Towards Bloom Filter-based Indexing of Iris Biometric Data

C. Rathgeb, H. Baier, C. Busch
da/sec - Biometrics and Internet Secrity Reserach Group
Hochschule Darmstadt, Germany
{christian.rathgeb,harald.baier,christoph.busch}@h-da.de

F. Breitinger
UNHcFREG - Cyber Forensics Research and Education Group
University of New Haven, CT, USA
fbreitinger@newhaven.edu

Abstract

Conventional biometric identification systems require exhaustive $1:N$ *comparisons in order to identify biometric probes,* i.e. *comparison time frequently dominates the overall computational workload. Biometric database indexing represents a challenging task since biometric data is fuzzy and does not exhibit any natural sorting order.*

In this paper we present a preliminary study on the feasibility of applying Bloom filters for the purpose of iris biometric database indexing. It is shown, that by constructing a binary tree data structure of Bloom filters extracted from binary iris biometric templates (iris-codes) the search space can be reduced to $\mathcal{O}(\log N)$. *In experiments, which are carried out on a database of* $N = 256$ *classes, biometric performance (accuracy) is maintained for different conventional identification systems. Further, perspectives on how to employ the proposed scheme on large-scale databases are given.*

1. Introduction

Biometrics in particular, technologies of iris recognition [7, 3] represent a rapidly evolving field of research and national-sized biometric systems are already deployed, *e.g.* the Unique IDentification Authority of India (UIDAI) [24], which aims at registering all 1.2 billion Indian citizens, is enrolling 1 million subjects on a daily basis. With about 700 million subjects enrolled (status October, 2014), against which the daily intake has to be compared to check for duplicate identities the daily workflow of iris cross-comparisons results in 7×10^{14}, or 700 trillion (!). Similar to performing a duplicate enrolment check, a conventional biometric identification systems requires exhaustive $1 : N$ comparisons in order to perform a single biometric identification, where N represents the number of subjects registered with the system. Consequentially, comparison speed becomes a crucial factor for any large-scale biometric deployments which should provide real-time identification.

Biometric indexing (or filtering) techniques are designed to reduce the number of candidate identities to be considered by an (iris) identification system when searching for a match in a large repository of biometric reference data (templates) [16]. Due to the fuzziness of biometric data indexing biometric databases in order to minimize the response time of the system, represents a great challenge. Focusing on iris biometrics, different approaches have been proposed in past years, *e.g.* [10] or [15]. However, the vast majority of existing schemes suffer from a significant decrease in biometric performance, *i.e.* fast identification comes at the cost of accuracy. In addition, indexing techniques are frequently based on complex data structures, where the insertion and/or deletion of subjects may result in a re-structuring of the entire dataset.

Recently, Breitinger *et al.* [5] introduced a theoretical concept of a logarithmic divide & conquer approach for similarity digests database lookup. Focusing on use cases in the field of digital forensics, *e.g.* blacklisting of files, the authors design a scheme for approximate matching that allows a file-against-set comparison with a lookup complexity of $\mathcal{O}(\log N)$. In this work the aforementioned concept which utilizes a novel Bloom filter-based hierarchical tree data structure is adapted for the purpose of iris biometric database indexing. Binary search trees based on Bloom filters are constructed from databases of binary iris biometric templates (iris-codes). By introducing an adequate comparison procedure the lookup complexity is reduced to the magnitude of $\mathcal{O}(\log N)$. In addition, the proposed search structure which requires storage space in the magnitude of $\mathcal{O}(N)$ enables an insertion of data subjects in at most $\mathcal{O}(\log N)$ steps. On the medium-sized IITDv1 iris database accuracy of open-source re-implementations of different conventional systems [13, 14] obtaining identification rates (IRs) and false match rates (FMRs) of approximately 94% and 0.75%, respectively, is maintained within a challenging open-set evaluation scenario. Furthermore, we give perspectives on the scalability of the presented approach w.r.t. large-scale applications.

This paper is organized as follows: fundamentals and related works are briefly described in Sect. 2. A detailed description of how to construct a Bloom filter-based hierarchical tree data structure is given in Sect. 3. Subsequently, the workflow of the proposed indexing approach is summarized in Sect. 4. In Sect. 5 experimental evaluations are presented. Finally, conclusions are drawn in Sect. 6.

2. Background and Related Work

2.1. Bloom Filter-based Approximate Matching

A Bloom filter [2] is represented as a binary array $\mathbf{b}$ of length 2^m, where initially all bits are set to zero. In order to represent a finite set $\mathbf{S}$ all elements $s_i \in \mathbf{S}$, $i = 1, \ldots, n$ are 'inserted' into $\mathbf{b}$ by applying k independent hash functions $h_1, h_2, \ldots, h_k$ to each $s_i \in \mathbf{S}$ and setting resulting indexes to one, $\mathbf{b}[H_j(s_i)] = 1$, with $j = 1, \ldots, k$ (hash functions generate hashes in the range $[0, ..., 2^m - 1]$). Each bit of $\mathbf{b}$ can be set to one multiple times, but only the first change has an effect. In order to perform efficient membership queries, a given element s' is hashed using the pre-defined hash functions and it is checked whether the all values of $\mathbf{b}$ at indexes $H_1(s'), H_2(s'), \ldots, H_k(s')$ are set to one. If this is the case s' can be assumed to be a member of $\mathbf{S}$ with a certain (non-trivial) probability of false positive, if not, s' is clearly not a member of $\mathbf{S}$. Bloom filters convince by their wide field of applications, *e.g.* database or network applications [17, 6].

In [5] the concept of Bloom filters is employed in order to construct an binary search tree. Given a set of $\mathbf{S}$ of files the authors suggest to map all files into a single root Bloom filter, *i.e.* chunks of files are hashed and according bits are set to one. While such a Bloom filter may require Gigabytes of storage it allows a file-against-set comparison in $\mathcal{O}(1)$ steps [4], providing a binary decision (if or if not a file is in the blacklist). Obviously, such a binary decision can also be given for a subset of $\mathbf{S}$, enabling the construction of a binary search tree. That is, the first and second half files of $\mathbf{S}$ are mapped into two separate Bloom filters representing children nodes of the root Bloom filter. This procedure is performed recursively and corresponding files identifiers are appended at leaves. In order to implement approximate matching, *i.e.* tolerating a certain variance within files, the authors suggest to define a threshold t for the number of chunks (bits in Bloom filters) which have to match

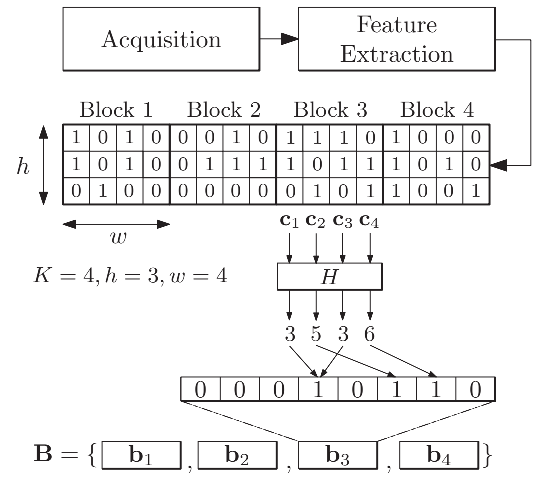


Figure 1. The concept of generating a set of Bloom filters $\mathbf{B}$ from a binary biometric feature vector consisting of $K = 4$ blocks of height $h = 3$ and width $w = 4$.

for a given file s'. Based on the concept of Bloom filters the probability of false positives is reduced at each level yielding a lookup complexity of $\mathcal{O}(\log N)$ where N represents the number of files stored in the tree. For further details on this concept the reader is referred to [5].

2.2. Fast Biometric Identification

With respect to workload reduction within biometric identification, we coarsely categorize three key approaches: (1) classification, (2) indexing, and (3) a serial combination of a computationally efficient and an accurate (but more complex) algorithm. Let N denote the number of subjects registered with a biometric system and ω be the workload for a pair-wise comparison of two templates. Then the overall workload $\mathcal{W}$ for biometric identification is defined as $\mathcal{W} = \omega N + \delta$, where δ summarizes any additional one-time costs, *e.g.* sorting of candidates. In case the entire feature space is divided into c classes (*i.e.* subsets), $\mathcal{W}$ can be reduced to $\omega N/c + \delta$, given that the registered subjects are equally distributed among all classes. For instance, in [12, 19] and [18] fingerprint and ear images are assigned to $c = 5$ and $c = 4$ classes, respectively. It is generally concluded that small intra-class variations as well as sufficient image quality represent essential preliminaries in order to achieve acceptable pre-selection error rates [12].

Biometric indexing aims at reducing the overall workload in terms of $\mathcal{O}$-notation. While an optimal indexing scheme would require a workload in $\mathcal{O}(1)$, existing approaches focus on reducing the workload to at least $\mathcal{O}(\log N)$, yielding $\mathcal{W} = \omega \log(N)$. In the majority of cases this is achieved by introducing hierarchical search structures which tolerate a distinct amount of biometric variance. Focusing on iris biometric indexing Hao *et al.* [10] proposed a fast search algorithm for iris-codes based on Beacon Guided Search combining a multiple colliding segments principle and early termination strategy. The technique is evaluated using 632 500 iris-codes enrolled in the United Arab Emirates (UAE) border control system, showing a substantial improvement in search speed with a negligible loss of accuracy. In [16] two techniques for indexing iris-codes as well as iris textures are proposed. On the CASIAv3 database the search space is reduced to ∼30% for both schemes, yielding rather unpractical recognition rates of ∼85%. In [8] Burrows-Wheeler transform is applied to index iris images, which further reduces the search space on the same dataset to 8% at comparable accuracy. In [21] small biometric keys are generated from iris textures which are used as starting position within a Karnaugh map-based search structure. Reducing the search space to 3% on the CASIAv3 database the authors report an accuracy of ∼90%, however, the employed data structure requires several Gigabytes of additional storage. Mehrotra *et al.* [15] presented an indexing scheme for iris textures using Energy Histogram of DCT subbands. The authors report a trade-off between the number of employed subbands and the resulting accuracy. For practical accuracy the search space is reduced to ∼35% for three medium-sized databases.

Within serial combinations computationally efficient algorithms are used to extract a short-list of $\mathcal{L}N$ most likely candidates, with $\mathcal{L} \ll 1$. Therefore, $\mathcal{W}$ is reduced to $\hat{\omega}N + \omega\mathcal{L}N$, where $\hat{\omega}$ is the workload of a pair-wise comparison of the computationally efficient algorithm, $\hat{\omega} \ll \omega$. In other words, identification is accelerated if $\omega(1-\mathcal{L}) > \hat{\omega}$ holds. In [9] and [1] $\mathcal{L}$ was reduced to ∼10% for iris and voice, respectively, significantly accelerating biometric identification. Compared to indexing and classification a serial combination of algorithms enables a more accurate operation of the resulting trade-off between computational effort and accuracy by setting an adequate threshold for $\mathcal{L}$.

3. Bloom Filter-based Tree Data Structure

3.1. Bloom Filter Generation

Generic iris recognition systems [3] extract binary feature vectors based on a row-wise analysis of normalized iris textures, *i.e.* iris-codes typically represent two-dimensional binary feature vectors (see Fig. 4 (d)-(e)). With respect to the employed feature extraction algorithms (see Sect. 5) we divide the two-dimensional binary matrix into K blocks of equal size where each block consists of $w \times h$ bits. From each bit block we extract a Bloom filter such that the transformed iris-code $\mathbf{B}$ consists of K separate Bloom filters, $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_K\}$. In order to map one block to a Bloom filter $\mathbf{b}$ the entire sequence of columns of the block is successively transformed to decimal indexes and bits within the according Bloom filter are set to one. Instead of using multiple hash functions, we employ a single mapping $H : \{0,1\}^h \rightarrow \{0,1\}^h$, *i.e.* the image set and the in-

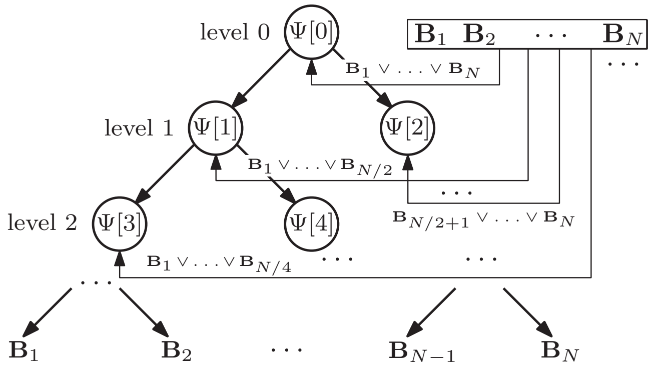

level 0
$\Psi[0]$
$\mathbf{B}_1$ $\mathbf{B}_2$ · · · $\mathbf{B}_N$
$\mathbf{B}_1 \vee \ldots \vee \mathbf{B}_N$
level 1
$\Psi[1]$
$\Psi[2]$
$\mathbf{B}_1 \vee \ldots \vee \mathbf{B}_{N/2}$
$\mathbf{B}_{N/2+1} \vee \ldots \vee \mathbf{B}_N$
level 2
$\Psi[3]$
$\Psi[4]$
$\mathbf{B}_1 \vee \ldots \vee \mathbf{B}_{N/4}$
$\mathbf{B}_1$
$\mathbf{B}_2$
$\mathbf{B}_{N-1}$
$\mathbf{B}_N$

Figure 2. Construction of the Bloom filter-based binary search tree by ORing according Bloom filter-based templates.

verse image set of H are of equal size and no collisions occur. The transform is implemented by mapping each column $\mathbf{c}_i \in \{0,1\}^h$, $i = 1, \ldots, w$, to the index of its decimal value, which is shown in Fig. 1,

$$\mathbf{b}[H(\mathbf{c}_i)] = 1, \text{ with } H(\mathbf{c}_i) = \sum_{a=0}^{h-1} \mathbf{c}_i[a] \cdot 2^a. \qquad (1)$$

The proposed transform is alignment-free to a certain degree [20], *i.e.* generated templates do not need to be aligned at the time of comparison. Equal columns within certain blocks are mapped to identical indexes within according Bloom filters, *i.e.* self-propagating errors caused by an inappropriate alignment of iris-codes are eliminated (radial neighborhoods persist). The rotation-compensating property of the proposed system comes at the cost of location information of iris-code columns. At block boundaries miss-alignment of iris-codes will distribute a certain amount of potentially matching columns among different blocks, which would be mapped to neighboured Bloom filters [20]. The entire procedure builds upon the scheme proposed in [20] designed for biometric template protection [22].

3.2. Tree Generation

For the entire dataset consisting of N subjects we generate sets of Bloom filters $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_N$ of size $K \times 2^h$ bits. Based on these sets of Bloom filters we generate the binary search tree $\mathbf{\Psi}$ of height $\log_2 N+1$, comprising $\log N$ levels, a total number of $N-1$ nodes and N leaves (w.l.o.g. we assume this tree to be a full binary tree, *i.e.* N is a power of 2). Since nodes as well as leaves consist of sets of Bloom filters of equal size the entire tree requires $(2N-1)\times K\times 2^h$ bits of storage. We interpret $\mathbf{\Psi}$ as a level-wise sequence of sets of Bloom filters, *i.e.* $\mathbf{\Psi}[0]$ represents the root node, $\mathbf{\Psi}[1]$ represents the left child of the root node, and so forth.

In the first step the root node is constructed as the union of all sets of filters, $\mathbf{\Psi}[0] = \mathbf{B}_1 \vee \mathbf{B}_2 \vee \cdots \vee \mathbf{B}_N$. Note that the union of all sets of filters corresponds to a mapping of all columns of iris-codes of all subjects to a single set of

Bloom filters. In the second step both child nodes $\mathbf{\Psi}[1]$ and $\mathbf{\Psi}[2]$ are constructed as $\mathbf{\Psi}[1] = \mathbf{B}_1 \vee \mathbf{B}_2 \vee \cdots \vee \mathbf{B}_{N/2}$ and $\mathbf{\Psi}[2] = \mathbf{B}_{N/2+1} \vee \mathbf{B}_{N/2+2} \vee \cdots \vee \mathbf{B}_N$. That is, we define the i-th node/ leaf as,

$$\mathbf{\Psi}[i] = \mathbf{B}_{N/2^l(i+1 \mod 2^l)} \vee \cdots \vee \mathbf{B}_{N/2^l(i+1 \mod 2^l)+N/2^l}, \quad (2)$$

where l represents the current level within the tree, $l = 0, \ldots, \log N$. In case of uniformly distributed data the fraction of bits expected to be set to one within Bloom filters at level l is $1 - (1 - 1/2^h)^{wN/2^l}$. For instance, at the level 0 (root) $1 - (1 - 1/2^h)^{wN}$ are expected to be one while at level 1 (children of root) only $1 - (1 - 1/2^h)^{wN/2}$ bits are expected to be one. Focusing on biometric data we expect a reduction in w since bits within neighbouring columns of bits are not mutual independent [7]. An example of such a Bloom filter-based binary tree is schematically depicted in Fig. 2.

4. Bloom Filter-based Indexing

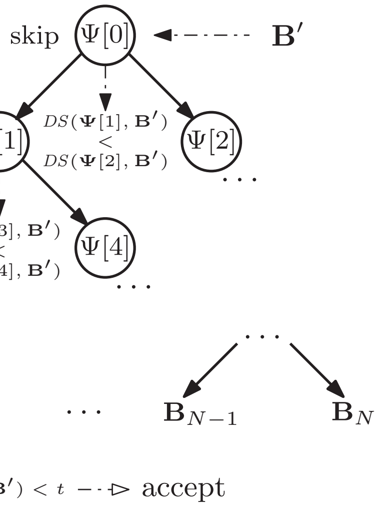
4.1. Lookup Strategy

The comparison between two Bloom filter-based templates $\mathbf{B}$ and $\mathbf{B}'$ is implemented as a score level fusion of all pairwise comparisons of according Bloom filters, $\mathbf{b}_i$, $\mathbf{b}'_i$, $i = 1, \ldots, K$. Since Bloom filters comprise a variable amount of ones (depending on the number of identical columns within processed bit blocks) we estimate the (dis)similarity DS between two Bloom filters $\mathbf{b}$ and $\mathbf{b}'$ as,

$$DS(\mathbf{b}, \mathbf{b}') = \frac{|\mathbf{b} \oplus \mathbf{b}'|}{|\mathbf{b}| + |\mathbf{b}'|}, \quad (3)$$

where the XOR operator counts the number of disagreeing bits which is normalized by the Hamming weight of both Bloom filters. The (dis)similarity between two sets of Bloom filters, $\mathbf{B}$ and $\mathbf{B}'$, is defined as $1/K \sum_{i=1}^{K} DS(\mathbf{b}_i, \mathbf{b}'_i)$.

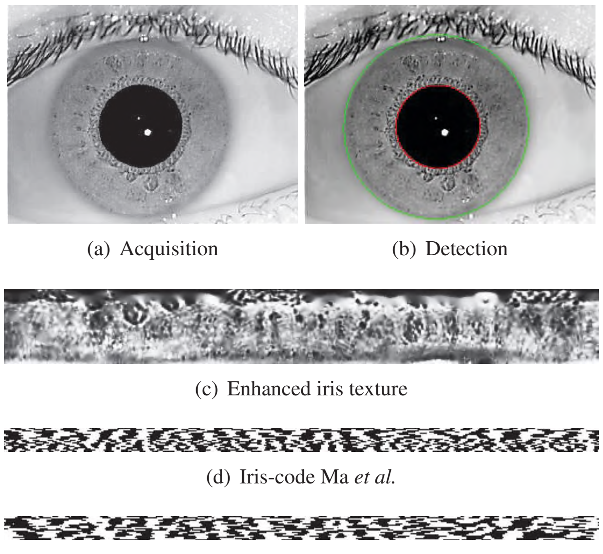
Obviously, the proposed comparator can be applied to compare a Bloom filter-based template, extracted from a single iris-code, to nodes or leaves of the Bloom filter-based binary search tree $\mathbf{\Psi}$, since all nodes and leaves consist of sets of Bloom filters of equal size. However, as mentioned in Sect. 2, in case of nodes $\mathbf{\Psi}[i], i \in \{0, \ldots, N-2\}$, Bloom filters represent a union of Bloom filter-based templates extracted from more than one iris-code, *i.e.* false positives may occur. In other words, when comparing a probe template $\mathbf{B}'$ to a node of $\mathbf{\Psi}$ matching bits may originate from Bloom filter-based templates extracted from iris-codes of subjects other than the one searched for. This false positive rate of matching bit is non-trivial and depends on the amount of Bloom filter-based templates ORed for a distinct node. In order to overcome this issue, for each node of $\mathbf{\Psi}[i]$, $i = 0, \ldots, N-1$ we estimate DS-scores for both child nodes $\mathbf{\Psi}[i+2^l]$ and $\mathbf{\Psi}[i+2^l+1]$ and identify the minimum of the obtained scores, $\min(DS(\mathbf{\Psi}[i+2^l], \mathbf{B}'), DS(\mathbf{\Psi}[i+2^l+1], \mathbf{B}'))$ as the correct direction. This means, we skip the root node $\mathbf{\Psi}[0]$ and the complexity of a look-up increases to $\mathcal{O}(2\log N - 1)$, which is in the same complexity class as $\mathcal{O}(\log N)$. Due to potential false positives, DS-scores are expected to increase at each level, *i.e.* we further require the obtained sequence of scores to constantly increase. In case a DS-score decreases compared to the score obtained at the previous node we identify the retrieval as an impostor attempt. Such a condition will most likely occur at the very last levels before returning a leaf.


skip
$\Psi[0]$
$\mathbf{B}'$
$\Psi[1]$
$DS(\mathbf{\Psi}[1], \mathbf{B}')$ $<$ $DS(\mathbf{\Psi}[2], \mathbf{B}')$
$\Psi[2]$
$\Psi[3]$
$DS(\mathbf{\Psi}[3], \mathbf{B}')$ $<$ $DS(\mathbf{\Psi}[4], \mathbf{B}')$
$\Psi[4]$
$\mathbf{B}_1$
$DS(\mathbf{B}_1, \mathbf{B}')$ $\geq$ $DS(\mathbf{B}_2, \mathbf{B}')$
$\mathbf{B}_2$
$\mathbf{B}_{N-1}$
$\mathbf{B}_N$
$DS(\mathbf{B}_2, \mathbf{B}') < t$ accept
Figure 3. Basic operation mode of the proposed lookup strategy for a positive identification attempt (in addition, all "winning scores" are stored and are required to constantly increase).

Once a leaf is reached the sequence of obtained DS-scores may be arbitrarily high, *i.e.* impostor-retrievals may reach leaves in case DS-scores obtained down the path of the binary tree are constantly high. To solve this issue we analyse iris-codes of applied feature extraction algorithms within a training stage. Based on a disjoint training set of iris-codes we extract Bloom filter-based templates and perform all possible impostor comparisons and store the best score, *i.e.* the smallest DS-score, as threshold t. This threshold is used as a final decision threshold once a leaf is returned (note that we consider an open-set scenario), *i.e.* the DS-score between a Bloom filter-based template of a leaf and a given probe template $\mathbf{B}$ has to be smaller than t in order to achieve a positive identification. That is, the equation $DS(\mathbf{\Psi}[i], \mathbf{B}') < t$, $i \in \{N-1, \ldots, 2N-1\}$, has to hold. The proposed lookup strategy is schematically depicted in Fig. 3.

4.2. Insertion and Deletion

The insertion of a Bloom filter-based template into the proposed binary search tree can be efficiently handled in $\mathcal{O}(\log N)$ steps. This is done by OR-ing the according tem-


(a) Acquisition (b) Detection

(c) Enhanced iris texture

(d) Iris-code Ma *et al.*

(e) Iris-code 1-D Log-Gabor filter

Figure 4. Iris detection, pre-processing, and applied feature extraction for image `001-01` of the IITDv1 Iris Database.

plate with all corresponding nodes and inserting the template as an additional leaf. In case another level is added, one leaf becomes a node and the template of this leaf is inserted to another leaf.

In contrast, the deletion of a Bloom filter-based template from the binary search tree is non-trivial. Since bits within nodes in the path down to the according leaf may also originate from iris-codes of other subjects, deletion has to be performed offline. This means, in order to delete a Bloom filter-based template from the tree, the entire tree has to be replace by a tree generated from all remaining leaves requiring $\mathcal{O}(N \log N)$ steps. However, we do not consider this as a critical issue since in contrast to insertion deletion may not be required to be performed in real-time.

5. Experiments

5.1. Experimental Setup

Experiments are carried out using the IITD Iris Database version 1.0[1] which comprises 2,240 NIR iris images of 320×240 pixels from 224 different subjects. For each subject the first five iris images were acquired from the left eye while the remaining five images were acquired from the right eye, yielding a total number of 448 classes. In order to perform an open set identification the database is partitioned into a set of genuine and impostor classes, as well as a training set which is employed to obtain an adequate threshold t for according feature extraction algorithms. For genuine classes the first sample is used for enrolment and the remaining four samples for identification. Table 1 summarizes employed sets according to their size and the resulting number of retrievals.

Table 1. Datasets employed for evaluation and training.

Set	No. Classes	No. Samples	No. Retrievals
Genuines	256	1,280	1,024
Impostor	160	800	800
Training	32	160	–

Biometric performance is evaluated in terms of (true-positive) identification rate (IR) and false match rate (FMR) [11]. The IR of a biometric system defines the proportion of identification transactions by subjects enrolled in the system in which the subject's correct identifier is the one returned. The FMR defines the proportion of zero-effort impostor attempt samples falsely declared to match the compared non-self template. Further, we report the average amount of comparisons required for genuine and impostor retrievals, which is commonly referred to as penetration rate (PR).

At pre-processing the iris of a given sample image is detected, un-wrapped to an enhanced rectangular texture of 512×64 pixel, shown in Fig. 4 (a)-(c) applying the weighted adaptive Hough algorithm proposed in [23]. In the feature extraction stage custom implementations[2] of two different iris recognition algorithms are employed where normalized iris textures are divided into stripes to obtain 10 one-dimensional signals, each one averaged from the pixels of 5 adjacent rows (the upper 512×50 rows are analysed). The first feature extraction method follows an implementation proposed by Ma *et al.* [13] (DW) based on a dyadic wavelet transform and the second follows the 1D-LogGabor feature extraction algorithm of Masek [14] (LG). For further details on the employed feature extraction algorithms the reader is referred to [20]. Both feature extraction techniques generate iris-codes of 512×20 = 10,240 bit. Sample iris-codes generated by both feature extraction methods are shown in Fig. 4 (d)-(e).

We compare the proposed technique to a conventional identification system which calculates N Hamming distance-based comparison scores, applying ± 8 circular bit shifts in each direction for the purpose of feature alignment. Therefore, we only choose settings where the amount of bit comparisons at each level is less than 17×10,240. Accordingly, we set $h = 10, 11, 12$ and $w = 8, 16, 32, 64$, *i.e.* w.r.t. the chosen heights we process the upper and lower half of a given iris-code separately, in case $h > 10$ columns of both halves overlap. Further, we analyse the biometric performance obtained by using different tree sizes w.r.t. to the number of leaves in particular, $L = 64, 128, 256$. Obviously, the maximum considered number of $N = 256$

[1]IITD Iris Database version 1.0, `http://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database_Iris.htm`

[2]USIT – University of Salzburg Iris Toolkit v1.0, `http://www.wavelab.at/sources/`

Table 2. Identification rates, penetration rates and false match rates for the DW algorithm for different parameter settings.

w	h	L	IR	PR Gen.	FMR	PR Imp.
8	10	64	97.656	4×11.937	0.375	4×8.377
8	10	128	97.265	2×13.937	0.375	2×10.352
8	10	256	92.480	1×15.908	0.375	1×12.335
16	10	64	98.046	4×11.921	0.250	4×8.210
16	10	128	97.460	2×13.906	0.250	2×10.340
16	10	256	91.992	1×15.865	0.250	1×12.232
16	11	64	98.046	4×11.945	0.375	4×8.332
16	11	128	97.460	2×13.894	0.375	2×10.352
16	11	256	92.675	1×15.910	0.250	1×12.410
32	10	64	98.046	4×11.921	0.125	4×8.350
32	10	128	94.140	2×13.812	0.250	2×10.205
32	10	256	83.300	1×15.585	0.250	1×12.282
32	11	64	98.046	4×11.929	0.375	4×8.345
32	11	128	96.093	2×13.871	0.500	2×10.300
32	11	256	90.136	1×15.824	0.625	1×12.407
32	12	64	98.046	4×11.960	0.875	4×8.625
32	12	128	97.656	2×13.953	1.000	2×10.572
32	12	256	92.089	1×15.890	1.250	1×12.520
64	10	64	93.359	4×11.812	0.125	4×8.457
64	10	128	83.789	2×13.484	0.250	2×10.447
64	10	256	59.765	1×14.787	0.250	1×12.412
64	11	64	96.093	4×11.882	0.375	4×8.812
64	11	128	90.039	2×13.722	0.250	2×10.580
64	11	256	76.757	1×15.382	0.375	1×12.535
64	12	64	97.265	4×11.921	0.875	4×8.825
64	12	128	93.750	2×13.808	0.375	2×10.587
64	12	256	87.792	1×15.757	0.500	1×12.600

Table 3. Identification rates, penetration rates and false match rates for the LG algorithm for different parameter settings.

w	h	L	IR	PR Gen.	FMR	PR Imp.
8	10	64	97.265	4×11.921	0.125	4×9.347
8	10	128	97.656	2×13.917	0.375	2×11.265
8	10	256	92.382	1×15.878	0.500	1×13.220
16	10	64	98.046	4×11.953	0.750	4×9.370
16	10	128	97.656	2×13.917	0.625	2×11.090
16	10	256	91.796	1×15.890	0.625	1×12.977
16	11	64	98.046	4×11.953	1.125	4×9.337
16	11	128	97.851	2×13.945	1.000	2×11.102
16	11	256	93.457	1×15.902	0.875	1×13.082
32	10	64	98.046	4×11.929	1.000	4×9.305
32	10	128	95.507	2×13.882	1.000	2×11.205
32	10	256	87.792	1×15.769	0.750	1×13.207
32	11	64	98.046	4×11.960	1.500	4×9.462
32	11	128	97.460	2×13.941	0.875	2×11.300
32	11	256	91.406	1×15.910	1.000	1×13.350
32	12	64	98.046	4×11.937	1.750	4×9.525
32	12	128	97.265	2×13.945	1.375	2×11.322
32	12	256	92.675	1×15.902	1.250	1×13.165
64	10	64	96.484	4×11.898	1.000	4×9.552
64	10	128	91.406	2×13.800	0.875	2×11.312
64	10	256	75.488	1×15.507	0.375	1×13.335
64	11	64	95.703	4×11.906	0.750	4×9.742
64	11	128	94.726	2×13.890	0.625	2×11.422
64	11	256	86.914	1×15.773	0.375	1×13.367
64	12	64	97.265	4×11.929	1.250	4×9.795
64	12	128	96.093	2×13.925	1.500	2×11.460
64	12	256	90.332	1×15.861	1.125	1×13.395

classes could also be stored in 2 or 4 separate search trees comprising 128 and 64 leaves, respectively, which increases the PR but reduces the false positive rate within these subtrees.

In the training stage all 32 classes comprising 5 iris-codes are used to perform 32×5×155=24,800 impostor cross-comparisons for all considered settings of the proposed system as well as the conventional Hamming distance-based comparator.

5.2. Performance Evaluation

For performing an $1:N$ search in corporation with an adequate decision threshold the original DW and LG system achieve a baseline performance of IRs of 94.042% and 93.750% at FMRs of 0.875% and 0.750%, respectively. The IRs, PRs and FMRs for both feature extractors w.r.t. different parameter settings are summarized in Table 2 and Table 3, respectively. As can be observed, IRs and FMRs are maintained (or even improved) for both algorithms in case of small block widths, *i.e.* $w = 8, 16, 32$. Further, IRs increase with the height of processed binary blocks. However, FMRs increase as well with larger heights in particular, $h = 12$. As can be seen, PRs are in the order of $\mathcal{O}(\log N)$, *e.g.* for mapping all classes to a single binary tree ($L = 256$) PRs for genuine identifications are reduced to approximately 15 $\simeq \mathcal{O}(2 \log N)$. In other words, only 6% of the entire dataset is considered at each genuine identification attempt. In addition, PRs for impostor identifications are further reduced, *e.g.* only 12.5 per identification attempt for the same setting. In case four sub-trees comprising $L = 64$ leaves are used instead of a single tree biometric performance is improved since false positives are reduced, however, PRs are multiplied by the number of employed sub-trees. This means the lookup complexity increases to $\mathcal{O}(N/L \log L)$.

Fig. 5 depicts the number of bits which are set to one at different levels of the Bloom filter-based binary search tree for mapping template of all classes to a single tree comprising 9 levels. As can be seen, for both feature extraction techniques the maximum capacity of the search

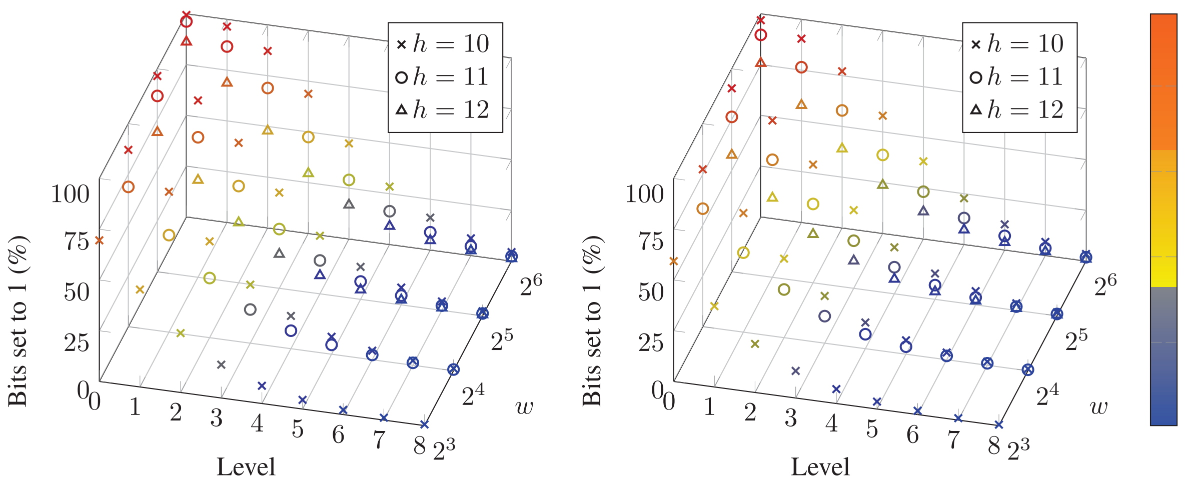


Figure 5. Amount of bits set to one (in %) for different parameter setting for mapping templates of all classes to a single Bloom filter-based binary search tree.

tree is reached relatively fast. For large block widths more columns are mapped to corresponding Bloom filters causing root nodes to comprise more than 90% of ones. For smaller block widths, *e.g.* $w = 8$, the proposed search tree is expected to be capable to store significantly more biometric templates. However, small block widths increases the storage requirements, since one Bloom filter of size 2^h bits has to be stored for each block. For dividing employed iris-codes into an upper and lower half the number of required Bloom filters can be estimated as $K = 2 \times 512/w$. Further, it is important to note that the required storage does not increase in case $N/L > 1$ sub-trees are stored instead of a single binary tree, since the total amount of storage is $N/L \times 2 \times 512/w \times 2^h \times (2L-1)$ bits. For the used parameters the required storage for both feature extractors is below 8MB, for the settings $w = 8$ with $h = 10$, $w = 16$ with $h = 11$, and $w = 32$ with $h = 12$. For setting which would require more amount of storage the amount of bit comparisons per node or leaf would exceed that of a conventional comparison of iris-codes. This means, w.r.t. storage requirements the presented scheme is scalable to any desired number of registered subjects. However, in case the maximum capacity of Bloom filter-based search tree is reached, *i.e.* too many bits are set to one within the first level, comparison decision can not be estimated reliably. In other words, depending on the employed feature extraction, an increasing number of registered subjects will force the construction of another sub-tree, which increases look-up complexity to $\mathcal{O}(N \log N)$, as previously mentioned.

6. Conclusions and Future Work

In this paper we proposed a Bloom filter-based binary search tree for fast indexing of iris biometric data. It is shown that the presented technique is capable to reduce the lookup complexity to $\mathcal{O}(\log N)$ maintaining the biometric performance obtained in a conventional $1 : N$ open set identification scenario. While the proposed scheme is evaluated on a medium-sized database comprising $N = 256$ enrolled classes, achieving a PR of approximately 6%, it is scalable with respect to storage requirements. In case larger datasets are employed these can be mapped to different sub-trees comprising templates of $N = 256$ subjects. For instance, in case of $N = 4,096$ a total number of $N/L = 16$ sub trees could be used, holding PRs constantly at $N/L \log L = 6\%$. Compared to existing iris-biometric indexing schemes which are applied to comparable datasets, *e.g.* [16, 15], the proposed scheme is evaluated in a more challenging open set scenario maintaining practical IRs. The proposed search structure is balanced since properties of iris-codes are not considered during insertion, *i.e.* there are no significant deviations between PRs obtained for different subjects. Finally, it is important to note that presented concept is evaluated for open-source iris recognition algorithms yielding fully reproducible research.

Future work comprises a more thorough analysis of biometric data in order to set adequate decision thresholds at all levels of employed search trees. This would enable a rejection of impostors at very first levels which would reduce the lookup complexity in case several sub-trees are used and would provide a efficient duplicate enrolment check.

Acknowledgements

This work was partially funded by the European FP7 FIDELITY project (SEC-2011-284862) and the Center for Advanced Security Research Darmstadt.

References

[1] S. Billeb, C. Rathgeb, M. Buschbeck, H. Reininger, and K. Kasper. Efficient two-stage speaker identification based in universal background models. In *Intl. Conference of the Biometrics Special Interest Group*, 2014.

[2] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, 1970.

[3] K. W. Bowyer, K. Hollingsworth, and P. J. Flynn. Image understanding for iris biometrics: A survey. *Comp. Vis. Image Underst.*, 110(2):281 – 307, 2008.

[4] F. Breitinger, H. Baier, and D. White. On the database lookup problem of approximate matching. *Digital Investigation*, 11, Supplement 1(0):S1 – S9, 2014. Proceedings of the First Annual DFRWS Europe.

[5] F. Breitinger, C. Rathgeb, and H. Baier. An efficient similarity digests database lookup a logarithmic divide and conquer approach. *Journal of Digital Forensics, Security and Law*, 9(2):155–166, 2014.

[6] A. Broder and M. Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1(4):485–509, 2005.

[7] J. Daugman. How iris recognition works. *IEEE Trans. Circ. and Syst. for Video Techn.*, 14(1):21–30, 2004.

[8] R. Gadde, D. Adjeroh, and A. Ross. Indexing iris images using the burrows-wheeler transform. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6, Dec 2010.

[9] J. Gentile, N. Ratha, and J. Connell. An efficient, two-stage iris recognition system. In *Biometrics: Theory, Applications, and Systems, 2009. BTAS '09. IEEE 3rd International Conference on*, pages 1–5, Sept 2009.

[10] F. Hao, J. Daugman, and P. Zielinski. A fast search algorithm for a large fuzzy database. *Trans. Info. For. Sec.*, 3(2):203–212, June 2008.

[11] ISO/IEC TC JTC1 SC37 Biometrics. *ISO/IEC 19795-1:2006. Information Technology – Biometric Performance Testing and Reporting – Part 1: Principles and Framework*. International Organization for Standardization and International Electrotechnical Committee, Mar. 2006.

[12] A. K. Jain, S. Prabhakar, and L. Hong. A multichannel approach to fingerprint classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(4):348–359, Apr. 1999.

[13] L. Ma, T. Tan, Y. Wang, and D. Zhang. Efficient iris recognition by characterizing key local variations. *IEEE Trans. on Image Processing*, 13(6):739–750, 2004.

[14] L. Masek. Recognition of human iris patterns for biometric identification. Master's thesis, University of Western Australia, 2003.

[15] H. Mehrotra, B. Srinivas, B. Majhi, and P. Gupta. Indexing iris biometric database using energy histogram of dct subbands. In *Contemporary Computing*, volume 40 of *Communications in Computer and Information Science*, pages 194–204. Springer Berlin Heidelberg, 2009.

[16] R. Mukherjee and A. Ross. Indexing iris images. In *ICPR*, pages 1–4, 2008.

[17] J. Mullin. Optimal semijoins for distributed database systems. *IEEE Transactions on Software Engineering*, 16(5):558 –560, may 1990.

[18] A. Pflug, A. Ross, and C. Busch. 2d ear classification based on unsupervised clustering. In *International Joint Conferent on Biometrics (IJCB)*, 2014.

[19] N. Ratha, K. Karu, S. Chen, and A. Jain. A real-time matching system for large fingerprint databases. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(8):799–813, Aug 1996.

[20] C. Rathgeb, F. Breitinger, C. Busch, and H. Baier. On the application of bloom filters to iris biometrics. *IET Biometrics*, 3(1), 2013.

[21] C. Rathgeb and A. Uhl. Iris-biometric hash generation for biometric database indexing. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2848–2851, Aug 2010.

[22] C. Rathgeb and A. Uhl. A survey on biometric cryptosystems and cancelable biometrics. *EURASIP J. on Inf. Sec.*, 2011, 2011.

[23] A. Uhl and P. Wild. Weighted adaptive hough and ellipsopolar transforms for real-time iris segmentation. In *Proc. 5th Int'l Conf. on Biometrics*, pages 1–8, 2012.

[24] Unique Identification Authority of India. Aadhaar: http://uidai.gov.in/. retrieved March, 2013.