics community.

The purported uniqueness of fingerprints is characterized by three levels of features: (i) level-1 features, such as pattern type, orientation field and ridge frequency field; (ii) level 2 features, e.g., minutiae and (iii) level 3 features which includes attributes at a very-fine scale, such as ridge shape,

[1] https://theintercept.com/2017/02/04/the-fbi-is-building-a-national-watchlist-that-gives-companies-real-time-updates-on-employees/

[2] https://uidai.gov.in/about-uidai/about-uidai.html

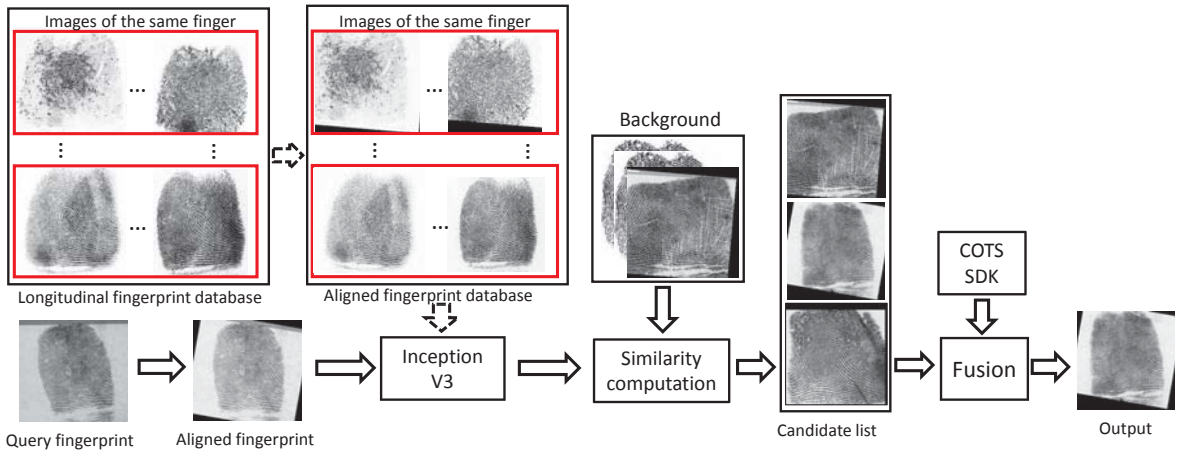


Figure 2: Overview of the proposed fingerprint indexing and matching algorithm. Dotted arrows denote the offline learning process while the solid arrows denote the online indexing process.

pores and incipient ridges. Given the difficulty of automatic extraction of level-3 features [8], fingerprint indexing approaches are primarily based on level-1 and level-2 features. Level-1 feature based indexing approaches [11, 9] typically align the fingerprint images based on reference points (e.g. core or maximum curvature point) or orientation field and then extract a fixed length feature vector of orientation field or ridge frequency for indexing. While level-1 indexing is faster compared to level-2 indexing, they have two main limitations: (i) the alignment is not robust and (ii) the indexing is not very effective due to low discriminability of level-1 features [6].

Minutiae based fingerprint indexing algorithms [6, 15, 3, 7, 19] generally extract a set of rotation and translation invariant features from minutiae points and use hashing techniques for indexing. Examples of the invariant features include geometric features from minutiae triplets [3] and minutiae quadruplets [7], and minutiae descriptors [6]. Table 1 summarizes these approaches. However, without any global constraints, local minutiae structures in the query fingerprint may lead to high similarity to non-mated gallery fingerprints. To alleviate this, fusion of level-1 and level-2 features for indexing has been used, e.g., [14], [5], to boost the indexing effectiveness. The minutiae-based fingerprint indexing approaches also have their own limitations: (i) it is difficult to extract fixed-length feature vector for indexing, (ii) indexing performance relies on accurate minutiae extraction which is difficult for poor quality fingerprints, and (iii) indexing may lead to some loss in identification accuracy. None of the indexing approaches in the literature are able to maintain identification accuracy at low penetration rate.

To address the limitations of level-1 and level-2 indexing, we propose a convolutional neural network (ConvNet) based fingerprint indexing. In order to align fingerprints into a unified coordinate system, a fingerprint orientation field dictionary is learned for fingerprint alignment. A longitudinal fingerprint database [21] (a total of 440K fingerprints from 37,410 different fingers) is used to train a ConvNet. The output of the last fully connected layer of the ConvNet is used as the fixed-length feature vector for indexing. This approach has the following advantages: (i) the fixed-length feature vector is efficient for indexing and, at the same time, useful for template protection and (ii) minutiae extraction is not needed for indexing. The proposed indexing algorithm outperforms state-of-the-art algorithms on two public domain databases, i.e., NIST SD4 and NIST SD14. Furthermore, we show that the identification accuracy is boosted by score level fusion of the proposed indexing and a COTS fingerprint SDK.

The main contributions of this paper are as follows:

1. Designed a fingerprint indexing algorithm by leveraging a large longitudinal fingerprint database to train a ConvNet to generate a fixed-length feature vector is efficient for indexing. The proposed algorithm does not explicitly extract fingerprint minutiae.

2. Developed a robust approach for aligning fingerprints into a unified coordinate system.

3. Demonstrated superior indexing performance on two different benchmark databases over state-of-the-art algorithms. By fusing the indexing scores and scores from a COTS fingerprint SDK, both the identification accuracy and search efficiency are improved.

Table 1: A summary of studies on fingerprint indexing for rolled prints*.

Algorithm	Fingerprint Features	Approach	Fingerprint Database**	ER@ PR=1%	ER@ PR=5%	ER@ PR=10%
Bhanu and Tan [3]	Minutiae	Triplets	NIST SD4	26%	16.5%	14.5%
Jiang et al. [9]	OF + RF	OF Clustering	NIST SD4	n.a.	13.7%	10.5%
Liu and Yap [12]	OF	Polar Complex Moments	NIST SD4	34%	20%	12%
Cappelli et al. [6]	Minutiae	MCC descriptor	NIST SD4	6.8%	4.7%	3.9%
			NIST SD14	8.9%	6.7%	4.9%
Cappelli & Ferrara [5]	Minutiae+OF	MCC descriptor	NIST SD4	9.5%	1.5%	0.8%
			NIST SD14	10.4%	2.1%	1.2%
Su et al. [15]	Minutiae+Pose	MCC descriptor + pose constraints	NIST SD4	5.2%	3.5%	2.97%
			NIST SD14	4.8%	3.0%	2.33%
Proposed***	Texture	ConvNet	NIST SD4	1.35%	0.75%	0.40%
			NIST SD14	1.07%	0.26%	0.19%

OF: orientation field; RF: ridge frequency field; MCC: minutiae cylinder code; ER: error rate; PR: penetrate rate.
* Some of the numbers shown in the table were estimated from the plots in the corresponding papers.
** All 2,000 fingerprint pairs in NIST SD4 are included while only the last 2,700 fingerprint pairs in NIST SD14 are included.
*** The performance reported here is just for fingerprint indexing without fusion with AFIS.

2. Fingerprint Indexing Algorithm

Fig. 2 outlines the proposed fingerprint indexing algorithm which consists of offline learning and online fingerprint indexing modules.

2.1. Offline Learning and Template Generation

2.1.1 Learning Orientation field Dictionary

(a) (b) (c)

Figure 3: Illustrating fingerprint image preprocessing. (a) Input fingerprint image, (b) contrast enhanced fingerprint image and (c) ROI.

Alignment is crucial to extract transformation invariant features for fingerprint indexing. In order to align the training fingerprints into a unified coordinate system, we learn a fingerprint orientation field dictionary by manually marking the orientation of the finger joint and automatically detect the reference point[3]. 2,000 high quality rolled fingerprints (NFIQ[4] ≤ 2) from 2,000 different fingers are selected from the large longitudinal fingerprint database used in [21] to learn an orientation field dictionary for fingerprint alignment. Given a fingerprint image I, it is first contrast enhanced using the following procedure:

$$I^*(i,j) = \frac{I(i,j) - meanI(i,j)}{varI(i,j)}, \quad (1)$$

where I^* is the contrast-enhanced image, $meanI$ and $varI$ are local pixel-wise mean and variation images, respectively. The gradient magnitude of I^* is used for region of interest (ROI) segmentation to extract the foreground consisting of friction ridge pattern. See Fig. 3. The ConvNet-based approach [4] is applied to I^* to compute the fingerprint orientation field with block size of 16×16 pixels. Fig. 4 (b) shows the orientation field within ROI of an input fingerprint shown in Fig. 4 (a). For training data, the orientation of finger joint ($\Delta\theta$) is manually marked and reference point (x, y) is detected automatically using the approach in [20]. The orientation field O is aligned to ensure orientation of finger joint is horizontal (rotation by $-\Delta\theta$) and the reference point is located at $x_0 = 16$ and $y_0 = 15$ on a target orientation field consisting of 32×32 blocks, as shown in Fig. 4 (d). Formally, the aligned orientation field $O_{x,y,-\Delta\theta}$ is computed as follows:

$$O_{x,y,-\Delta\theta}(p,q) = \Phi(O(i,j) - \Delta\theta), \quad (2)$$

[3]Fingerprint reference point is defined as the point with the maximum curvature on the convex ridge

[4]NFIQ ranges from 1 to 5, with 1 indicating the highest quality and 5 indicating the lowest quality fingerprint [17].

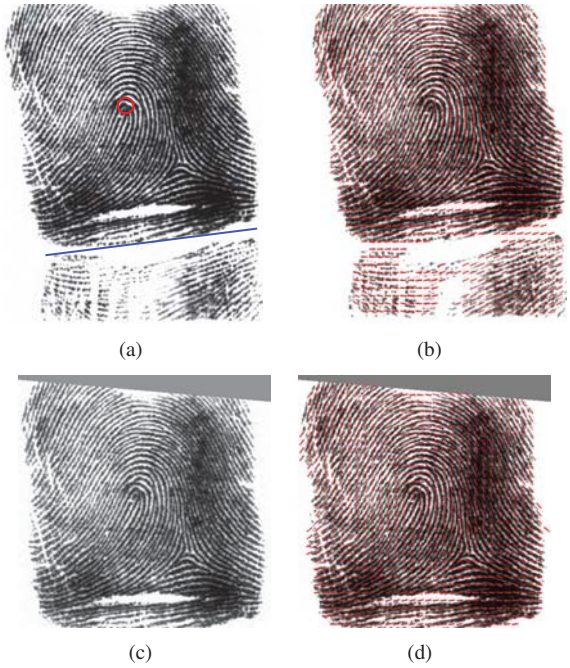


Figure 4: Illustrating fingerprint alignment for training dataset for learning orientation field dictionary. (a) Automatically extracted reference point and manually marked orientation of finger joint, and (c) aligned fingerprint based on reference point and orientation of finger joint, (b) and (d) show orientation fields of (a) and (b), respectively. The images in (a) and (b) are 700×700 and the images in (c) and (d) are 512×512 . The block size for orientation field extraction is 16×16.

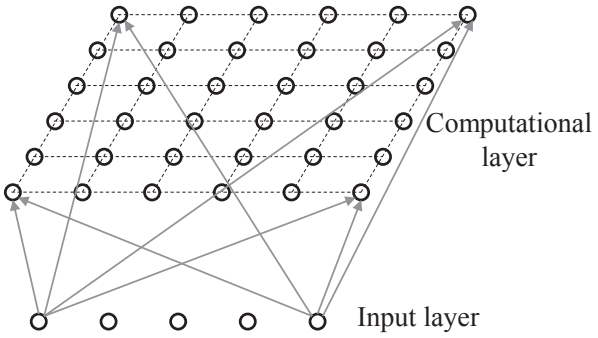


Figure 5: Architecture of Kohonen network [10] used for orientation field dictionary learning. There are 6 rows and 6 columns in the computational layer, which results in a total of 36 cluster centers. The inputs to the network are 2,048-dimentional vectors ($32 \times 32 \times 2$) by concatenating cosine and sine components of $32 \times 32 \times 2$ orientation field blocks.

where

$$q = (j - x)\cos(-\Delta\theta) - (i - y)\sin(-\Delta\theta) + x_0, \quad (3)$$
$$p = (i - y)\cos(-\Delta\theta) + (j - x)\sin(-\Delta\theta) + y_0, \quad (4)$$

and $\Phi(\cdot)$ is defined as

$$\Phi(\alpha) = \begin{cases} \alpha - \pi & \text{if } \alpha \geq \pi/2, \\ \alpha + \pi & \text{if } \alpha < -\pi/2, \\ \alpha & \text{otherwise.} \end{cases} \quad (5)$$

Given an aligned orientation field, O, with 32×32 block size, $[\cos 2O, \sin 2O]$ values are concatenated for every element of the 32x32 block to form a 2,048-dimensional vector. With the 2,000 aligned orientation fields, the Kohonen network [10], a self-organizing map (SOM) with a single computational layer arranged in rows and columns (Fig. 5), is adopted to learn an orientation field dictionary, $D = \{D_1, D_2, ..., D_{36}\}$. Fig. 6 shows all 36 learned dictionary elements along with their singular points (core and delta), if present. The five fingerprint types are easily identified among the learned dictionary elements.

2.1.2 Fingerprint Alignment

Given a fingerprint I, its ROI R, orientation field O within R and reference point (x, y) are computed following the procedures in section 2.1.1. Given the aligned orientation field dictionary elements and the reference point of the input fingerprint, rotation $\Delta\theta$ is the only parameter to be determined. The optimal value $\Delta\theta_{opt}$ is the value which minimizes the difference between $O_{x,y,\Delta\theta}$ and one of the orientation field dictionary elements. Formally,

$$\Delta\theta_{opt} = \arg\min_{\Delta\theta}\{\min_{k} \mathsf{Diff}(O_{x,y,\Delta\theta}, D_k)\} \quad (6)$$

where $O_{x,y,\Delta\theta}$ is as defined in E.q (2) and $\mathsf{Diff}(O_{x,y,\Delta\theta}, D_k)$ is the dissimilarity between $O_{x,y,\Delta\theta}$ and orientation field dictionary element D_k defined as follows:

$$\mathsf{Diff}(O_{x,y,\Delta\theta}, D_k) = \sum_{(i,j)\in R_{x,y,\Delta\theta}} d(i,j) \quad (7)$$

$$d(i,j) = \sin^2(O_{x,y,\Delta\theta}(i,j) - D_k(i,j)), \quad (8)$$

where $R_{x,y,\Delta\theta}$ is the aligned ROI based on Eqs. (3) and (4). For each orientation field dictionary element D_k, a reference point-based Hough transform (**Algorithm** 1) is proposed to find the best rotation $\Delta\theta_k$ between O and D_k and the dissimilarity is evaluated using Eq. (8). The rotation with the smallest dissimilarity is selected as the optimal $\Delta\theta_{opt}$, i.e., $\Delta\theta_{opt} = \arg\min_{\Delta\theta_k} \mathsf{Diff}(O_{x,y,\Delta\theta_k}, D_k)$. The fingerprint image I is then rotated by $\Delta\theta_{opt}$ and translated to ensure reference point coincides with (x_0, Y_0).

2.1.3 Training ConvNet

In order to train a powerful ConvNet for fingerprint indexing, multiple fingerprint impressions of the same finger are needed, which is similar to training a ConvNet for face search [18]. For this purpose, we select 3,741 subjects with more than 12 impressions from MSP longitudi-

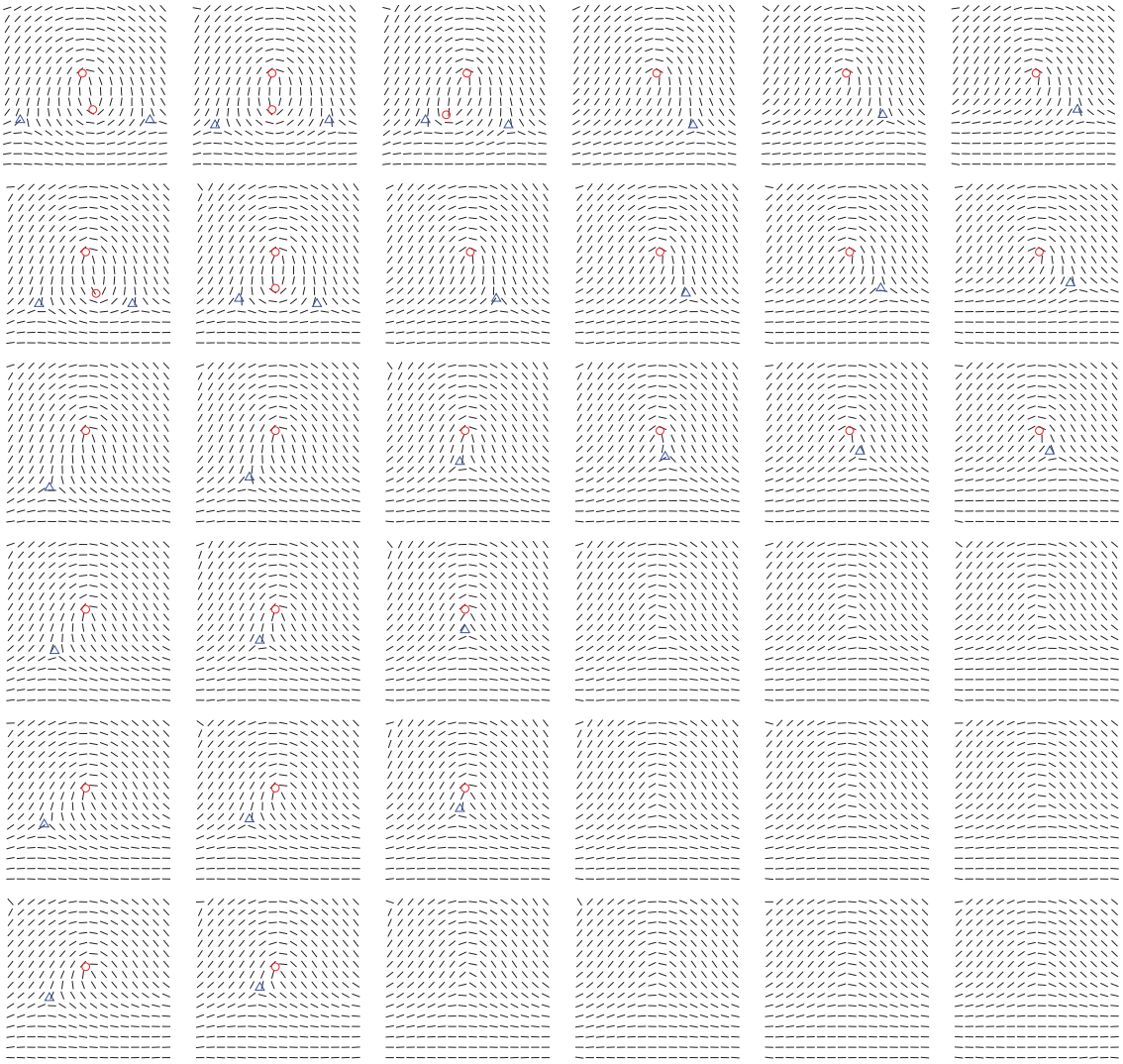


Figure 6: Orientation field dictionary with 36 elements, learned from 2,000 good quality rolled prints with NFIQ ≤ 2. Each orientation is 32×32 pixels shown as a block. Red circles denote the core points and the blue triangles denote delta points.

nal fingerprint database [21], which results in around 440K fingerprints from 37,410 different fingers, i.e., classes, for training the ConvNet from scratch. As far as we know, MSP longitudinal fingerprint database is the largest longitudinal fingerprint database in literature. After aligning all training fingerprints using the approach in section 2.1.2, a ConvNet is trained to group the fingerprints by fingers. A number of ConvNet architectures, such as AlexNet, VGG, GoogLeNet, and Inception V2-V3, have been proposed in literature for image and object recognition. Among these architectures, Inception V3 [16] not only achieves a very good recognition performance but also improves the efficiency by replacing two-dimensional filters by two one-dimensional filters. In this paper, we adopt Tensorflow based Inception V3[5] implementation. The 512×512 fingerprint images are first downsampled by a factor of 0.65 to 333×333 pixels and then randomly cropped to 299×299 pixels which is the input image size of Inception V3; the cropping is needed to accommodate the potential localization errors in reference point extraction. A factor of 0.65 is selected for downsampling because we need to cover most of the friction ridges in a 299×299 region. Data augmentation techniques, such as random contrast and random brightness, are adopted to ensure the model is robust to different variations in fingerprint images.

[5]https://github.com/tensorflow/models/tree/master/slim/nets

Algorithm 1 Reference point-based Hough transform

Input: Ridge flow O, ROI R and reference point (x, y), kth orientation field dictionary element D_k
Output: Rotation $\Delta\theta_k$
Initialize the accumulator array A
for all block pairs in $(i, j, \alpha) \in O$ and $(m, n, \beta) \in D_k$ **do**
    Compute p and q using Eqs. (3) and (4)
    if abs$(q - n) > 5$ or abs$(p - m) > 5$ **then**
        Continue
    $\Delta\gamma = \Phi(\beta - \alpha)$
    if abs$(\Delta\gamma) > \pi/2$ **then**
        $\Delta\gamma = \Phi(\Delta\gamma - \pi)$
    $A(\Delta\gamma) = A(\Delta\gamma) + 1$
Smooth A using a Gaussian low-pass filter with $\sigma = 1$
$\Delta\theta_k = \arg\max A$

2.1.4 Template Generation

For each fingerprint image in the gallery, it is first aligned as described in section 2.1.2 and then down sampled by a factor of 0.65 for reasons mentioned earlier. The central 299×299 region is input to the trained ConvNet and the output of the last fully connected layer, which is a 2,048-dimensional feature vector, is considered as fingerprint's representation. For fast computation of cosine distance used for similarity, the representation is normalized with unit length. Given a gallery G with N fingerprints, it is represented as $G = \{g_i | i = 1, ..., N\}$, where g_i is a 2,048-dimensional feature vector for the ith gallery fingerprint.

2.2. Online Indexing

Given a query fingerprint Q, its 2,048-dimensional representation q is extracted in a manner similar to gallery fingerprints. The indexing similarity score between q and $g_i \in G$, $S(q, g_i)$, is computed as the inner product of q and g_i since both q and g_i have the unit length. Formally,

$$S(q, g_i) = q^T \cdot g_i. \tag{9}$$

Finding the set of top-k most similar fingerprints $C_k(f)$ in the gallery G is then formulated as:

$$C_k(f) = \text{Rank}_k(\{S(q, g_i) | g_i \in G\}), \tag{10}$$

where $\text{Rank}_k(\cdot)$ is a function that finds the top-k largest values in a set.

3. Experimental Results

This section describes the database and experiments carried out to evaluate the proposed indexing algorithm and to compare it with the state-of-the-art approaches in the literature.

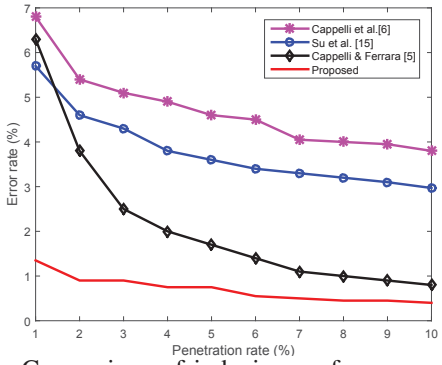


Figure 7: Comparison of indexing performance on NIST SD4. 2,000 "F" fingerprints are used as gallery and the remaining 2,000 "S" fingerprints are used as query.

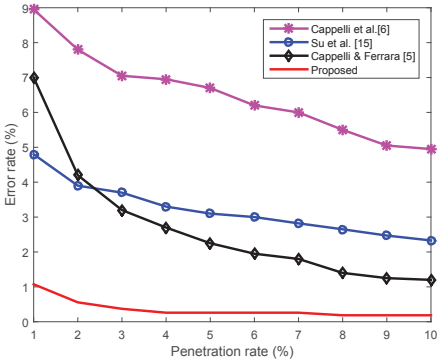


Figure 8: Comparison of indexing performance on last 2,700 fingerprint pairs (24301-27000) from NIST SD14 used in previous studies [6], [5] and [15] .

3.1. Datasets

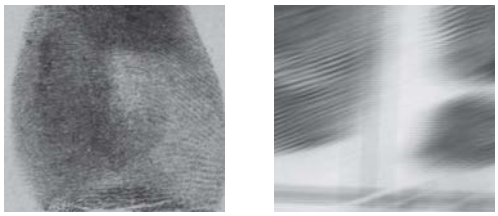
The experiments are conducted on the following two public domain fingerprint datasets:

- NIST SD4[6]: Contains 2,000 distinct fingers with 2 rolled impressions ("F" and "S") per finger (4,000 images in total). The database is evenly distributed over five fingerprint pattern types, i.e., arch, left loop, right loop, tented arch and whorl with 800 images/class.

- NIST SD14[7]: Contains 27,000 distinct fingers with 2 rolled impressions ("F" and "S") per finger (54,000 images in total).

In addition, 250,000 rolled fingerprints from a forensic agency are used to enlarge the gallery fingerprint database. A large gallery database allows us to investigate indexing efficacy and efficiency with increasing gallery size.

[6]http://www.nist.gov/srd/nistsd4.cfm
[7]http://www.nist.gov/srd/nistsd14.cfm



(a) (b)

Figure 9: Two poor quality fingerprints from NIST SD4. (a) F0427 and (b) F1192.

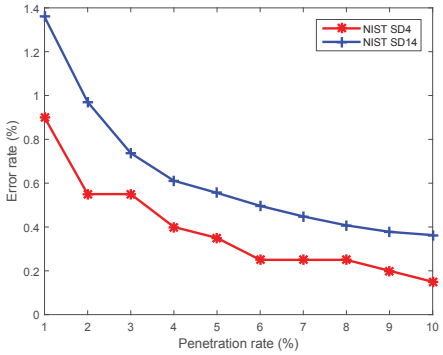


Figure 10: Indexing performance of the proposed approach on NIST SD4 and NIST SD14 when the gallery was augmented with 250K rolled images.

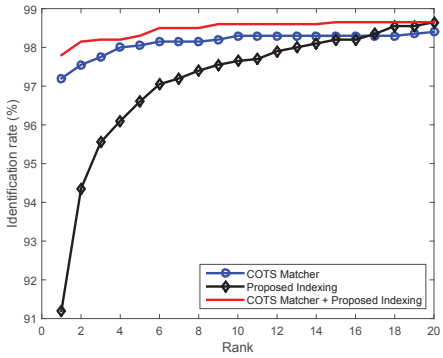


Figure 11: Identification performance by score-level fusion of a COTS SDK and the proposed indexing on NIST SD4.

3.2. Indexing Performance

For a fair comparison with other approaches reported in literature, we first report the indexing performance on each of these two databases without using additional rolled prints to enlarge the background database size. For experiments with NIST SD4, The 2,000 "F" impressions are used as gallery fingerprints, and the 2,000 "S" impressions as query. For experiments with NIST SD14, the "F" and "S" impressions of the last 2,700 pairs of fingerprints are used as gallery fingerprints and query fingerprints, respectively, as used in previous studies [6], [5], [15]. Indexing performance is usually measured by plotting the error rate vs. penetration rate. The error rate at a given penetration rate $p\%$ refers to the fraction of query prints for which the correct mates could not be retrieved within $p\%$ of the background database (penetration rate). Figs. 7 and 8 report the trade-off curves between error rate and penetration rate for NIST SD4 and NIST SD14, respectively. Note that the error rates reported in [6], [5] and [15] are estimated from the plots in their papers. Figs. 7 and 8 show that the proposed ConvNet-based fingerprint indexing algorithm outperforms results reported in [6], [5] and [15] significantly, especially at low penetrate rates. On NIST SD4, our indexing system could not find the mates of only 8 fingerprints at penetrate rate of 10%. These errors are primarily due to the poor quality of query or gallery fingerprints as shown in Fig. 9. Similar observations can be made for NIST SD14.

Table 2: Error rate at a fixed penetration rate of 5% with different size, N, of augmented database. The error rate does not increase, even drops on NIST SD4, because more fingerprints are included in top 5% as the N increases.

Database	N				
	$0K$	$10K$	$50K$	$100K$	$250K$
NIST SD4	0.75%	0.40%	0.35%	0.35%	0.35%
NIST SD14	0.56%	0.54%	0.55%	0.55%	0.56%

In order to test the indexing performance on larger background database, we included additional 250,000 rolled fingerprints in the background database. The results in Fig. 10 and Table 2 confirm the scalability of our algorithm. Note that entire NIST SD14 including 27,000 fingerprint pairs is used for this experiments.

3.3. Fusion of Indexing and Comparison

To test if the proposed indexing algorithm can maintain or even boost the identification performance of a state-of-the-art fingerprint SDK at a low penetrate rate, we use a COTS SDK. Given a query fingerprint image, top 1% of background database are retrieved using the proposed indexing algorithm. The COTS SDK is then used for one-to-one comparison. The similarity scores output by the COTS SDK and indexing similarity scores (Eq. (9)) are, respectively, normalized to [0,1] using min-max normalization and these scores are fused using a weighted sum rule. Due the higher recognition performance of the COTS SDK, the weights assigned to the COTS SDK scores and indexing scores are empirically determined as 0.7 and 0.3, respectively. Fig. 11 compares the Cumulative Match Curve (CMC) curves of COTS SDK and the fusion of COTS SDK, indexing score, and proposed indexing algorithm on NIST SD4 using "F" impressions as gallery and "S" impressions as query. Experiments on a large background database are pending and we will update the results once the paper is accepted. It is evident that the proposed indexing algo-

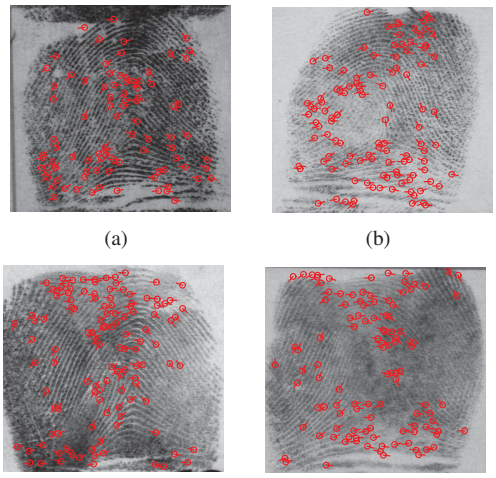


Figure 12: Two fingerprint pairs ((a) and (c); (b) and (d)) from NIST SD4 along with the minutiae sets extracted by the COTS SDK. The file names of (a), (b), (c) and (d) are S1174, S1869, F1174 and F1869, respectively. The COTS SDK could not find the mates of (a) and (b) at rank-1 because it could not extract sufficient number of corresponding minutiae for comparison. However, the score-level fusion of the COTS SDK and proposed indexing algorithm was successful in finding the true mates of (a) and (c) (respectively, (b) and (d)) at rank-1.

rithm can improve not only the identification efficiency (100 times) but also the identification accuracy (from 97.2% to 97.8% at rank 1). Fig. 12 shows two fingerprint pairs for which the COTS SDK could not find the mates of the queries at rank-1 but the fusion could. The main reason is that the COTS SDK could not extract sufficient number of reliable minutiae for comparison.

3.4. Computational Time

The fingerprint alignment and indexing were implemented on a server with 12 cores @ 2.50 GHz, 256 GB RAM running Linux O.S using Matlab 2014b. The ConvNet training and feature extraction using trained ConvNet were conducted on a desktop with i7-6700k @ 4.00 GHz, 32 GB RAM, GeForce GTX 1080, using Tensorflow[8]. For the fingerprint alignment, 24 threads (Matlab function: *parpool*) were used and the average computational time is 203 milliseconds per fingerprint. The average computational time for feature extraction from the aligned fingerprints is about 74ms. The average indexing time for a query against the 250K background is about 300 milliseconds. The template size per fingerprint image is 8KB.

[8]https://www.tensorflow.org/

4. Conclusions

Fingerprint indexing, despite tremendous progress in processor technologies, is crucial for efficient search of growing size of fingerprint databases in a variety of identification applications. The state-of-the-art fingerprint indexing algorithms are primarily based on minutiae which are not robust to poor quality fingerprints. Furthermore, it i difficult to extract a fixed-length representation from minutiae set for efficiently indexing. We have proposed a ConvNet based fingerprint indexing approach by learning an orientation field dictionary for fingerprint alignment; the ConvNet is trained on a large longitudinal fingerprint database. The performance of the proposed indexing algorithm on two rolled fingerprint databases demonstrates that it outperforms the state-of-the-art indexing algorithms, especially at low penetrate rates. Our work can be extended in the following directions: (i) improving the speed of fingerprint alignment, (ii) investigating different ConvNet architectures and loss functions to improve the indexing accuracy, and (iii) evaluating and improving the indexing efficiency and accuracy on larger background databases.

References

[1] Office of Biometric Identity Management. http://bias.dhs.gov/obim.

[2] Unique Identification Authority of India. http://uidai.gov.in/.

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

[4] K. Cao and A. K. Jain. Latent orientation field estimation via convolutional neural network. In *ICB*, pages 349–356, 2015.

[5] R. Cappelli and M. Ferrara. A fingerprint retrieval system based on level-1 and level-2 features. *Expert Systems with Applications*, 39(12):10465 – 10478, 2012.

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

[7] O. Iloanusi, A. Gyaourova, and A. Ross. Indexing fingerprints using minutiae quadruplets. In *CVPR 2011 WORKSHOPS*, pages 127–133, June 2011.

[8] A. K. Jain, Y. Chen, and M. Demirkus. Pores and ridges: High-resolution fingerprint matching using level 3 features. *IEEE TPAMI*, 29(1):15–27, 2007.

[9] X. Jiang, M. Liu, and A. C. Kot. Fingerprint retrieval for identification. *IEEE TIFS*, 1(4):532–542, Dec 2006.

[10] T. Kohonen. *Self-Organization and Associative Memory*. Springer, 1989.

[11] M. Liu, X. Jiang, and A. C. Kot. Fingerprint retrieval by complex filter responses. In *18th ICPR*, volume 1, pages 1042–1042, 2006.

[12] M. Liu and P.-T. Yap. Invariant representation of orientation fields for fingerprint indexing. *Pattern Recognition*, 45(7):2532 – 2542, 2012.

[13] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Second Edition. Springer, 2009.

[14] A. A. Paulino, E. Liu, K. Cao, and A. K. Jain. Latent fingerprint indexing: Fusion of level 1 and level 2 features. In *IEEE BTAS*, pages 1–8, 2013.

[15] Y. Su, J. Feng, and J. Zhou. Fingerprint indexing with pose constraint. *Pattern Recognition*, 54:1 – 13, 2016.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv*, 2015.

[17] E. Tabassi, C. Wilson, and C. Watson. Fingerprint image quality. *NISTIR 7151*, 2004.

[18] D. Wang, C. Otto, and A. K. Jain. Face search at scale. *IEEE TPAMI*, PP(99):1–1, 2016.

[19] Y. Wang, L. Wang, Y. M. Cheung, and P. C. Yuen. Learning compact binary codes for hash-based fingerprint indexing. *IEEE TIFS*, 10(8):1603–1616, Aug 2015.

[20] S. Yoon, K. Cao, E. Liu, and A. K. Jain. LFIQ: Latent fingerprint image quality. In *IEEE BTAS*, pages 1–8, Sept 2013.

[21] S. Yoon and A. K. Jain. Longitudinal study of fingerprint recognition. *Proceedings of the National Academy of Sciences*, 112(28):8555–8560, 2015.