Learning Compact Binary Codes for Hash-Based Fingerprint Indexing

Yi Wang, *Member, IEEE,* Lipeng Wang, Yiu-Ming Cheung, *Senior Member, IEEE,* and Pong C. Yuen

***Abstract*— Compact binary codes can in general improve the speed of searches in large-scale applications. Although fingerprint retrieval was studied extensively with real-valued features, only few strategies are available for search in Hamming space. In this paper, we propose a theoretical framework for systematically learning compact binary hash codes and develop an integrative approach to hash-based fingerprint indexing. Specifically, we build on the popular minutiae cylinder code (MCC) and are inspired by observing that the MCC bit-based representation is bit-correlated. Accordingly, we apply the theory of Markov random field to model bit correlations in MCC. This enables us to learn hash bits from a generalized linear model whose maximum likelihood estimates can be conveniently obtained using established algorithms. We further design a hierarchical fingerprint indexing scheme for binary hash codes. Under the new framework, the code length can be significantly reduced from 384 to 24 bits for each minutiae representation. Statistical experiments on public fingerprint databases demonstrate that our proposed approach can significantly improve the search accuracy of the benchmark MCC-based indexing scheme. The binary hash codes can achieve a significant search speedup compared with the MCC bit-based representation.**

***Index Terms*— Fingerprint recognition, binary codes, Markov random fields, nearest neighbour searches.**

I. Introduction

CORE to personal identification systems is the retrieval of relevant identities based on biometric traits. With ever-increasing data volume and access demand, it is necessary to develop efficient search methods subject to a certain accuracy level. This is particularly important for biometric identity management in critical national security applications that involve large-scale computationally intensive tasks. For example, identity de-duplication is essential to ensure the uniqueness of an enrolment. Current de-duplication service typically performs cross-matching over the data set [1], which can become prohibitive when operating at large scale.

In general, a search can be made more efficiently by 1) increasing the speed of a comparison and 2) reducing the number of comparisons. For one-to-one comparison, the speed largely depends on how the similarity measure is evaluated between two instances, whereas the choice of the similarity measure generally depends on the feature representations. For instance, an exhaustive search can finish one million comparisons of binary iris codes in one second on a single computer whereas comparisons between real-valued feature vectors require specific circuits and are usually done much slower. On the other hand, database filtering can be accomplished by either exclusive classification or nearest neighbour search algorithms. In this paper, we propose an integrative approach to index and search fingerprints in Hamming space.

Biometric fingerprint templates are not typically represented as fixed-length binary strings but real-valued point-sets with a dynamic number of minutiae points. In the fingerprint literature, there are binary embedding algorithms for converting point-set features into binary strings, such as [2]–[5]. However, they are mostly developed for feeding particular security sketches of template protection in biometric cryptosystems [6], [7]. As the objectives therein are accuracy and secrecy, the resulting binary representations are often of long bit-length to achieve high discriminability for one-to-one verification. For example, the bit-implementation of the minutiae cylinder code (MCC) [2] encodes a minutiae point into a 384-bit long binary feature vector. In [3], 8960 spectral bits and 2000 phase bits are required to encode a single spectral minutiae. The enhanced multi-line code proposed in [4] takes $1476B$ bits per minutiae, where B is the quantization bit-length, the value of which is usually larger than one. Note that, even with the 384-bit MCC, it requires 15 kilobits to encode a template with 50 minutiae points, not even including the validity mask.

Such long binary representations are acceptable for one-to-one verification owing to the fast operations of XOR and bit counts. However, they can be problematic for large-scale searches. In general, a nearest neighbour search in Hamming space can be done by exploring the Hamming-ball around a query or using hash tables. As the average Hamming distance increases, the Hamming-ball volume quickly becomes prohibitive to explore. Beyond 64 bits or so,

Manuscript received October 7, 2014; revised January 19, 2015 and March 21, 2015; accepted March 26, 2015. Date of publication April 9, 2015; date of current version June 18, 2015. This work was supported in part by the HBKU Strategic Development Fund, Hong Kong Research Grants Council (HKBU211612 and HKBU12202214), and in part by the National Natural Science Foundation of China (61272366 and 61403324). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Matti Pietikainen.

Y. Wang is with the Department of Computer Science, Institute of Computational and Theoretical Studies, Hong Kong Baptist University, Hong Kong (e-mail: yiwang@comp.hkbu.edu.hk).

L. Wang is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: 15008401372@126.com).

Y.-M. Cheung is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, and also with the United International College, Beijing Normal University—Hong Kong Baptist University, Zhuhai 519085, China (e-mail: ymc@comp.hkbu.edu.hk).

P. C. Yuen is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: pcyuen@comp.hkbu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2421332

the Hamming-ball method is no longer significantly faster than an exhaustive search. It can also be quite often the case that a query may not find any neighbour within the restricted volume [8]. For hash-based methods, long binary codes tend to result in a low recall because the collision probability decreases exponentially with an increasing code length [9]. As a result, there is a strong motivation to develop compact binary codes for large-scale search applications.

The main contributions of this paper are:

- We study data characteristics of MCC and observe that its binary representation is bit-correlated. This points to the possibility of representing a minutiae feature in a more compact binary form.
- We propose a theoretical framework for systematically learning compact binary hash codes. In particular, we apply the theory of Markov random field (MRF) to model adjacent bit correlations in the binary representation of MCC. This enables us to learn hash bits from a generalized linear model (GLM) whose maximum likelihood estimates can be conveniently obtained using efficient algorithms.
- We design a hierarchical fingerprint indexing scheme based on the proposed hash codes. Under the new framework, the binary search code length can be significantly reduced from 384 bits to 24 bits. This allows less hash functions and tables to be used for nearest neighbour search in Hamming space.

The remainder of this paper is organized as follows. Section II provides a review of the related work in fingerprint indexing. Section III presents the theoretical framework for modelling bit correlations and learning hash codes from MCC binary representations. Section IV describes the hierarchical fingerprint indexing scheme for nearest neighbour search based on the binary index codes. Section V reports experimental results. Finally, we draw conclusions in Section VI.

II. Related Work

Fingerprint indexing creates index values for every identity in the database so that those with higher matching scores can be mapped closer to each other in the index space. Earlier work along this line has been focused on real-valued indexing features and relevant similarity-preserving transformations for dimensionality reduction [10]–[12]. Typically, distances of *all* points to a probe in the index space are sorted to return a candidate list.

For fast retrieval, the index terms of database entries may be organized into certain data structures. This can be done following the partitioning principle as in most nearest neighbour search algorithms [13]. Examples include tree-like data structures such as Kd-tree [14] and minutiae tree [15]. A critical step of these methods is to identify the pivots for partitioning the space. In this regard, the index codes [16] based on matching scores as the indexing features also fall into this category. However, finding an optimal set of pivots is not trivial.

Another class of nearest neighbour search techniques is based on the *collision* principle [17]. The basic idea is to hash similar points into the same "buckets" such that, with a relatively high probability, it will find colliding segments from two similar instances in at least some of these buckets. The hashing strategy has been shown to outperform tree-based techniques in high dimensions [18]. Geometric hashing is one of the earliest collision-based methods used for point-set (e.g., fingerprint) recognition [19]. The model points are represented in a transformation-invariant way and stored redundantly with their identifiers in a hash table. Recognition of a query object is based on accumulating the collision scores of similar local invariants and their geometric relations. The retrieved model with the maximum collision score is considered as the most likely candidate for a match.

Conventional fingerprint geometric hashing algorithms use minutiae triangulation for extracting local geometric invariants [20], [21]. However, Delaunay triangulation is sensitive to noise and distortion. To improve the stability and robustness, complicated construction schemes and extra geometric invariants are required [22]. They often result in real-valued and high-dimensional feature descriptors in order to achieve accuracy. Moreover, most of these algorithms generate index values by quantizing the geometric invariant measures. As a result, homogeneous local features are used for both index creation and feature comparisons. As only local information is exploited, these methods can become problematic if two fingerprints have small overlapping areas.

Although there is an extensive literature on nearest neighbour search in Euclidean space, only a few strategies exist for retrieval in Hamming space [8]. The most popular method is locality sensitive hashing (LSH) [23], [24]. Recently, LSH-like algorithms were introduced to search large-scale biometric databases of iris codes [25], palmprint codes [26], and MCC [27]. In [28] and [29], LSH was also applied to combined level-1 and level-2 fingerprint features. In these applications, LSH mainly serves for two purposes: 1) reducing dimensionality of the input binary strings, and 2) clustering data points into buckets. For binary feature vectors, the LSH functions of random sampling bits can preserve Hamming distance. This is due to the fact that, if the number of sampled bits is sufficiently large, the collision probability of two hashes is equal to the fraction of bit positions on which the two binary strings agree [8]. Thus, to achieve a good precision, LSH-related methods require more sampling bits and hash tables. Both can lead to a significant increase in query time and storage requirement for long inputs, typically seen in biometric representations, that often contain hundreds, if not thousands, of bits in a single instance.

Recently, machine learning techniques were leveraged to pursue compact binary hash codes for similarity search of natural images. This results in various data-dependent algorithms by considering the distribution of data points [30]–[32]. It is known that the solutions heavily depend on specific data characteristics and performance requirements of their applications [33]. Therefore, binary hashes developed for general images may not be used directly for biometric indexing because biometric data has its own characteristics

and performance requirements. Compared to a nature image search, biometric identification requires higher accuracy for more critical security applications. Due to the large sample variations inherent in biometric feature acquisition, the index value of a probe is not going to be identical to that of a match in database. For fingerprints in particular, the number of feature points is dynamic and there is no alphabetical nor numerical order among these points. Such specific challenges require special hash designs in the context of fingerprint search applications.

III. Learning Compact Binary Codes

In this section, we provide the details of our approach to learn compact binary codes from the binary representation of MCC for fingerprint search applications. The MCC representation is a robust and effective local feature descriptor. Recent studies showed that MCC, being a minutiae-only algorithm, can provide the best performance in terms of accuracy [2], [34] even for cross-device matching [35]. Its bit implementation requires hundreds of bits, compared to thousands of bits as in [3] and [4], for representing a single minutiae feature in the minutiae point set of a fingerprint template. It enables binary feature based fingerprint indexing [27], [28] and high performance fingerprint matching via parallelism [34]. Therefore, we choose to build on MCC by exploiting the data characteristics of its binary representation.

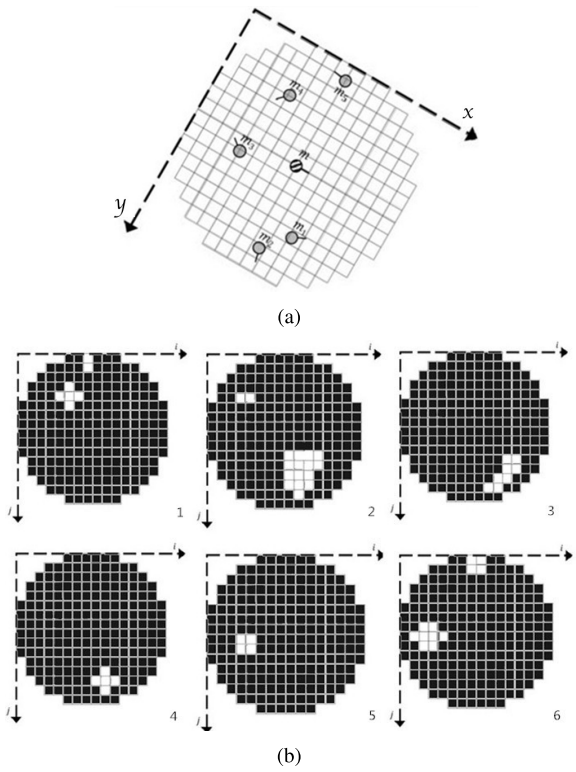

Fig. 1. Illustration of the MCC descriptor: (a) The local neighbourhood of a minutiae m; the reference frame created with m located at the origin and the x-axis pointing to its direction. (b) A cylinder is divided into six sections, each of which encodes some relative minutiae information in (a).

A. Data Characteristics of MCC

The MCC representation is derived from the minutiae-only representation (i.e., x-y coordinates and angles) of standard fingerprint templates. It encodes the neighbourhood information of each minutia into a 3D data structure, called minutiae cylinder, which is invariant to translation and rotation, and is robust against skin distortion and small feature extraction errors. The 3D cylinder structure is divided into sections, each corresponding to a directional difference in the range $[-\pi, \pi]$. Sections are discretized into a fixed number of $N \times N$ cells. Each cell value is calculated by accumulating spatial and directional contributions from all other minutiae in the neighbourhood for encoding. Note that the spatial contribution affects cell values in base and the directional contribution affects the height (i.e., which section to assign a base value) of the 3D cylinder. An example is provided in Fig. 1 for illustration of the MCC descriptor. In the original MCC algorithm, each cylinder cell is associated with two bits: one denoting the cell value and the other specifying the cell "validity". The corner cells may be labelled as "invalid" so that they are not used in the cylinder matching phase. The readers are referred to [2] for more details.

The cell values are quantized into binary values for the bit implementation. In practice, bits from all sections are concatenated into one fixed-length binary feature vector. It is worth to note that the resulting binary representation is long and with far more zeros than ones after quantization. This effect can be demonstrated by the statistics over 100,000 MCC bit-based representations that we collected from 2000 fingerprints in a benchmark database, which shows that the average bit value is 0.05. That is, about 95% of MCC bits are zeros on average.

We may use entropy [36] to derive a conservative estimate of the number of bits required for representing the local minutiae feature. In particular, let $X_1, X_2, \ldots, X_n$, where $n = N \times N$, be a sequence of random variables, each representing the binary value of a cell on the regular lattice of an MCC section. If $X_1, X_2, \ldots, X_n$ are independent and identically distributed, the probability $p = Pr\{X = 1\}$ is the same across all bit positions and is equivalent to the average bit value. Given $p = 0.05$ as obtained in our test above, the entropy $H(p)$ per MCC bit is thus approximately 0.3. Without loss of information, it is theoretically possible to find a more compact description of $384 \times 0.3 = 115.2$ bits for $n = 384$ in this case. If $X_1, X_2, \ldots, X_n$ are independent but *not* identically distributed instead, we estimate $p_i = Pr\{X_i = 1\}$ at each bit position for $i = 1, \ldots, n$ and obtain the entropy as

$$H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i). \tag{1}$$

In this latter case, the expected description length becomes 102.1 bits for the MCC bit-based representation.

Certainly, if the bits are somewhat correlated, the expected description length can be even shorter because

$$H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i) \tag{2}$$

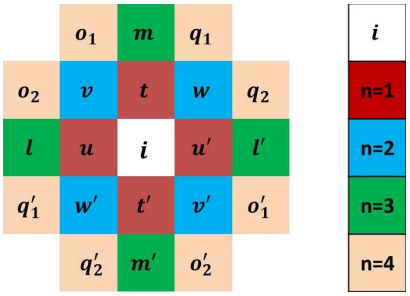

Fig. 2. Neighbours of cell i on a regular lattice where realizations in lower-case letter are written for random variables associated with each site. The outermost neighbours are encoded in different colors for the nth-order neighbourhood system with $n = 1, 2, 3, 4$.

with equality if and only if the X_i's are independent [36]. Indeed, correlations are likely to exist in the MCC bits due to the way they are generated. As shown in Fig. 1, when generating the MCC descriptors, the cell values are obtained from accumulating contributions of minutiae in the neighbourhood. In particular, the spatial contribution that a neighbouring minutia, denoted by m_t, gives to a cell m is a standard Gaussian function of the Euclidean distance between m_t and m [2]. The continuous function can extend minutiae contributions to adjacent cells, resulting in correlated values and hence bit dependencies even after quantization. This effect enables us to develop more compact binary codes by modelling bit correlations in the binary representation of MCC.

B. Modelling Bit Correlations

We propose to model each MCC section as an MRF for capturing bit correlations. MRFs have been widely used in image processing and computer vision tasks [37], [38]. They have also been adapted to fingerprint recognition for smoothing ridge orientation fields [39] and fingerprint enhancement [40]. Most of the work follow the paradigm of texture modelling [41], whereas in this paper we apply MRF for hashing long binary representations into more compact forms.

We consider a neighbourhood system $\mathcal{N} = \{\mathcal{N}_i | \forall i \in \mathcal{S}\}$, where $\mathcal{S}$ is the regular lattice of an MCC section and $\mathcal{N}_i$ is the set of sites neighbouring to cell i within an integer-numbered radius. The radius value defines the order of the neighbourhood system $\mathcal{N}$. Figure 2 illustrates an example by colouring outermost neighbours that encompass lower-order ones in the nth-order system for $n = 1, 2, 3, 4$.

Specifically, we treat the set of random variables $X_1, X_2, \ldots, X_n$ on $\mathcal{S}$ as a homogeneous MRF. In this way, only the neighbouring cell values are correlated. That is, $p(x_i|\mathbf{x}_{\mathcal{S}-\{i\}}) = p(x_i|\mathbf{x}_{\mathcal{N}_i})$, where $\mathbf{x}_{\mathcal{S}-\{i\}}$ denotes the values of bits on $\mathcal{S}$ excluding i, and $\mathbf{x}_{\mathcal{N}_i}$ denotes only the values of bits neighbouring to i. Further, the homogeneity property specifies that $p(x_i|\mathbf{x}_{\mathcal{N}_i})$ is translation invariant to i. That is, if $x_i = x_j$ and $\mathbf{x}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_j}$, then $p(x_i|\mathbf{x}_{\mathcal{N}_i}) = p(x_j|\mathbf{x}_{\mathcal{N}_j})$ for $i \neq j$.

There are a few methods for MRF parameter estimation [37]. One approach is by coding [42]. The basic idea is to partition $\mathcal{S}$ into several disjoint sets $\mathcal{S}^{(k)}$ such that no two sites in $\mathcal{S}^{(k)}$ are neighbours to each other. Under the Markovian property, variables associated with the sites in $\mathcal{S}^{(k)}$ are mutually independent, which provides convenience to calculate the maximum likelihood estimate of MRF parameters.

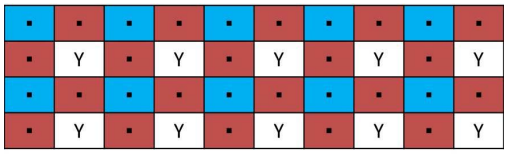

Fig. 3. Coding of a second-order MRF system. The "Y" sites are mutually independent in the presence of the "·" sites.

We are inspired by the coding method for a lossy compression. To illustrate the idea, Fig. 3 shows coding for a second-order MRF neighbourhood system. The "Y" sites form a coding set $\mathcal{S}^{(\mathrm{Y})}$. Due to Markovianity and homogeneity, the "Y" site variables are independent and identically distributed given values at the remaining "·" sites. Accordingly, the independent observations y_i on the "Y" sites have a joint distribution conditional on the MRF parameters θ, which is the product of density functions $f(y_i|\theta)$, i.e.,

$$f(\mathbf{y}|\theta) = \prod_{i \in \mathcal{S}^{(Y)}} f(y_i|\theta). \tag{3}$$

Since the expected value $E(Y_i) = Pr\{y_i = 1\}$ is a function of $f(y_i)$, we may set

$$E(Y_i) = g(\mathbf{x}_{\mathcal{N}_i}|\theta), \tag{4}$$

where $g(\cdot)$ is an unknown function defining the relationship between the expected value $E(Y_i)$ and some explanatory terms $\mathbf{x}_{\mathcal{N}_i}$ in the local neighbourhood, conditional on θ. By the homogeneity property, we can further drop the subscripts in (4) for building the model.

Once the model is trained, we can estimate $E(Y_i)$ and apply a threshold to produce one bit at each "Y" site i. In this way, (4) can be regarded as hashing the neighbourhood information, carried by $\mathbf{x}_{\mathcal{N}_i}$, into a single bit, by quantizing $E(Y_i)$. The compression ratio is effectively $4:1$ for a second-order neighbourhood system, as shown in Fig. 3, and $9:1$ for a third- or fourth-order scheme accordingly.

Usually, the set of MRF parameters θ is estimated by taking an arithmetic average of the maximum likelihood estimates over all coding sets. This is done by shifting the coding framework over the MRF lattice [41], [42]. For fingerprint data and with the homogeneity property, it is possible to collect a large number of $(y_i, \mathbf{x}_{\mathcal{N}_i})$ samples on a single coding set from MCC sections across all minutiae templates. With such a large sample size available, the MRF parameters θ can be approximated by a single coding estimate as the sample statistics converge to the same value from all coding sets.

C. Learning Binary Hash Codes

Instead of estimating the MRF parameters θ, we propose to regard the statistical relationship in (4) as a regression with

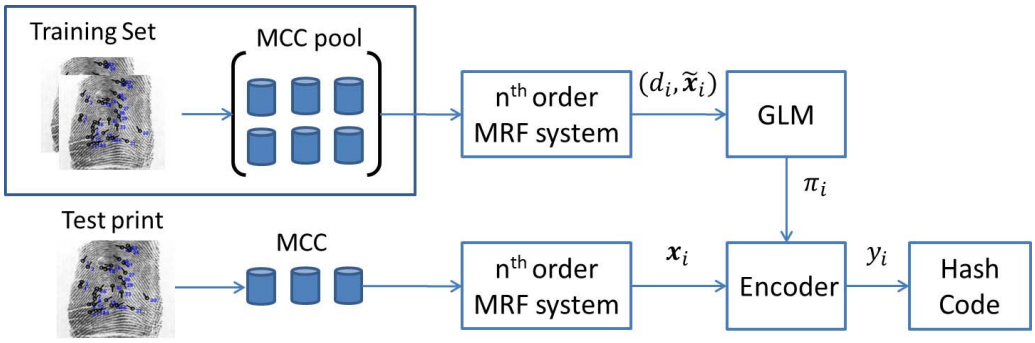

Fig. 4. Schematic diagram of learning the proposed binary hash codes.

a set of parameters β under the framework of GLM [43]. GLM provides a unification of various statistical methods particularly for distributions from the exponential family. In GLM, the evaluation interests are usually not the specific distribution parameters θ but a smaller set of parameters β that link the random variables to the explanatory terms. This is precisely the situation that we encounter in (4) in our context.

One strength of GLM is that it is not limited to distributions from the exponential family. In fact, it suffices to know the mean-variance relationship. This provides flexibility and robustness to model specification and parameter evaluation. Moreover, the maximum likelihood estimate of GLM parameters can be readily obtained using established optimization algorithms [43]. Under the GLM framework, our bit correlation model can be specified in terms of three essential components, namely, the probability distribution, the linear predictor and the monotone link function.

Specifically, let $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_t$ denote t possible different neighbourhood configuration patterns in one MCC section. By definition $\mathbf{x}_{\mathcal{N}_i}$ in the neighbourhood of cell i must match one of the t configuration patterns. Then, for each explanatory pattern $\tilde{\mathbf{x}}_i, i = 1, 2, \ldots, t$, we collect n_i samples of the pair $(y_i, \tilde{\mathbf{x}}_i)$ and count the number Z_i for which $y_i = 1$. Let $Z_i \sim \text{Binomial}(\pi_i, n_i)$, where the binomial distribution parameter π_i denotes the probability of $y_i = 1$ in response to a particular configuration pattern $\tilde{\mathbf{x}}_i$. Thus, the parameter π_i is in fact $E(Y_i)$ which is the left-hand side of (4) and is of our main interest. We can simplify the binomial distribution as $D_i = Z_i/n_i \sim \text{Binomial}(\pi_i)$. The log-likelihood function for a maximum likelihood estimate of $\pi_1, \pi_2, \ldots, \pi_t$, given observations of the response probabilities $d_1, d_2, \ldots, d_t$, is therefore

$$l(\pi_1, \ldots, \pi_t | d_1, \ldots, d_t) = \sum_{i=1}^{t} \left[d_i \cdot \log\left(\frac{\pi_i}{1-\pi_i}\right) + \log(1-\pi_i) \right]. \quad (5)$$

The linear predictor has the form $\mathbf{X}\beta$ for all explanatory terms $\mathbf{X} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_t]^T$, given $p < t$ parameters in $\beta = [\beta_1, \beta_2, \ldots, \beta_p]^T$. We propose to compose $\tilde{\mathbf{x}}_i$ from the combination of cliques which can effectively summarize the neighbourhood information. The details will be discussed separately in Section III-D.

The monotone link function connects the expected value of those response variables to the linear predictor. For example, the canonical link function of a binomial distribution with parameter π_i is the logit link $\log[\pi_i/(1-\pi_i)] = \tilde{\mathbf{x}}_i^T \beta = \eta_i$. Therefore,

$$\pi_i = g(\tilde{\mathbf{x}}_i|\beta) = \frac{\exp(\tilde{\mathbf{x}}_i^T \beta)}{1 + \exp(\tilde{\mathbf{x}}_i^T \beta)} = \frac{\exp(\eta_i)}{1 + \exp(\eta_i)}. \quad (6)$$

Accordingly, we obtain $l(\beta_1, \ldots, \beta_p | d_1, \ldots, d_t; \tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_t)$, denoted by $l(\beta|\cdot)$ for short, from (5) and (6). Note that $p < t$ results in a significant reduction on the number of parameters required for model specification. To obtain the maximum likelihood estimate for β, we need to use the chain rule of differentiation on $l(\beta|\cdot)$ with respect to each β_j. This gives the score $U_j = \partial l/\partial \beta_j$ for $j = 1, 2, \ldots, p$. The variance-covariance matrix of the U_j's is also termed the information matrix $\mathbf{J}$ with each element $J_{j,k} = E[U_j U_k]$. The method of scoring gives an iterative numerical evaluation of maximum likelihood estimates based on the Newton-Raphson algorithm:

$$\mathbf{b}^{(m)} = \mathbf{b}^{(m-1)} + [\mathbf{J}^{(m-1)}]^{-1} \mathbf{U}^{(m-1)}, \quad (7)$$

where $\mathbf{b}^{(m)}$ is the vector of maximum likelihood estimates for $\beta_1, \ldots, \beta_p$ at the m-th iteration, and the score matrix $\mathbf{U}^{(m-1)}$ and the information matrix $\mathbf{J}^{(m-1)}$ are evaluated based on $\mathbf{b}^{(m-1)}$ whose initial value can be randomly selected. Multiplying $\mathbf{J}^{(m-1)}$ at both sides, (7) can be written explicitly in matrix notation as [43]:

$$\mathbf{X}^T \mathbf{W}^{(m-1)} \mathbf{X} \mathbf{b}^{(m)} = \mathbf{X}^T \mathbf{W}^{(m-1)} \mathbf{z}^{(m-1)}, \quad (8)$$

where $\mathbf{W}$ is a diagonal matrix with elements

$$w_{ii} = \frac{1}{\pi_i(1-\pi_i)} \left(\frac{\partial \pi_i}{\partial \eta_i}\right)^2,$$

and $\mathbf{z}$ is a vector with elements

$$z_i = \sum_{k=1}^{p} x_{i,k} \beta_k + (d_i - \pi_i) \left(\frac{\partial \eta_i}{\partial \pi_i}\right).$$

Both π_i and η_i are evaluated at $\mathbf{b}^{(m-1)}$. Evaluation of (8) is called *iterative weighted least squares* (IRLS) as it takes the same form as a linear model obtained by weighted least square except being solved iteratively for $\mathbf{W}$ and $\mathbf{z}$ depending on $\mathbf{b}$. When $\Delta \mathbf{b} = \mathbf{b}^{(m)} - \mathbf{b}^{(m-1)}$ is sufficiently small, the IRLS procedure is converged with β.

Figure 4 plots a schematic diagram of our proposed bit reduction scheme. At the training stage, the MCC's are pooled

Algorithm 1 Training Model Parameters

INPUT : A training set of minutiae templates $DB_t = \{T_1, T_2, \ldots\}$
OUTPUT: model parameter matrix $\beta = \{\beta[k, j]\}$ where $k = 1, \ldots, p$ for p explanatory variables and $j = 1, \ldots, 6$ for each MCC section.

Reset counters $Z[i, j] = 0$ *and* $n[i, j] = 0$, *where* $i = 1, \ldots, t$ *for* t *possible* $\mathbf{x}$ *patterns*;
foreach *minutiae template* T *in* DB_t **do**
  foreach *minutiae point in* T **do**
    $M \leftarrow$ *MCC binary representation*;
    foreach $\mathcal{S}_j \subset M$ *of the* j*-th MCC section* **do**
      foreach Y *site in the coding set* $\mathcal{S}_j^{(Y)} \subset \mathcal{S}$ **do**
        $y \leftarrow$ *site values of* Y;
        $\tilde{\mathbf{x}}_i \leftarrow$ *neighbour site values of* y;
        $n[i, j] = n[i, j] + 1$;
        if $y == 1$ **then**
          $Z[i, j] = Z[i, j] + 1$;
        end if
      end foreach
    end foreach
  end foreach
end foreach
foreach $\mathcal{S}_j \subset M$ *of the* j*-th MCC section* **do**
  foreach i *in* $[1, \ldots, t]$ **do**
    $d_i \leftarrow Z[i, j]/n[i, j]$;
  end foreach
  substitute $(d_1, \ldots, d_t; \tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_t)$ *into (8) and evaluate the* j*-th column of* β;
end foreach

Algorithm 2 Learning Binary Hash Codes

INPUT : MCC binary descriptor M; model parameters β;
OUTPUT: hash code C

foreach *section* $\mathcal{S}$ *in* M **do**
  foreach Y *site in the coding set* $\mathcal{S}^{(Y)} \subset \mathcal{S}$ **do**
    $\mathbf{x}_{\mathcal{N}_i} \leftarrow$ *neighbour site values*;
    $\pi_i = g(\mathbf{x}_{\mathcal{N}_i}; \beta)$ as in (6);
    $y_i = \{0, 1\} \leftarrow$ *threshold* π_i *at* ϵ;
    add y_i *to the binary string* C;
  end foreach
end foreach

to collect the configuration patterns $\tilde{\mathbf{x}}_i$ and their observed responses d_i. All the $(d_i, \tilde{\mathbf{x}}_i)$ pairs are then input to GLM for model fitting as outlined in Algorithm 1. The expected "Y" cell response π_i is estimated and used to encode the local neighbourhood with a particular configuration pattern $\mathbf{x}_i$ into a single bit $y_i \in \{0, 1\}$. In particular, the parameter ϵ serves as the encoding threshold where $y_i = 0$ if $\pi_i < \epsilon$; $y_i = 1$ if $\pi_i \geq \epsilon$. The process is outlined in Algorithm 2.

Here we give a numerical example for the key quantities shown in Fig. 4. In a second-order MRF system, for example, the configuration pattern $\tilde{\mathbf{x}}_i = [1, 1, 0, 1, 1, 0, 1, 0, 0]$ has appeared $n_i = 176$ times, and in ten of these instances the "Y" cell response is one. This gives $d_i = 10/176 = 0.0568$. Note that most statistical packages (see [43] and [44]) for fitting GLM have implemented the IRLS procedure of (8). We used the glm function from the stats package in R where the sample size n_i may be used as an optional prior weight for model fitting. As a result, the estimated value of π_i in response to the pattern $\mathbf{x}_i = [1, 1, 0, 1, 1, 0, 1, 0, 0]$ is $\pi_i = 0.0880$, and the bit output is $y_i = 0$ given $\epsilon = 0.1$.

D. Order of the MRF

The joint distribution of the MRF variables $\mathbf{x}$ has the general form $P(\mathbf{x}) = \exp[Q(\mathbf{x})]/\sum \exp[Q(\mathbf{x})]$, where $Q(\mathbf{x})$ is called an *energy function* by taking a sum over all possible cliques, i.e., subsets of internal sites with a single site or mutual neighbours. For a rectangular lattice, the order of an MRF determines the number and types of cliques in a neighbourhood scheme. For example, the first order neighbourhood system contains the single-site and horizontal and vertical pair-site cliques. The second order system includes, in addition to those of the first order system, diagonal pair-site cliques and triple-site and quadruple-site cliques. As the MRF order increases, the probability distribution depends on not only more variables from neighbouring sites but also possibly more complex interactions of these neighbours. Since each MCC section has only 8×8 sites, we limit our attention to the case of a maximum fourth-order dependence for practical constructions.

The homogeneous first-order scheme for binary random variables on a rectangular lattice has the energy function [42]:

$$Q(\mathbf{x}) = \beta_1 \sum x_{i,j} + \beta_2 \sum x_{i,j} x_{i+1,j} + \beta_3 \sum x_{i,j} x_{i,j+1}, \tag{9}$$

where $\beta = \{\beta_1, \beta_2, \beta_3\}$ are parameters for the single-site cliques and vertical and horizontal pair-site cliques. This leads to the probability structure of y conditional on a given neighbourhood pattern $\mathbf{x}$:

$$p_i(y|\mathbf{x}) = \frac{\exp\{y \cdot f(\mathbf{x})\}}{1 + \exp\{f(\mathbf{x})\}} \tag{10}$$

with

$$f(\mathbf{x}) = \beta_1 + \beta_2(t + t') + \beta_3(u + u') \tag{11}$$

in the notation of Fig. 2.

The homogeneous second-order scheme has more terms added to (11), including:

- Two types of diagonal pair-site cliques:

$$\beta_4(v + v') + \beta_5(w + w'). \tag{12}$$

- Four types of triple-site cliques:

$$\beta_6(tu + u'w + w't') + \beta_7(tv + v'u' + ut') + \beta_8(tw + w'u + u't') + \beta_9(tu' + uv + v't'). \tag{13}$$

- Quadruple-site cliques:

$$\beta_{10}(tuv + t'u'v' + tu'w + t'uw'). \tag{14}$$

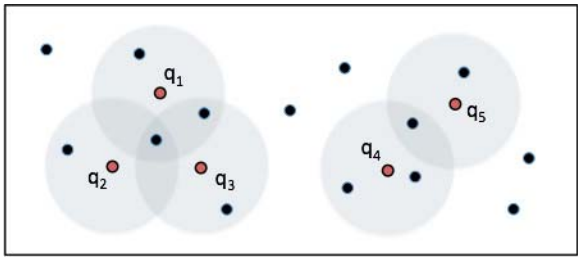

Fig. 5. Each minutiae point in its binary representation creates a Hamming ball in the search space. We propose to reduce the search radius of each point using more compact codes.

In particular, (13) and (14) can be regarded as the effect of interaction terms of the neighbouring sites. The scheme becomes an auto-logistic model if parameters for the interaction terms, i.e., β_6 to β_{10}, become all zeros.

Therefore, for $Pr(Y = 1|\mathbf{x})$, we can construct the explanatory variables in $\mathbf{x}_{\mathcal{N}_i}$ as $[1, (t + t'), (u + u'), \ldots]^T$ with terms from (11) to (14) so that $\eta_i = f(\mathbf{x}) = \mathbf{x}^T\beta$ with $\beta = [\beta_1, \beta_2, \beta_3, \ldots]^T$ in (6). Note that $\mathbf{x}$ may be extended with terms from higher-order schemes. In the simplest case, a third-order auto-logistic model will include two additional terms $(m + m')$, $(l + l')$ and a fourth-order model will add $(o1 + o1' + o2 + o2')$ and $(q1 + q1' + q2 + q2')$. Note that a lower-order MRF is a special case of a high-order MRF by setting parameters associated with the additional terms to zero.

IV. Hash-Based Fingerprint Indexing

Minutiae templates are indexed by an *unordered* set of binary hash codes. The conventional Hamming ranking only applies to a single minutiae point and does not guarantee any correspondence if a point is compared to its counterparts (compatible points). The situation studied here is illustrated in Fig. 5, where every minutiae point creates a Hamming ball encompassing its nearest neighbours in Hamming space. Therefore, it is a matter of collecting evidence from all the Hamming balls of a query fingerprint to nominate the most likely candidates of its match. In the following, we propose a hierarchical collision-based approach for indexing fingerprints with binary hash codes.

A. Hierarchical Fingerprint Indexing

We showed in [45] that a geometric hashing based on the MCC (Geo-MCC for short hereafter) can improve the retrieval accuracy over existing fingerprint indexing algorithms including the state-of-the-art MCC-LSH [27]. The MCC-LSH and other fingerprint geometric hashing algorithms used homogeneous features for both indexing points and approximating point similarities, which generally require high-dimensional representations to generate key values. The Geo-MCC method incorporates complementary information of local invariants and their relative geometric configurations for matching points and generating key values, respectively. It is more robust against noise and uncertainties such as missing points which are not unusual for fingerprints.

However, Geo-MCC has two limitations. Firstly, a few geometric hash bins contain a large number of entries. As the longest list dominates the search time, a more uniform distribution of the entries is desired. Secondly, the point matching is based on the MCC bit-based representation. Although this is done by fast bit operations in Hamming space, it is still desirable to reduce extensive comparisons of long binary codes. LSH can distribute binary codes more evenly to buckets by random bit sampling. Therefore, in this paper, we combine the merits of both hash-based methods and propose a hierarchical indexing scheme, named as Geo-LSH for short hereafter. Figure 6 illustrates a general schematic diagram of the proposed Geo-LSH framework. In the following, we describe the proposed algorithms in details.

The local invariants are MCC features [2] that may be hashed into more compact forms as introduced in Section III. We call the minutiae point that defines the local neighbourhood invariants as *basis*. The basis representations are indexed by LSH functions for binary vectors [17]. In the approach, each LSH function randomly selects a few bits from the input binary vector. Suppose that the input binary vector has n bits and the random selection is h bits per LSH function. Then, all entries in an LSH bucket have a Hamming distance no larger than $(n - h)$ bits. In other words, all bucket entries retrieved from the L LSH tables for a query point are in the query's Hamming ball with a radius of $(n - h)$ bits. With the binary input of the same bit length, decreasing the number h of random selection effectively increases the radius of Hamming ball search, whereas increasing the number of LSH functions explores more instances of the same radius but different at bit positions in the search space. Both include more instances for a nearest neighbour search, which avoids missing the true match at a cost of increasing the search time.

Each entry in a hit bucket contains an identifier of the instance and a feature for similarity measure. We propose to construct the matching feature from geometric hashing, called *geometric dictionary* in this paper. The underlying idea of geometric hashing is to incorporate relative spatial configuration of the local invariants as the access keys [19]. This can be done by creating basis-defined coordinates as if "seeing" the invariants from multiple access points (geometric hash keys). The multiple views create redundancy of description for the local invariants, which improves the robustness of matching against missing points and increases the probability of successful collisions. Conversely, we may use a bunch of geometric hash keys to identify a basis that unambiguously defines an orthogonal coordinate system in which the access points reside. An example is shown in the subframe of Fig. 6. Denote a basis point as q. Thus, every other point p in the point-set can take an affine transform with respect to q followed by a quantization as in [45].

The quantized values can be aggregated into a single integer number for positioning p in the reference frame defined by a particular basis. Accordingly, we create an integer array to construct a geometric dictionary. In particular, we implement the dictionary as a hash table container with the created integers as access keys. The access keys of each dictionary

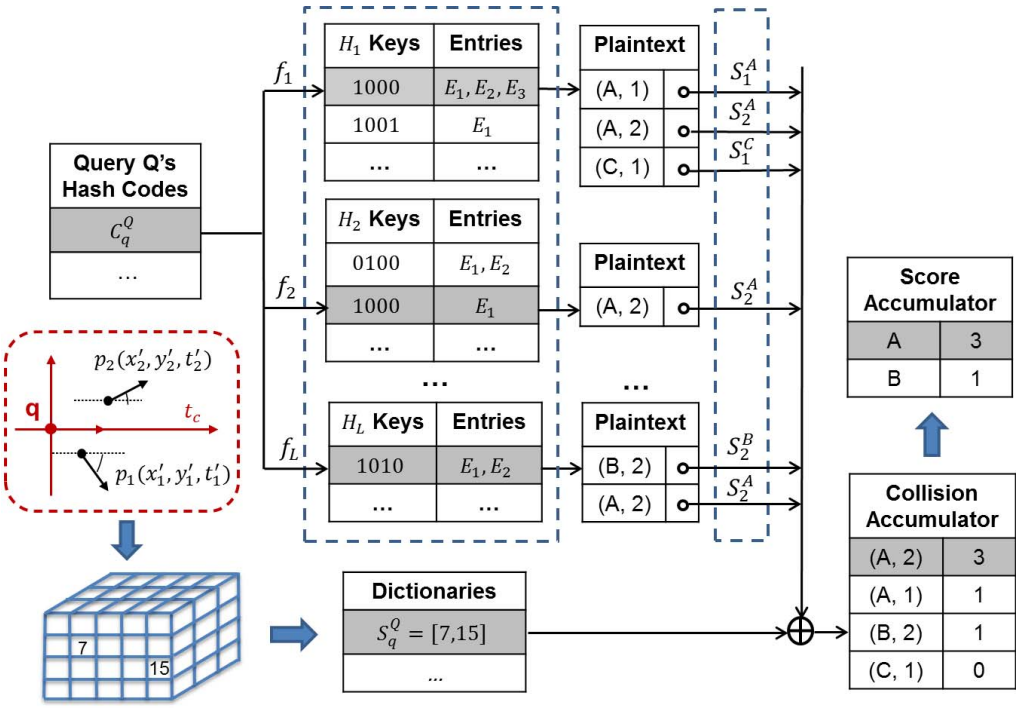

Fig. 6. The proposed Geo-LSH framework for a hierarchical collision-based fingerprint indexing approach.

encode the global geometric configuration from the view of a particular basis point. The matching of geometric dictionaries mimics geometric hashing in a micro way. In this way, we construct a hierarchical indexing framework where the first-level LSH is indexed by the basis points and the second-level geometric hashing is deployed by the geometric dictionaries.

The second-level geometric hashing is linked to the first-level LSH via the basis points. Each LSH table entry is composed of the pair $(Model\_ID, Minutiae\_ID)$ for identifying the basis point and the address of a geometric dictionary associated with that basis. For example, a typical LSH table entry may contain the identity (i, j) and a pointer value $Ptr(S_j^i)$ recording the address of dictionary S_j^i defined by the basis (i, j). Algorithm 3 outlines the main procedures for creating the hierarchical indices.

Algorithm 3 Creating Hierarchical Indices

INPUT : minutiae templates $DB_e = [T_1, T_2, \ldots, T_N]$; model parameters β; set of LSH functions $\mathcal{H} = \{f_1, f_2, \ldots, f_L\}$
OUTPUT: hash tables $H_1, H_2, \ldots, H_L$ and an unordered set of dictionaries $\{S_j^i\}$

foreach *minutiae template* T_i *in* DB_e **do**
  foreach *minutiae point* p_j^i *in* T_i **do**
    foreach *minutiae point* $p_k^i (k \neq j)$ *in* T_i **do**
      $s_k^i \leftarrow$ *geometric transform* p_k^i *w.r.t.* p_j^i;
      quantize and insert s_k^i *into dictionary* S_j^i;
    end foreach
    $E \leftarrow (i, j, S_j^i)$;
    $M_j^i \leftarrow$ *MCC binary descriptor for* p_j^i;
    $C_j^i \leftarrow$ `HashCode` (M_j^i, β);
    foreach f_l *in* $\mathcal{H}$ **do**
      $\mathbf{b} \leftarrow f_l(C_j^i)$;
      insert E *into bucket* $\mathbf{b}$ *of* H_l ;
    end foreach
  end foreach
end foreach

B. Searching by Accumulated Collisions

At retrieval time, a query template Q creates a set of binary index codes for the basis points and a set of dictionaries defined by each basis. For a query point, e.g., $q \in Q$ in Fig. 6, the LSH functions are applied to binary code C_q for allocating collision buckets in the corresponding hash tables. An authorized user is able to access all entries in the hit buckets and retrieve the identifier $(Model\_ID, Minutiae\_ID)$ and the dictionary address of potential matched points. Algorithm 4 summarises the main procedures.

The matched point similarity is measured by the number of collisions between the query and matched dictionaries, denoted as $collision(S_q, S_j^i)$ in Algorithm 4. Recall that the geometric dictionary is implemented as a hash table based container. Thus, the collision is simply the number of hit buckets via the access keys. The number is accumulated across the LSH tables. This is done with an accumulator matrix A reset for every basis point. The most likely match to the query basis is the one, among others belonging to the same $Model\_ID$, that has the maximum number of accumulated collisions. The value is taken as a score for $Model\_ID$ and accumulated for all query points in Q. The final list of candidates can be produced by sorting the score accumulator and return as

Algorithm 4 Searching by Accumulated Collisions

Input : A query template Q; model parameters β; set of LSH functions $\mathcal{H} = \{f_1, f_2, \ldots, f_L\}$; hash table $H_1, H_2, \ldots, H_L$ and dictionaries $\{S_j^i\}$
Output: a list of candidates $CL = \{(i, sco[i])\}$

reset score accumulator $sco[i] = 0$;
foreach *minutiae point* q *in* Q **do**
  reset collision accumulator $A[i, j] = 0$;
  foreach *minutiae point* $t (t \neq q)$ *in* Q **do**
    $s_t \leftarrow$ *geometric transform* t *w.r.t.* q;
    quantized and insert s_t *into* S_q;
  $M_q \leftarrow$ *MCC binary descriptor for* q;
  $C_q \leftarrow \texttt{HashCode}\,(M_q, \beta)$;
  foreach f_l *in* $\mathcal{H}$ **do**
    $\mathbf{b} \leftarrow f_l(C_q)$;
    foreach $(i, j, Ptr(S_j^i))$ *in* $\mathbf{b}$ *from* H_l **do**
      $A[i, j] = A[i, j] + collision(S_q, S_j^i)$;
    end foreach
  end foreach
  foreach T_i *with at least one collision in* A **do**
    $sco[i] \leftarrow sco[i] + max_j\{A[i, j]\}$;
  end foreach
end foreach
Return $CL = \{(i, sco[i])\}$ *with the* max_c *highest scores*

the top max_c number of $Model\_ID$'s that have the highest accumulated scores.

C. Memory Consumption

Assume a fingerprint database of N minutiae templates and each template contains F minutiae points. Every minutiae point in a template defines a basis for a specific geometric transformation of the point-set. This results in $O(NF)$ basis points indexed by LSH and $O(NF)$ dictionaries indexed by geometric hashing, respectively. It is possible to implement the dictionaries as an array index of $(F-1)$ 4-byte integers. But such a direct approach can require a logarithmic search time. So we implement the geometric dictionaries in hash table based containers that can offer constant-time lookups, as provided in the standard C++11 library. Since each dictionary is independent of each other, the related processing can be done in parallel and in an asynchronous manner. It also allows easy addition and removal of instances, which is vital with large datasets. The dictionaries may be stored on separate disks and accessed independently via pointers when needed.

The basis points are indexed by binary codes. There are $O(NF)$ entries in each LSH table. Each entry contains a 4-byte identifier $(Model\_ID, Minutiae\_ID)$ and a 4-byte address of the dictionary associated with the basis point. The number of bits required for an identifier should be larger than $\lceil \log_2(NF) \rceil$. Thus, 4 bytes (32 bits) can identify 16 million fingerprints each with maximum 128 minutiae points. Note that the hash functions are independent and do not require continuous memory. The LSH tables can be loaded and searched in parallel. With $N = 1.6 \times 10^6$ and $F = 128$, the memory required for a LSH table is about $(4+4) \times N \times F = 8 \times 1.6 \times 10^6 \times 128 = 1.5$GB, which can be easily fit into the memory of most desktop PCs today.

Note that each LSH table accommodates a copy of all points (hashed to different buckets though) from the enrolled subjects. As more hash tables are used, more replicates of the database entries have to be stored. Therefore, reducing the number of hash tables is needed for memory savings especially on large-scale data sets.

D. Computational Complexity

Existing fingerprint geometric hashing algorithms construct local invariant features based on minutiae triangulations. This preprocessing takes $O(F^3)$ time for arbitrary minutiae triplets and $O(F \log F)$ time for the Delaunay triangulation [22], [46]. Our approach requires two preprocessing parts. One is to obtain binary codes. Once the model parameters are learned from the training set, it takes $O(F)$ time to compute MCC and subsequently $O(F)$ time to compute the binary hash codes. This part of preprocessing is very fast as each MCC bit-based representation can be computed within a millisecond using the MCC SDK [47] on a 3.40 GHz Intel(R) machine. The other part is to obtain geometric dictionaries. Since every minutiae point is used as a basis, there are F transformations for calculating the relative positions of $F-1$ points in the basis-defined reference frame. This results in $O(F^2)$ operations. Therefore, the total preprocessing time is $2O(F) + O(F^2)$ for a template.

At retrieval time, the key value of a query point is hashed, and then the corresponding bucket is searched for the matching item. The hashing and bucket access are constant-time, but the time to search a bucket for the matching item is linear with the number of items in the bucket. It is desirable to keep the number of items in each bucket small, which is affected by two factors. Firstly, the number of buckets should be large enough and every bucket contains at least one item. Secondly, the hashing function should distribute items more evenly among the buckets so that they each contain about the same small number of items. In LSH, the number of buckets is controlled by the number of bits selected by each hash function. As the number of items increases, the bit length of binary index codes should also increase. Ideally, for binary codes that have maximum entropy, random sampling bits will produce bit patterns of the same length with an equal probability and thus distribute the binary descriptors to buckets indexed by the bit patterns uniformly. In that case, by selecting a sufficient number of bits, it is possible to provide constant lookup time in the LSH table.

The proposed hierarchical indexing casts a vote to every entry in the hit buckets and accumulates evidence from relevant dictionaries. Therefore, the number of entries in the most occupied bin in an LSH table determines the worst case scenario for a query search. By sampling long enough bits, LSH can generate a sufficient number of possible buckets. Thus, the search complexity largely depends on how the basis points, hashed by their binary code representations, are distributed over the buckets. Suppose the most occupied bin

has M entries (the worst case being $M = NF$). A query point needs to search M dictionaries and find the best match with maximum collisions of the geometric indices. For a fingerprint with F minutiae points, the search time complexity is thus $O(MF)$. The maximum collision counts are accumulated for all N fingerprints in the database. The final results of N scores are sorted to produce a list of match candidates, which can be done in $O(N \log N)$ time.

V. Experimental Results

In this section, we evaluate the performance of our proposed approach. In particular, we demonstrate the search accuracy achieved by our hierarchical indexing approach and the speed-up in search using more compact codes after hashing as proposed in this paper. The experiments are conducted on the public fingerprint benchmark databases of FVC2002 DB1 [48] and NIST SD 14 [49].

FVC2002 DB1 has two subsets: FVC2002 DB1a contains 800 optical-scanned fingerprints from 100 subjects each with 8 different impressions, and FVC2002 DB1b contains 80 fingerprints from 10 subjects separated from those recorded in FVC2002 DB1a. In the following FVC experiments, we used FVC2002 DB1b for training the model parameters and FVC2002 DB1a for testing the indexing performance. Unless otherwise specified, the first impression was used for enrolling the subject and the rest seven impressions of the subject were used as query for searching the FVC2002 DB1a database.

The NIST SD 14 database contains in total 27,000 pairs of ink-rolled impressions scanned from fingerprint cards. In general, the ink-rolled impressions tend to suffer more noise and distortions compared to the live-scanned fingerprints. The instances in NIST SD 14 are collected randomly to resemble the natural horizontal distribution of fingerprint classes.

In all experiments, we used the third-party commercial software VeriFinger 6.6 from NeuroTechnology to extract minutiae features from the fingerprints. No particular pre-processing steps, such as image enhancement, foreground segmentation or fingerprint alignment, were carried out before feature extraction. The extracted minutiae templates are input to the MCC SDK v1.3 software [47] using the same parameters reported in [27] to create the MCC bit-based representations. In particular, we followed the practice in [27] that disregards the cell validity bits by considering all cells valid. Therefore, MCC bit-based representation has effectively 384 bits created from $8 \times 8 \times 6$ cylinder cells for each minutiae feature.

The 384-bit MCC is input to Algorithm 1 and Algorithm 2 to generate binary hash codes. We tested the 96-bit and 24-bit binary hash codes generated from the second- and third-order MRF system, respectively. Recall in Algorithm 2 that the encoding threshold ϵ can be used to control the output ratio of zero- and one-valued bits. Increasing ϵ has the effect of producing more zero-valued bits and resulting in more instances hashed to the same bin. In the extreme case where we have all zeros, our proposed hierarchical indexing scheme, Geo-LSH, will reduce to exhaustive comparisons based on geometric dictionary. On the other hand, decreasing ϵ will produce more one-valued bits that may introduce spurious information and thus impair the indexing performance. Our empirical results showed that in this context the encoding threshold $\epsilon = 0.1$ is appropriate for the FVC database and $\epsilon = 0.2$ is appropriate for the NIST database. With $\epsilon = 0.1$ for the FVC database, the percentage of zero-valued bits is on average 91.7% for the 96-bit code and 91.2% for the 24-bit code.

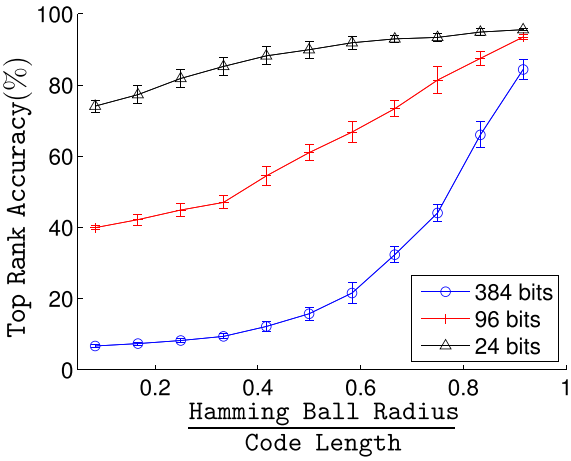

Fig. 7. ANN search performance w.r.t. Hamming ball radius for binary codes.

A. Nearest Neighbour Search in Hamming Space

As illustrated in Fig. 5, a fingerprint is not represented by a single but a set of binary codes. Each minutiae feature in the point-set is associated with a Hamming ball of its nearest neighbours in the binary vector space. Thus, one should aggregate evidence collected from all relevant Hamming balls for comparing two fingerprints. It is possible to approximate the Hamming distance between two binary vectors by the number of hash collisions in LSH [27]. However, the approximation only works for moderately large value of n (long binary vectors) and compatible points already in the Hamming ball. In our approach, both conditions are relaxed as the point similarities are measured by maximum collision counts between two geometric dictionaries. The approximate nearest neighbour (ANN) search is done via exploring a Hamming ball by retrieving entries from the hit buckets that are LSH indexed.

We compared MCC bit-based representations before and after hashing as input to the LSH. Since each hash table accommodates a copy of all points from the enrolled subjects, we used one LSH function (hence one hash table) at a time to avoid redundant points in an ANN search and ran the random selection of bits for ten times. The experiments were conducted on the FVC database. Since the training sample size in FVC2002 DB1b is small (only 80 fingerprints), we combined all the observation patterns from the six MCC sections to obtain more reliable statistics for fitting the GLM in Algorithm 1. This adjustment was carried out in all the FVC experiments.

Figure 7 reports the ANN search performance from the ten runs. The performance is plotted with respect to an increasing radius of Hamming ball for the comparing binary codes

of 384 bits, 96 bits and 24 bits, respectively. For the ease of comparison, the search radius r is normalized by the code length n as the x-axis. As r increases, more points are probed in the search space. When $r/n = 1$, it becomes an exhaustive search of all enrolled points. Note that the number of random selection bits $h = n - r$. When only one bit is selected, half of the points will be put into the same bucket. In particular, we increase r by every $1/12$ of the respective code length n. For example, the normalized radius $r/n = 0.08$ indicates a Hamming ball radius of $r = 2$ bits for a 24-bit code, $r = 8$ bits for a 96-bit code and $r = 32$ bits for a 384-bit code, respectively.

The y-axis plots the top rank accuracy corresponding to the search radius, i.e., the percentage of true match in the top-rank results retrieved from the hit bucket in one hash table. It can be seen that the ANN search is more effective using more compact codes in Hamming space. With a smaller search radius, even in the normalized scale, the chance of finding a true match is significantly higher after hashing. For example, at a normalized radius $r/n = 0.08$, finding a true match in the top rank is on average over 65% more likely by the 24-bit code and 30% more likely by the 96-bit code after the proposed bit reduction on the 384-bit MCC representation. The search performance will converge to the maximum value (below 1.0 and is determined by the point comparison method) in all three cases when $r/n = 1$ equivalent to an exhaustive search.

The search accuracy of 384-bit long binary codes can be largely boosted by using multiple hash tables. However, this will also increase the search time significantly as can be seen in the indexing experiments below. It is also worth to note that more hash tables implies more replicates of data points as they are hashed to different buckets via the LSH functions. This increases the storage requirement, which can become impractical for large data sets.

B. Fingerprint Indexing Experiments

Here, we evaluate the performance of our hierarchical indexing approach, Geo-LSH, proposed in this paper. In particular, we focus on the effect of using binary hash codes in the proposed framework. The indexing performance is typically indicated by the trade-off between identification accuracy and efficiency [48]. The identification accuracy is often measured by the *hit rate* which is the percentage of queries found with correct identities, while the efficiency is measured by the *penetration rate* which is the proportion of database that the system has to search.

The 384-bit MCC and the resulting binary hash codes are used as input to the Geo-LSH scheme, which are named hereafter as 384-bit Geo-LSH, 96-bit Geo-LSH and 24-bit Geo-LSH, respectively. For comparison, we used the 384-bit MCC based LSH algorithm as the performance benchmark, since the LSH algorithm outperforms many other non-hash based and conventional hash based methods for fingerprint indexing [27], [45]. The MCC-LSH results were produced from the MCC SDK v1.3 software [47] with the MCC and LSH parameter setting following those reported in [27].

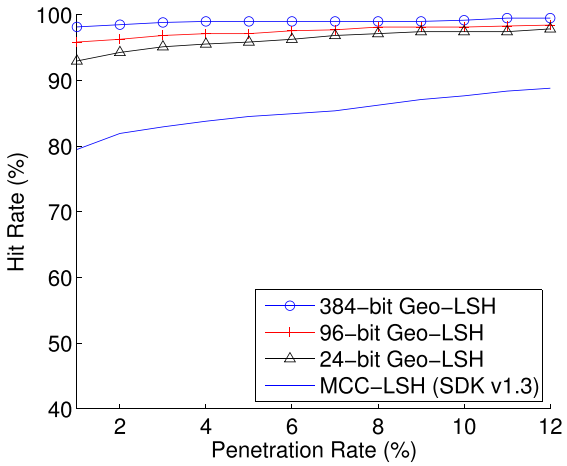

Fig. 8. Indexing performance on FVC2002 DB1.

TABLE I
LSH PARAMETERS USED FOR FVC2002 DB1

Binary inputs	384 bits	96 bits	24 bits
Number of hash functions	32	8	4
Bits selected per hash function	24	16	16

Figure 8 plots the FVC2002 DB1 indexing results. A drop of the MCC-LSH performance can be noticed in Fig. 8 compared to that reported in [27] on the same database. We believe that the discrepancy is largely due to different minutiae extraction tools. In our experimental settings, we do not carried out any pre-processing step on the fingerprints before the minutiae feature extraction. Thus, the extracted features are likely to contain more noise such as spurious detections and missing points. The noisy input to the SDK software may affect the indexing performance. Nevertheless, we used the same minutiae features as input to all the testing methods in our experiments. We consider that the indexing results are still fairly comparable in this case. Moreover, the noisy inputs can be useful to test the robustness of the comparing methods in the presence of noise.

Our experimental results showed that the proposed Geo-LSH scheme outperforms MCC-LSH in terms of the search accuracy by the candidates returned. We believe that the performance gain is credited to two main reasons. Firstly, the point similarity is measured instead of being approximated. Secondly, the global geometric configuration is incorporated via the dictionary set and provides complementary information to the local feature for matching. The gain may come with a cost of time though. This is especially so for long binary codes, as will be seen in later experiments. Table I provides the LSH parameters used for the binary codes of different length.

In Fig. 8, it can be observed that the 384-bit MCC has the best hit rate among all. Note that its normalized search radius is $(n - l)/n = (384 - 24)/384 = 0.94$ which is fairly close to 1 in Fig. 7. Thus, there are more points hashed to each used bucket. By using multiple (32 in this case) LSH functions, the search performance is largely boosted by accumulating collisions under many hash functions.

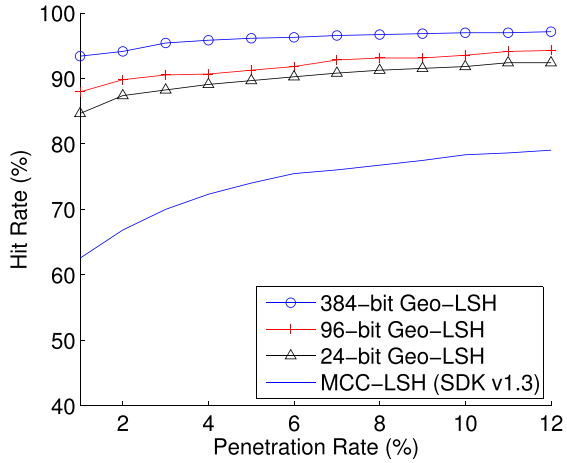

Fig. 9. Searching partial minutiae points on FVC2002 DB1.

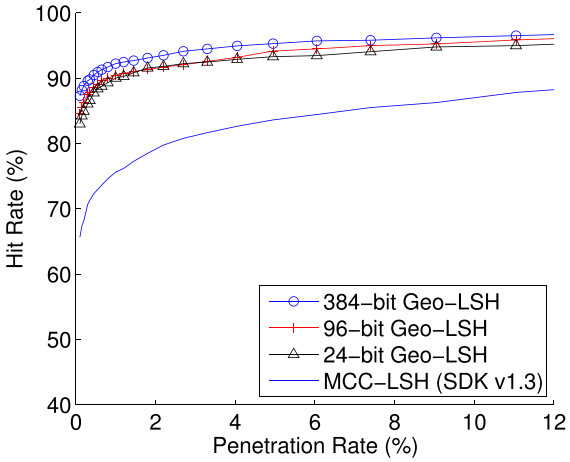

Fig. 10. Indexing performance on 2,000 segmented fingerprints of NIST SD14.

Whereas after bit reduction, the binary hash codes require much less hash functions and smaller search radius ($r/n = 0.33$ for 24 bits and $r/n = 0.83$ for 96 bits) to reach an accuracy close to that of 384 bits.

Figure 9 plots the indexing performance by searching partial fingerprints on the FVC database. In particular, we used only the first half part and discarded the second half part of the minutiae features in each query template generated by the VeriFinger 6.6 software to search the FVC database. It can be observed that the proposed search algorithm again outperforms MCC-LSH and the proposed hash codes are able to maintain a similar level of accuracy as that of the 384-bit MCC.

We also performed the indexing experiments on NIST SD 14. We firstly conducted a small-scale experiment on the last 2,700 pairs of fingerprints from NIST SD 14. The 2,700 pairs of fingerprint images are foreground segmented. Among them, about 700 fingerprints are used for training parameters and the remaining 2,000 pairs are used for the tests. Figure 10 plots the indexing results of enrolling 2,000 fingerprints and searching on their counterparts. Again, there is a significant gain (about 20%) of accuracy achieved by the proposed Geo-LSH method over LSH. With the same search algorithm, the maximum difference in hit rate between the 96-bit code and the 384-bit MCC is about 2.1% while that between the 24-bit code and the 384-bit MCC is about 4.5%. The LSH parameters in this case are listed in Table II.

TABLE II
LSH PARAMETERS USED FOR NIST SD 14

Binary inputs	384 bits	96 bits	24 bits
Number of hash functions	32	12	4
Bits selected per hash function	24	18	16

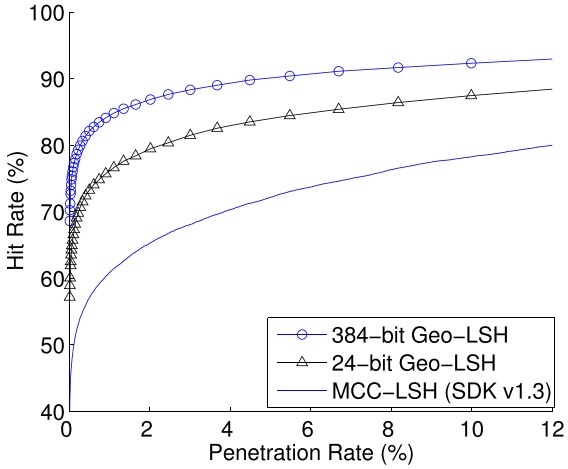

Fig. 11. Indexing performance on 20,000 fingerprints from NIST SD 14.

Figure 11 plots the indexing results of the comparing method on NIST SD 14 by enrolling the first 20,000 fingerprints and excluding the last 7,000 prints for training model parameters. Compared to Fig. 10, the search accuracy is dropped in both schemes because the 20,000 pairs of fingerprints used in the test are not foreground segmented and thus tend to contain more noise in the extracted minutiae templates. However, the proposed Geo-LSH method is more robust in the presence of noise.

It is important to note that the binary hash codes can achieve a significant speed-up as well as savings of storage space by reducing the number of hash functions (thus hash tables) and bits selected per hash function. For example, by using 32 hash functions, the 384-bit MCC representation requires 32 replicates of all points from the enrolled templates, whereas the 24-bit code only requires 4 hash functions. This is particularly important for large-scale data sets.

Figure 12 plots the average time by searching one query print against an increasing number of templates enrolled in the database. The timer refers to C# implementations on a 3.4 GHz Intel(R) machine. The results show a general linear increment of search time as the database rolls bigger. The 384-bit Geo-LSH scheme is much slower than the MCC-LSH because of the second-level hashing for point similarity comparisons. However, this time inefficiency of Geo-LSH can be largely mitigated by using the more compact binary codes proposed in this paper. As shown in Fig. 12, Geo-LSH based on the 24-bit code can significantly reduce the search time per query after bit reduction from the 384-bit code. Table III reports the

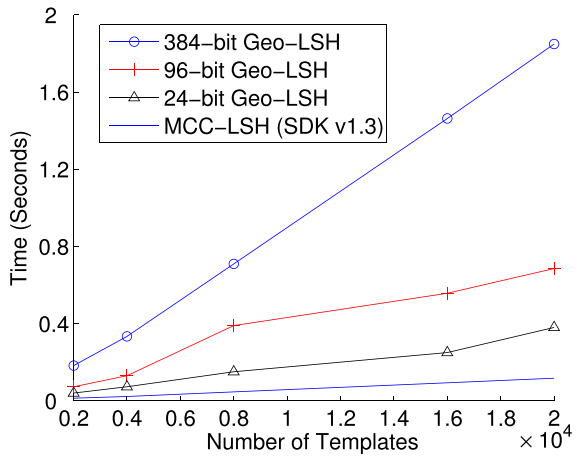

Fig. 12. Average time of searching one query against an increasing data set.

TABLE III
AVERAGE SEARCH SPEED-UP BY THE HASH CODES

No. of Templates	2000	4000	8000	16000	20000
384-bit Geo-LSH	1	1	1	1	1
96-bit Geo-LSH	2.51	2.57	1.82	2.63	2.70
24-bit Geo-LSH	4.59	4.60	4.72	5.86	4.87

average search time speed-up obtained by the compact codes. It can be seen that a Geo-LSH search based on the 96-bit code and that based on the 24-bit code is about 2.5 times and up to 5.86 times, respectively, faster than that based on the original 384-bit MCC representation.

VI. CONCLUSION

In this paper, we proposed to learn compact binary hash codes from high-dimensional binary representations of biometric fingerprints. This is done by applying the theory of MRF to model bit correlations in the translation-invariant local structure of the 384-bit MCC. The hash ratio is 4:1 by using a second-order MRF and 9:1 by using a third-order MRF, resulting in 96-bit and 24-bit binary codes, respectively. We also developed a hierarchical indexing scheme that combines the merits of LSH and geometric hashing. The proposed Geo-LSH indexing approach can effectively achieve superior identification accuracy and is more robust in the presence of noise and missing points. Moreover, it does not require any pre-processing steps such as image segmentation and pre-alignments for feature extraction. Although the gain comes at an expense of more memory consumption and longer search time due to the additional point similarity comparison, the cost can be largely mitigated by using compact binary hash codes developed in this paper. Our indexing experiments showed that a Geo-LSH search based on the 24-bit hash code can maintain a comparable identification accuracy while achieving a significant speed-up of over four-fold in the average search time compared to that based on the 384-bit MCC representation. Our future work includes extending the proposed framework to indexing latent fingerprints and developing compact binary codes with higher entropy values. We also plan to submit the proposed approach to FVC-onGoing [50] for independent evaluation.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their valuable comments that contributed to the improved quality of this paper.

REFERENCES

[1] Neurotechnology. (2014). *Duplicates Search Service*. [Online]. Available: http://www.neurotechnology.com/duplicates-search-service.html

[2] R. Cappelli, M. Ferrara, and D. Maltoni, "Minutia cylinder-code: A new representation and matching technique for fingerprint recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2128–2141, Dec. 2010.

[3] H. Xu and R. N. J. Veldhuis, "Binary representations of fingerprint spectral minutiae features," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 1212–1216.

[4] W. J. Wong, A. B. J. Teoh, M. L. D. Wong, and Y. H. Kho, "Enhanced multi-line code for minutiae-based fingerprint template protection," *Pattern Recognit. Lett.*, vol. 34, no. 11, pp. 1221–1229, Aug. 2013.

[5] Z. Jin, M.-H. Lim, A. B. J. Teoh, and B.-M. Goi, "A non-invertible randomized graph-based Hamming embedding for generating cancelable fingerprint template," *Pattern Recognit. Lett.*, vol. 42, pp. 137–147, Jun. 2014.

[6] A. Nagar, K. Nandakumar, and A. K. Jain, "Multibiometric cryptosystems based on feature-level fusion," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 255–268, Feb. 2012.

[7] N. Zhang, X. Yang, Y. Zang, X. Jia, and J. Tian, "Generating registration-free cancelable fingerprint templates based on minutia cylinder-code representation," in *Proc. IEEE 6th Int. Conf. Biometrics, Theory, Appl., Syst.*, Sep./Oct. 2013, pp. 1–6.

[8] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," in *Machine Learning for Computer Vision*, vol. 411, R. Cipolla, S. Battiato, and G. M. Farinella, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 49–87.

[9] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast exact search in Hamming space with multi-index hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1107–1119, Jun. 2014.

[10] R. Cappelli, D. Maio, and D. Maltoni, "Multispace KL for pattern representation and classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 977–996, Sep. 2001.

[11] Y. Wang, J. Hu, and 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.*, vol. 29, no. 4, pp. 573–585, Apr. 2007.

[12] O. Iloanusi, A. Gyaourova, and A. Ross, "Indexing fingerprints using minutiae quadruplets," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2011, pp. 127–133.

[13] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity Search: The Metric Space Approach*. New York, NY, USA: Springer-Verlag, 2006.

[14] U. Jayaraman, S. Prakash, and P. Gupta, "Indexing multimodal biometric databases using Kd-tree with feature level fusion," in *Information Systems Security* (Lecture Notes in Computer Science), Berlin, Germany: Springer-Verlag, 2008, pp. 221–234.

[15] P. Mansukhani, S. Tulyakov, and V. Govindaraju, "A framework for efficient fingerprint identification using a minutiae tree," *IEEE Syst. J.*, vol. 4, no. 2, pp. 126–137, Jun. 2010.

[16] A. Gyaourova and A. Ross, "Index codes for multibiometric pattern retrieval," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 518–529, May 2012.

[17] P. Indyk, *Nearest Neighbors in High-Dimensional Spaces*. Boca Raton, FL, USA: CRC Press, 2003.

[18] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.

[19] H. J. Wolfson and I. Rigoutsos, "Geometric hashing: An overview," *IEEE Comput. Sci. Eng.*, vol. 4, no. 4, pp. 10–21, Oct./Dec. 1997.

[20] R. S. Germain, A. Califano, and S. Colville, "Fingerprint matching using transformation parameter clustering," *IEEE Comput. Sci. Eng.*, vol. 4, no. 4, pp. 42–49, Oct./Dec. 1997.

[21] B. Bhanu and X. Tan, "Fingerprint indexing based on novel features of minutiae triplets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 616–622, May 2003.

[22] X. Liang, A. Bishnu, and T. Asano, "A robust fingerprint indexing scheme using minutia neighborhood structure and low-order Delaunay triangles," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 4, pp. 721–733, Nov. 2007.

[23] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.

[24] J. Ji, S. Yan, J. Li, G. Gao, Q. Tian, and B. Zhang, "Batch-orthogonal locality-sensitive hashing for angular similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 1963–1974, Oct. 2014.

[25] F. Hao, J. Daugman, and P. Zielinski, "A fast search algorithm for a large fuzzy database," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 2, pp. 203–212, Jun. 2008.

[26] F. Yue, B. Li, M. Yu, and J. Wang, "Hashing based fast palmprint identification for large-scale databases," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 5, pp. 769–778, May 2013.

[27] R. Cappelli, M. Ferrara, and D. Maltoni, "Fingerprint indexing based on minutia cylinder-code," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 1051–1057, May 2011.

[28] R. Cappelli and M. Ferrara, "A fingerprint retrieval system based on level-1 and level-2 features," *Expert Syst. Appl.*, vol. 39, no. 12, pp. 10465–10478, 2012.

[29] A. A. Paulino, E. Liu, K. Cao, and A. K. Jain, "Latent fingerprint indexing: Fusion of level 1 and level 2 features," in *Proc. IEEE 16th Int. Conf. Biometrics, Theory, Appl., Syst.*, Sep./Oct. 2013, pp. 1–8.

[30] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reason.*, vol. 50, no. 7, pp. 969–978, Jul. 2009.

[31] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang, "Inductive hashing on manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1562–1569.

[32] X. Bai, H. Yang, J. Zhou, P. Ren, and J. Cheng, "Data-dependent hashing based on p-stable distribution," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5033–5046, Dec. 2014.

[33] J. He, S. Kumar, and S.-F. Chang, "On the difficulty of nearest neighbor search," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 1127–1134.

[34] P. D. Gutierrez, M. Lastra, F. Herrera, and J. M. Benitez, "A high performance fingerprint matching system for large databases based on GPU," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 1, pp. 62–71, Jan. 2014.

[35] Y. Zang, X. Yang, X. Jia, N. Zhang, J. Tian, and J. Zhao, "Evaluation of minutia cylinder-code on fingerprint cross-matching and its improvement with scale," in *Proc. IEEE Int. Conf. Biometrics*, Jun. 2013, pp. 1–6.

[36] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.

[37] S. Z. Li, *Markov Random Field Modeling in Image Analysis*, 3rd ed. London, U.K.: Springer-Verlag, 2009.

[38] A. Blake, P. Kohli, and C. Rother, Eds., *Markov Random Fields for Vision and Image Processing*. Cambridge, MA, USA: MIT Press, 2011.

[39] S. C. Dass, "Markov random field models for directional field and singularity extraction in fingerprint images," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1358–1367, Sep. 2004.

[40] R. K. N. V. Rama and A. M. Namboodiri, "Fingerprint enhancement using hierarchical Markov random fields," in *Proc. IEEE Int. Joint Conf. Biometrics*, Oct. 2011, pp. 1–8.

[41] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-5, no. 1, pp. 25–39, Aug. 1983.

[42] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *J. Roy. Statist. Soc., B (Methodological)*, vol. 36, no. 2, pp. 192–236, 1974.

[43] A. J. Dobson and A. G. Barnett, *An Introduction to Generalized Linear Models*, 3rd ed. Boca Raton, FL, USA: CRC Press, 2008.

[44] Y. Wang, U. Naumann, S. T. Wright, and D. I. Warton, "mvabund—An R package for model-based analysis of multivariate abundance data," *Methods Ecol. Evol.*, vol. 3, no. 3, pp. 471–474, Jun. 2012.

[45] Y. Wang, L. Wang, Y.-M. Cheung, and P. C. Yuen, "Fingerprint geometric hashing based on binary minutiae cylinder codes," in *Proc. 22nd Int. Conf. Pattern Recognit.*, Stockholm, Sweden, Aug. 2014, pp. 690–695.

[46] G. Bebis, T. Deaconu, and M. Georgiopoulos, "Fingerprint identification using Delaunay triangulation," in *Proc. IEEE Int. Conf. Inf. Intell. Syst.*, Bethesda, MD, USA, Oct. 1999, pp. 452–459.

[47] Biometric System Laboratory at University of Bologna. *Minutiae Cylinder-Code SDK*. [Online]. Available: http://biolab.csr.unibo.it/, accessed Sep. 16, 2014.

[48] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, 2nd ed. London, U.K.: Springer-Verlag, 2009.

[49] *NIST Special Database 14: Mated Fingerprint Cards Pairs 2 Version 2*. [Online]. Available: http://www.nist.gov/srd/nistsd14.cfm, accessed Sep. 16, 2014.

[50] B. Dorizzi *et al.*, "Fingerprint and on-line signature verification competitions at ICB 2009," in *Proc. 3rd Int. Conf. Biometrics*, Alghero, Italy, Jun. 2009, pp. 725–732.

Yi Wang (S'05–M'09) received the B.E. degree in electronics and information engineering from the South China University of Technology, Guangzhou, China, in 2002, the M.E. degree in telecommunications engineering from the University of Melbourne, Melbourne, Australia, in 2003, and the Ph.D. degree in computer science from RMIT University, Melbourne, in 2009.

She was with the School of Mathematics and Statistics, The University of New South Wales, Sydney, Australia, from 2009 to 2012, as a Post-Doctoral Research Associate. In 2012, she joined Hong Kong Baptist University, Hong Kong, as a Research Assistant Professor with the Department of Computer Science. Her research interests include biometrics and statistical pattern recognition.

Lipeng Wang received the M.Sc. degree in computer science from Sichuan University, Chengdu, China, in 2012. In 2013, he joined the Institute of Computational and Theoretical Studies, Hong Kong Baptist University, as a Research Assistant. He is currently with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His research interests include biometric indexing, information security, and image processing.

Yiu-Ming Cheung (SM'06) received the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, in 2000. He is currently a Full Professor with the Department of Computer Science, Hong Kong Baptist University. His research interests include machine learning, pattern recognition, visual computing, and optimization. He is the Founding Chairman of Computational Intelligence Chapter of the IEEE Hong Kong Section. He is also a Senior Member of the Association for Computing Machinery.

Pong C. Yuen received the B.Sc. (Hons.) degree in electronic engineering from The City University of Hong Kong, in 1989, and the Ph.D. degree in electrical and electronic engineering from The University of Hong Kong, in 1993. He joined the Hong Kong Baptist University, in 1993, where he is currently a Professor and the Head of the Department of Computer Science.

Dr. Yuen was a recipient of the University Fellowship to visit The University of Sydney in 1996. In 1998, he spent a 6-month sabbatical leave with The University of Maryland Institute for Advanced Computer Studies (UMIACS), University of Maryland at College Park. From June 2005 to January 2006, he was a Visiting Professor with the GRAVIR Laboratory (GRAphics, VIsion and Robotics) of INRIA Rhone Alpes, France. He was the Director of Croucher Advanced Study Institute (ASI) on biometric authentication in 2004 and the Director of Croucher ASI on Biometric Security and Privacy in 2007.

Dr. Yuen has been actively involved in many international conferences as an Organizing Committee and/or a Technical Program Committee Member. He was the Track Cochair of International Conference on Pattern Recognition in 2006 and the Program Cochair of the IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems in 2012. He also serves as an Advisory Board Member of the BTAS conference 2015. He is an Editorial Board Member of *Pattern Recognition* and an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, and *SPIE Journal of Electronic Imaging*. He is also serving as a Hong Kong Research Grant Council Engineering Panel Member.

Dr. Yuen's current research interests include video surveillance, human face recognition, and biometric security and privacy.