A Fast Search Algorithm for a Large Fuzzy Database

Feng Hao, John Daugman, and Piotr Zieliński

***Abstract*—In this paper, we propose a fast search algorithm for a large fuzzy database that stores iris codes or data with a similar binary structure. The fuzzy nature of iris codes and their high dimensionality render many modern search algorithms, mainly relying on sorting and hashing, inadequate. The algorithm that is used in all current public deployments of iris recognition is based on a brute force exhaustive search through a database of iris codes, looking for a match that is close enough. Our new technique, Beacon Guided Search (BGS), tackles this problem by dispersing a multitude of "beacons" in the search space. Despite random bit errors, iris codes from the same eye are more likely to collide with the same beacons than those from different eyes. By counting the number of collisions, BGS shrinks the search range dramatically with a negligible loss of precision. We evaluate this technique 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 addition, we demonstrate that the empirical results match theoretical predictions.**

***Index Terms*—Biometric search, iris scanning and recognition (BIO-IRIS), multiple colliding segments principle.**

I. Introduction

IRIS recognition is a relatively new biometric technology. As deployed publicly today, it takes an infrared image of a person's eye, isolates the iris, demodulates the pattern of iris texture into a binary iris code, and compares it exhaustively against an enrolled database for a match [2]. Due to its high accuracy, this technology has been deployed at many airports as a replacement for passports and, in particular, it is deployed at all 27 air, land, and seaports of entry into the United Arab Emirates (UAE) as a border control security system to prevent expellees from reentering the country [1].

To deploy a large-scale biometric recognition system, the first concern is the probability of false matches, which increases with the number of records enrolled in the database. Iris patterns contain a high degree of randomness, which provides the biological basis for their uniqueness. Daugman's algorithm [2], [3] is the technique used in all public deployments of iris recognition. It encodes the iris texture into a 256-byte iris code; it also produces a 256-byte mask, which excludes those iris code bits affected by eyelids, eyelashes, specular reflections, and other noise. Alternative encoding methods have been studied which represent irises by filter-bank samples (floating-point scalars) followed by various correlation measures (for an extensive recent survey, see [25] and [23]), but they were found to be excessively slow and none could complete the 2006 Iris Challenge Evaluation (ICE) [24]. The fast search algorithm that is the focus of this paper would not apply to such scalar-based iris representations. Statistical analysis reveals that the accumulative false match rate using Daugman's algorithm remains negligible even over large databases [1], [3]. To date, there have been more than a million iris codes enrolled in the UAE central database, and the UAE Ministry of Interior reports that the system has yet to make a false match [1].

The success of the UAE deployment since 2001 has encouraged even larger deployments. One such may be seen in the United Kingdom, where the Government plans to introduce biometrically enabled ID cards in 2010 [16]. Under this scheme, biometric data, including the iris codes of the 45 million citizens who are older than 16, may be stored in a central database. A similar program exists in India. The Andhra Pradesh State government has been enrolling iris codes for 80 million local people since July 2005 under a ration-card scheme and, within the first year, about 26 million people had been enrolled [17]. With the advent of large biometric databases, information retrieval and database management will become increasingly challenging problems [21].

In the iris-matching algorithm considered here, comparing two iris codes is simple; it mainly involves counting the bits that differ between two binary vectors. Iris images may be tilted to various degrees. This problem is handled by repeating the comparisons of the iris codes over a range of relative rotations [2]. Since the comparison requires no expensive nonlinear warping operations as in [4] and [18], searching iris-code databases can be quite fast. The exhaustive search (ES) method can compare about a million iris codes per second on a 3.2-GHz central processing unit (CPU) [1]. However, continual database expansion will slow down the search speed linearly, and the default solution is to use several search engines in parallel [1].

Far more severe problems arise in applications requiring that all records in a national-sized database be compared with all others in order to detect fraudulent multiple identities, which is one of the purposes of the U.K. ID card scheme ("One person, one identity" [16]). The number of cross-comparisons then scales with the square of the national population. Although this process needs only to be done over the time course of ID card issuance, its demands are still daunting. The 45 million U.K. enrollees generate about 2×10^{15} iris pair comparisons. At one million iris code comparisons per second per 3.2-GHz CPU, this would require two billion CPU seconds, which is 63 CPU years.

In the U.K. biometric ID plan with a database of 90 million iris codes, the memory management and maintenance requirements for an ES approach are also daunting. A single ES would

Manuscript received June 5, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Vijaya Kumar Bhagavatula.

The authors are with the University of Cambridge, Computer Laboratory, Cambridge CB3 0FD, U.K. (e-mail: Feng.Hao@cl.cam.ac.uk; John.Daugman@cl.cam.ac.uk; Piotr.Zielinski@cl.cam.ac.uk).

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.2008.920726

require loading $90 \times 10^6 \times 512$ bytes $=46$ GB into memory, and every update would require manipulating a file of this size. There is thus a strong motivation to try to develop some kind of indexing-based approach instead, in which each iris code (despite its fuzziness due to unavoidable measurement discrepancies) could be treated almost as an address that points directly to the identity of the person from whom it was generated.

This search problem is more general. There are a wide range of closely related applications, for instance, searching audio fingerprints [14], [15]; paper fingerprints [5]; and optical fingerprints [6]. All of these applications produce iris-code-like fuzzy data: high-dimensional binary vectors with the comparison based on the Hamming distance. In practice, there could be billions of audio fingerprints or paper fingerprints in a database [5], [14], [15]. Using the ES would require thousands of parallel machines, which is clearly infeasible under cost constraints.

We therefore set out to solve this problem at the algorithmic level, without relying on customized hardware or parallelism. Though our work focuses on searching databases of iris codes, the devised technique is generally applicable to other fuzzy search domains.

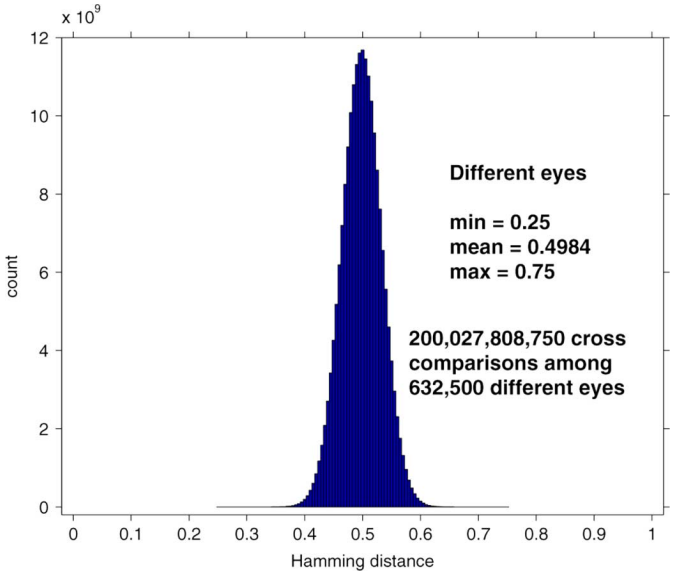

Fig. 1. Histogram of 200 billion similarity scores among all different-eye pairings across the UAE database (632 500 records)

II. Past Work

The problem we investigate is as follows: given a 2048-bit vector with random errors, how quickly can its nearest neighbor in the 2048-D Hamming space be found. A more general problem, in any metric space, is called the nearest neighbor search (NNS) [7].

NNS is defined as follows: given a set P of points in a high-dimensional space, construct a data structure which, given any query point q, finds the point p closest to q under a defined distance metric [7]. The NNS problem has been extensively studied for the past two decades. The results, however, are far from satisfactory, especially in high-dimensional spaces [10]–[12]. We will now review the past work in this line of research.

Most NNS techniques are based on the "partitioning principle" [7]. The idea is intuitive: dividing the search space into regions, so that once a query is given, only some regions are searched. Generally, the first step is to choose pivots—the reference points that divide the space. For instance, in the "ball partitioning" method, the ball center is a pivot. The ball cuts the search space into halves: inside the ball and outside. Such spheric cuts are performed recursively, leading to a balanced binary tree with pivots placed at nodes and data at leaves. An alternative method is the "generalized hyperplane partitioning," which separates the data set into two based on their relative distances to two pivot points.

The "partitioning principle" spawns many tree-like data structures. Specific techniques differ in how trees are constructed and traversed, as well as tradeoffs. For example, *m* tree, *vp* tree, *fq* tree, *mvp* tree, *mwvp* tree are based on "ball partitioning," while bisector tree *gh* tree, *gna* tree, *kd* tree, and *pcp* tree are derived from the "generalized hyperplane partitioning." Selecting the right pivots is important and not trivial. Due to the complexity of the problem, most techniques choose pivots at random [8]. Some techniques, such as the *gna* tree and *mvp* tree, employ precomputation to reduce the distance computations, but require a large amount of memory. The state of the art of the partitioning-based approach is summarized in [7].

Unfortunately, all of the aforementioned tree-like data structures succumb to the "curse of dimensionality" [10], [11], [12]. While they work reasonably well in a 2-D or 3-D space, as the dimensionality of the data increases, the query time and data storage would exhibit an exponential increase, thereby doing no better than the brute-force linear search [12]. The reason is that the intuition of dividing a space into regions no longer holds in the high-dimensional case. In the 2048-D Hamming space, for instance, the distances between different eye iris codes are sharply centered at 0.5 (see Section III-C-1); an iris code is very nearly equidistant from all the others. Hence, it is very difficult, if not impossible, to divide this space into regions. Furthermore, iris codes form no clusters in this space (as will be seen in the smooth unimodal distribution of different eye comparisons in Fig. 1), so various clustering-based NNS techniques [7] do not apply here either.

Indyk and Motwani proposed a different approach for NNS: locality sensitive hashing (LSH) [10], with the follow-on work in [11]–[13]. The core part in LSH is the definition of a hash function family $\mathcal{G}$, from which l functions $g_1, g_2 \ldots g_l$ are chosen uniformly at random. During preprocessing, every point $p \in P$ is stored in each of the l buckets, identified by $g_i(p)$. To process a query q, LSH searches all buckets $g_1(q), g_2(q) \ldots g_l(q)$ exhaustively for the nearest match. With a high probability, the nearest match exists in one of the buckets. To avoid repeated checking on the same points, the searched points are marked in memory and, thus, ignored for the following occurrences in other buckets [12], [13].

The biggest limitation facing LSH is that it often needs to search a significant percentage of the database [11]. The suggested measure is to define a threshold, and interrupt the search when there are too many points in buckets [10]–[12]. We will explain this problem in further detail in Section IV. In addition, LSH is developed based on main memory and, thus, ignores the

delay of data retrieval. However, most large-scale applications store data in databases, which are disk based.

Haitsma and Kalker proposed a similar technique and applied it to search audio fingerprints [15]. They divide a 1024-byte audio fingerprint into 256 subfingerprints of 32 bits each. They show that for a "mildly degraded" audio signal, at least one of the subfingerprints is error free. Thus, the error free one can serve as a pointer to look for possible matches, by following a 32-bit lookup table. However, it is acknowledged in this paper that "mild degradation" may be too strong of an assumption in practice. Furthermore, the 32-bit lookup table requires at least 4-GB memory, which is more than many computers can afford.

To summarize, LSH and Haitsma–Kalker's algorithm are built on essentially the same principle, which we call the "single collision principle." This principle assumes that if two records are similar, they would have a relatively high chance of having at least one "colliding" identical segment, obtained from either hashing [10]–[12] or direct sampling [15]. (Other related techniques, for example, classifying iris images into one of the two [20] or four [22] categories can be seen as working under the same principle.) However, both algorithms ignore multiple occurrences of a collision. We improve their work by introducing the "multiple colliding segments principle," and demonstrate that counting the number of colliding segments is crucial for achieving optimal performance. Applying this new principle, we are able to resolve the limitations explained before (also see [11]–[13], [15]).

III. Algorithms

A. Experiment Setup

The UAE database contains $N = 632\,500$ nonduplicate iris records. Each record includes an iris code and a mask [2]. It is assigned a unique 32-bit ID, and is stored as a 512-byte binary blob in a MySQL database [19]. The evaluation is performed on a 3-GHz PC with 3-GB memory, running FreeBSD. A Java client program retrieves the data in real time, using the Connector/J JDBC driver.

B. Exhaustive Search

With several optimizations, an ES is implemented as in Algorithm 1. First, the system retrieves the stored records into memory. We find that the fastest method is loading all data (about 320 MB) at once with one SQL query, instead of fetching one by one, which requires 632 500 SQL queries. Next, the query iris code is compared exhaustively with all retrieved records. The comparison between two iris samples is based on their smallest Hamming distance obtained from seven relative rotations. Computing the Hamming distance mainly involves counting the bits that differ between two binary vectors. A lookup table is used to speed up the counting. Furthermore, the system performs a preliminary check before fully comparing two vectors. It selects every fourth byte from two iris codes correspondingly, and counts the bits that differ. Only if less than a third of these bits disagree will ES proceed to a full comparison. Let the two iris codes be code A and code B, and their masks be mask A and mask B (see [2]). The matching decision is based on the normalized Hamming distance $\mathrm{HD}_{\mathrm{norm}}$ [1]

$$\mathrm{HD}_{\mathrm{norm}} = 0.5 - (0.5 - \mathrm{HD}_{\mathrm{raw}})\sqrt{\frac{n}{911}} \tag{1}$$

where $n = ||\mathrm{maskA} \cap \mathrm{maskB}||$, and

$$\mathrm{HD}_{\mathrm{raw}} = \frac{||(\mathrm{codeA} \oplus \mathrm{codeB}) \cap \mathrm{maskA} \cap \mathrm{maskB}||}{||\mathrm{maskA} \cap \mathrm{maskB}||}. \tag{2}$$

Algorithm 1: Exhaustive Search

Input: Query iris code q.

Output: Match iris code p or "No match".

1) ID $i = 1$ to N do.
2) $p \leftarrow$ IrisTable $[i]$.
3) **for** $k = -3, -2, \ldots, 3$ do.
4) Obtain q_k by shifting q by k bytes (precomputed).
5) IF Preliminary Check $(p), q_k)$ is OK, then
6) IF $\mathrm{HD}_{\mathrm{norm}}(p, q_k) <$ MatchThreshold, then
7) Return p.
8) Return "No match".

The normalization performed in (1) is based on the fact that the standard deviation of a fractional binomial distribution varies inversely as the square root of the number of Bernoulli trials done, which corresponds here to the number of bits mutually available for comparison between two iris codes. If this normalization was not used, then false matches could arise merely because few bits were mutually available for comparison (for example, because of excessive eyelid occlusion), just as runs of few coin tosses can spuriously yield "all heads" with good likelihood. The normalization maps all different iris-code comparisons into the same stable binomial distribution and, thus, allows the decision confidence levels to depend only on $\mathrm{HD}_{\mathrm{norm}}$, independent of how many bits were mutually available. Parameter 911 is the typical number of bits mutually available between different iris codes. Fewer than this are penalized (meaning that the degree of match required must be better, before a match is declared); and if more than 911 bits are available for comparison, then a high confidence match can be declared even with less similarity.

In the experiment, the matching threshold for $\mathrm{HD}_{\mathrm{norm}}$ is set at 0.22, which is suggested in [1] for the U.K. population. Fig. 1 shows the distribution of the normalized Hamming distances for the UAE database based on $\frac{N \times (N-1)}{2} \approx 200$ billion cross comparisons.

To find an existent match, ES searches, on average, half of the database. With Java code, the average delay of computing these Hamming distances for each given query is 2.675 s, which is slower than the C-based program that can compare one million iris codes per second [1]. This is because Java executes slower than C (which is a tradeoff for Java's portability) [19]. However, this difference is irrelevant since we aim to compare the two algorithms based on the same platform.

In addition, loading data from the database incurs a significant delay too. In the experiment, it takes 6.508 s to load 632 500 iris records into memory. Therefore, if the search returns no match, the total delay is $6.508 + 2.675 \times 2 = 11.858$ s. At first glance, it appears trivial to reduce the delay by loading all data once and holding it in memory. However, this would create problems for memory management: it is complex and expensive to maintain consistency between the data in memory and a constantly updated database of records. Hence, for an efficient search algorithm, both the search and loading times need to be substantially reduced.

C. Beacon-Guided Search

We now describe a new search algorithm called the beacon-guided search (BGS). Here, a beacon is a collection of iris-code IDs that have the same defined feature. It differs from a bucket [10] in that it is more than a storage unit; more important, it is a guiding landmark. Technically, it works as follows.

1) Preprocessing: During preprocessing, all iris codes in the database are indexed to create a beacon guiding structure (Algorithm 2). This process creates $n = 128$ beacon spaces, with 2^m $(m = 10)$ beacons in each space. An m-bit beacon index b_i uniquely identifies every beacon in the ith space. Indexing an iris code involves saving its unique 32-bit ID on one beacon per beacon space. The extra storage required is $128 \times 4 \times N = 512 \times N$ bytes, the same size as the original database.

Algorithm 2: Beacon Guided Search–Preprocessing

Input: A table of N iris codes.

Output: beacon guiding structure $\mathrm{BeaSpace}_1, \mathrm{BeaSpace}_2, \ldots, \mathrm{BeaSpace}_n (n = 128)$.

1) FOR ID $i = 1$ to N do.
2) $p \leftarrow$ IrisTable $[i]$.
3) FOR $j = 1$ to n do.
4) Compute the beacon index b_j from p.
5) Insert i into $\mathrm{BeaSpace}_j[b_j]$.

To understand the assignment of an iris code feature to a beacon index $b_i (1 \leq i \leq 128)$, we must consider the properties of iris codes more closely. The 2 048 data bits in any iris code are deemed to have varying reliability, and those deemed unreliable (e.g., affected by eyelashes) are flagged as such by the mask bits by using the logic in (2). The set bits (i.e., "1"s) in the mask indicate the positions deemed reliable, so the corresponding iris-code data bits are used in comparisons. Fig. 2(a) plots the probability of having a set data bit, across bit positions sorted by descending likelihood of the mask bit being set. It shows that unmasked iris-code bits (whether computed with the real or imaginary parts of the complex demodulating wavelets [1]–[3]) generally have an equal probability of being "0" or "1." Fig. 2(b) shows by each dot, the position within iris codes of the 1408 bits that are most often unmasked. (The 2048 possible bit positions are spooled into a 32×64 matrix for graphical convenience.) The sinusoidal structure reflects the fact that eyelids are detected and the corresponding data bits are masked out (mask bits set to "0"); the deep troughs in each cycle are caused by the upper eyelids, and the shallower troughs by the lower eyelids. As a consequence, bits positioned near the pupil are more likely to be selected as reliable and, therefore, are unmasked, than bits that are computed closer to the outer boundary of the iris. We therefore use only the lower five bits (closer to the pupil) in each radial sequence of eight, and concatenate these for two corresponding wavelets, when defining the 10 bits of each beacon.

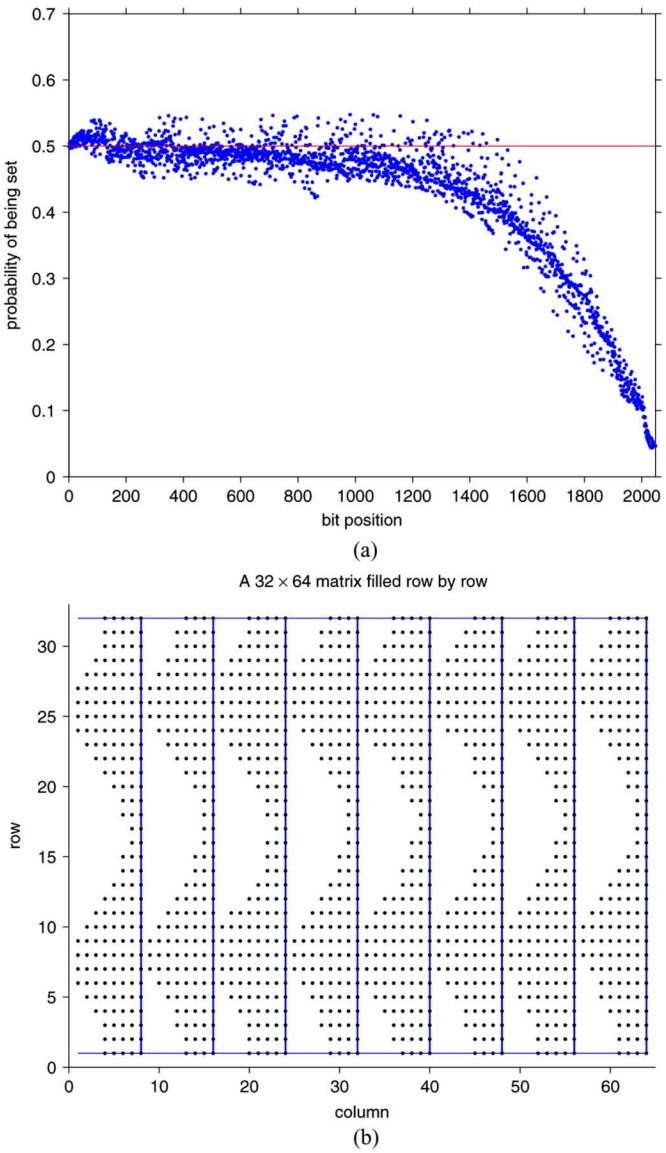

Fig. 2. Analysis of the probabilities of data bits being set and of mask bits being set, across the 2048 bit positions, based on 632 500 iris records. (a) The probability of a data bit being set to "1," sorted over all bit positions from the least likely to be masked to the most likely. (b) The 1408 bit positions that are least likely to be masked, among the 2048 iris code bit positions when spooled into a 32×64 matrix. The sinusoidal structure reflects the occlusion of upper and lower eyelids, which are detected and masked.

Based on the aforementioned observations, we compute the beacon index b_i from an iris code in two steps. First, we permute the iris code by interleaving the bytes, and then rotating the bit columns (see Fig. 3). This well separates the neighborhood bits which are strongly correlated [3]. Second, we divide the permuted vector into the consecutive 128 blocks of 2 bytes.

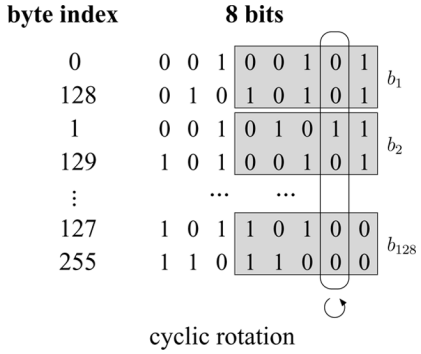

Fig. 3. Permutation by rotating bit columns.

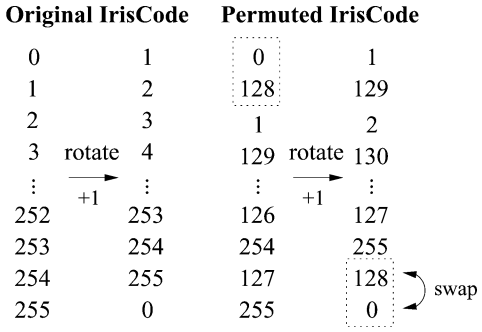

Fig. 4. Cyclic iris code after permutation.

In each block, b_i is obtained by concatenating the five least significant bits of both bytes; the selected bits correspond to the iris region near the pupil, where the obstruction by eyelashes/eyelids is unlikely, and the signal-to-noise ratio (SNR) is generally high.

The beacons derived before are cyclic. Intuitively, the eight bit columns in Fig. 3 correspond to eight concentric rings overlaying the iris region; rotating each ring by a different angle separates the bits in a byte, while still keeping the rings cyclic. Since we only extract the last five bits of a byte, it is sufficient to rotate just the last four columns. The reason for interleaving the bytes is to keep the beacon's and the iris code's rotations synchronous—rotating the iris code by one byte corresponds to shifting b_i to the next (or previous) beacon space (see Fig. 4). When b_i is shifted out of the first space and, thus, into the last space (and vice versa), the first and second halves of the bits in b_i need to be swapped to ensure the correct beacon value. Such a calibration is done in memory with a negligible delay.

After indexing all iris codes, BGS generates a beacon guiding structure, with $128 \times 2^m = 0.13$ million beacons in total. Due to the permutation, the distribution of the beacon storage density is approximately even, with an average of $N/2^m = 617$ IDs stored on each beacon. Preprocessing is fast—it took less than 5 min to index the entire UAE database.

2) Searching: Searching an iris code involves traversing the beacon spaces (Algorithm 3). In each space, BGS selects one beacon and retrieves the IDs stored in that beacon. The array $\mathrm{counter}_k$ records the accumulated occurrences of the retrieved IDs, with the subscript k referring to a particular rotation. If a specific ID has been encountered c times (e.g., $c = 3$), BGS loads the full 512-byte iris data using the ID as the primary key. The subsequent comparison is based on the same matching condition as in ES (see Algorithm 1). Note that in Algorithm 3, the computation of the space index after shifting is based on the modular operation, since the beacon spaces are cyclic.

TABLE I
COST COMPONENTS IN BEACON GUIDED SEARCH

No	Components	Compl	Operation	Cost
1	Compute beacons	$O(1)$	memory	$<$ 1ms
2	Retrieve beacon IDs	$O(n)$	disk	lightweight
3	Count collisions	$O(n)$	memory	very fast
4	Load iris codes	$O(n)$	disk	heavyweight
5	Calculate HD	$O(n)$	memory	fast

Algorithm Beacon Guided Search—Searching

Input: Query Iris code q.

Output: Match iris code p or "No match."

Preprocessed
$\mathrm{BeaSpace}_1, \mathrm{BeaSpace}_2, \ldots, \mathrm{BeaSpace}_n (n = 128)$.

1) Compute the beacon indices $b_1, b_2, \ldots, b_n$ from q.
2) FOR $i = 1$ to n.
3) FOR $k = -3, -2, \ldots, 3$ do
4) IDs $\leftarrow \mathrm{BeaSpace}_i[b_{i+k}]$.
5) FOR $j \in$ IDs do.
6) Increment $\mathrm{counter}_k[j]$ by 1.
7) IF $\mathrm{counter}_k[j] = c$
8) $p \leftarrow \mathrm{IrisTable}\,[j]$.
9) IF $\mathrm{HD}_{\mathrm{norm}}(p, q_k) <$ MatchThreshold then
10) Return p.
11) Return "No match".

There are five cost components in BGS, as summarized in Table I. The first one is to compute which beacons a given query belong to. This operation incurs a negligible delay (< 1 ms). The second one is to retrieve IDs stored in the beacons. This I/O operation is fast, since the IDs are lightweight, and can be fetched rapidly in the form of blobs (see [19]). The third operation is to count collisions. This simply involves read/write accesses to the memory array. The next is to load the full 512-byte iris data. This is an expensive input/output (I/O) operation. The final one is to fully compare the retrieved iris codes with the query. Delays in operations 2–5 increase linearly with more records enrolled into the database, giving the algorithm a linear complexity overall. Our strategy is to make the best use of the very fast operations 2 and 3, while avoiding 4 and 5. In the following section, we will explain how this goal is achieved.

IV. RESULTS

In this section, we evaluate the BGS performance both analytically and empirically.

A. Theory

The binomial probability distribution function [3] is $f(x, n, p) = n!/(x!(n - x)!)p^x(1 - p)^{n-x}$, where x is the number of successful trials, n is the total number of trials,

and p is the success probability. The binomial cumulative distribution function is defined as $F(x, n, p) = \sum_{i=0}^{x} f(i, n, p)$.

For simplicity, we assume the beacon bits are equiprobable and uncorrelated, and will address the effect of correlation later. The probability of finding a match, thus terminating the search, in the ith space depends on two conditions: 1) there had been $c-1$ collisions with the matching iris code before the ith space and 2) there is one more collision in the ith space. It is expressed as

$$P_{\text{term}} = f(c-1, i-1, (1-p_{\text{b}})^m) \times (1-p_{\text{b}})^m \quad (3)$$

where p_{b} is the bit-error rate (BER) of the query after the 7-rotation adjustment (i.e., the smallest result among seven rotations); it typically ranges from 10 to 20%.

The miss rate is the probability that the query iris code (after the seven-rotation adjustment) and the supposed match collide with less than c beacons. It is expressed as

$$P_{\text{miss}} = F(c-1, 128, (1-p_{\text{b}})^m). \quad (4)$$

In the ith space, the probability for two different iris codes to have less than c beacon collisions for all seven rotations is $F^7(c-1, i, 0.5^m)$. Here, we define the search size as the number of the iris records retrieved for comparison. When the search is terminated in the ith space, the search size can be estimated by

$$S_i = N \times (1 - F^7(c-1, i, 0.5^m)). \quad (5)$$

The value S_{128} represents the search size when a match is not found.

B. Experiment

The search size in BGS is only a fraction of the whole database. This is mainly due to the "multiple colliding segments principle" and the early termination strategy, whose effects are expressed by S_{128}/N and S_i/S_{128} $(i \leq 128)$, respectively.

First, we study the performance when a search returns no match; hence, has no early termination. We conduct the experiment using two types of queries: a string of random bits and an iris code with no match in the database. Fig. 5 summarizes the search delays.

When $c = 1$, the implementation would be similar to [10]–[12] and [15], which are based on the "single collision principle." However, a significant percentage of the database needs to be searched. Based on random queries, a theoretical estimate of the search size is 368 947, and the experiment shows 384 648. This problem might be less evident if all data are held in memory, but could prove severe for database-based applications. It is particularly slow to retrieve sporadically dispersed data from the disk; we observe that, even for fetching 10% of the records, the resulting delay would be longer than that using ES.

Table II reports the search sizes using BGS, together with the theoretical estimates S_{128} [see (5)]. The different results for the two query types are due to data correlation, for which a 2048-bit iris code has only 249 degrees of freedom [3]. Applying permutation helps select uncorrelated bits into b_i, but cannot remove

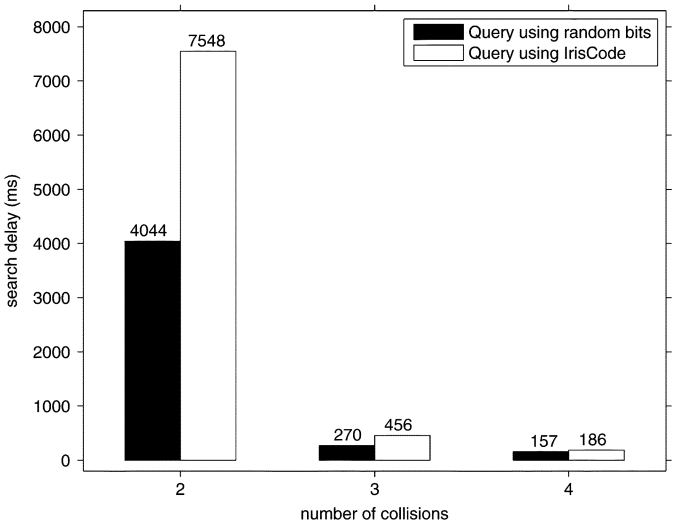

Fig. 5. Delays in BGS when no matches are found, for three different rules on the number c of beacon collisions required before a detailed comparison of the iris codes is performed.

TABLE II
SEARCH SIZES IN BGS WHEN THERE ARE NO MATCHES

	c = 2		c = 3		c = 4	
Query type	theory	experi	theory	experi	theory	experi
Random	30959	30819	1283	1280	38	38
Iris code	–	43966	–	3333	–	227

the correlation between b_i $(1 \leq i \leq 128)$. As a result, iris codes tend to cling to the same beacons in different spaces. For a query of random bits, the beacon selection is random too, which cancels the effect of correlation. On the other hand, if the query is an iris code, the beacons derived are correlated; thus, more IDs fulfill the c-collision requirement, bulging the search size.

However, the bulge in the search size has a limited effect on the search performance. When $c = 3$, it causes the search size to increase from 1280 to 3333, by a factor of 2.6. Even after bulging, the size occupies a small fraction of the database. As a result, the delay increases from 270 to 456 ms, by a factor of only 1.6.

On the other hand, correlation has almost no effect on the search accuracy. To study the variation of intraeye iris codes, we collected multiple iris samples from each of 70 eyes (more details about the data collection can be found in Section IV-C-1). From each eye, one sample is added into the database, and the rest are used as queries. Fig. 6 plots the probability of finding matches versus the BERs of the queries as well as the theoretical results $1 - P_{\text{miss}}$ [see (5)]. The theoretical and empirical data are found to be consistent. Overall, the value $c = 3$ strikes a suitable tradeoff between speed and accuracy, as we will explain in further detail in Section IV-C1.

The number of beacons in one beacon space 2^m presents another tradeoff between speed and accuracy. From (4), a smaller m leads to better error tolerance, as shown in Fig. 7. On the other hand, faster speed is achieved by choosing a bigger m, since fewer iris records need to be retrieved. Without considering early termination, the fractions S_{128}/N for $m = 8$, 10, 12 are 9.5%, 0.2%, and 0.0034%, respectively [see (5)]. Defining

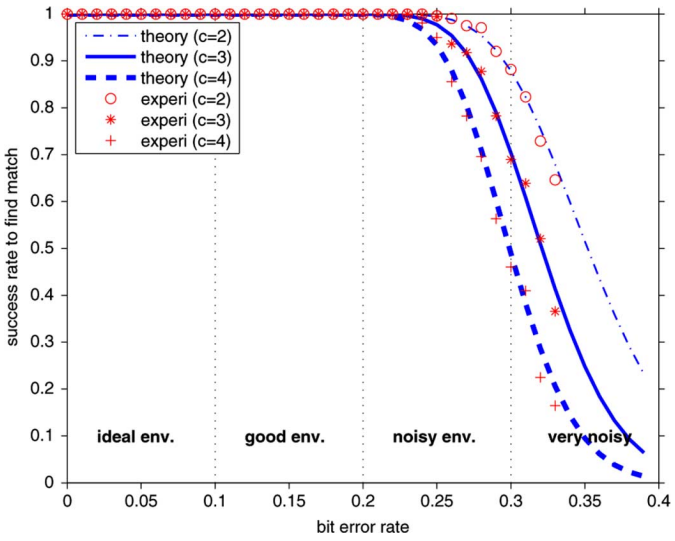

Fig. 6. Probability of finding an existing match as a function of the BER, for three different settings of c, the number of required beacon collisions.

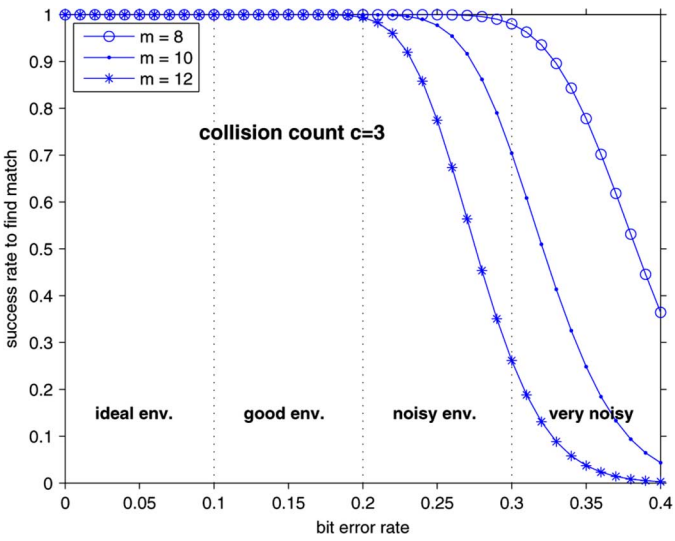

Fig. 7. Probability of finding an existing match as a function of the BER, for three different settings for the number m of bits in a beacon.

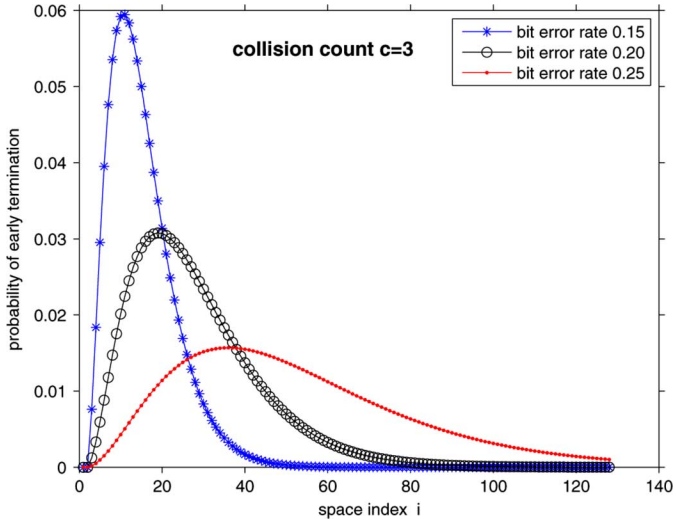

Fig. 8. Probability of finding an existing match and, thus, terminating the search at the ith beacon space $(i < 128)$, for three different BERs, when the number of required beacon collisions is $c = 3$.

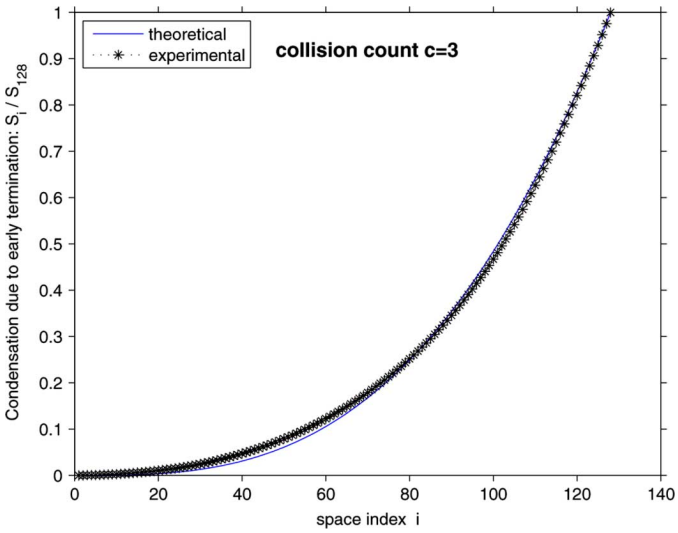

Fig. 9. Reduction in search size (condensation factor) as a function of the beacon space index i at which a criterion of $c = 3$ beacon collisions becomes satisfied.

a suitable m value may well depend on particular application requirements. In the experiment, we chose $m = 10$.

Besides the "multiple colliding segments principle," the early termination strategy also contributes to the speedup. The same strategy is commonly used in past work [7]. The fewer errors there are in a query, the more likely the match is found at an early stage. Fig. 8 plots the probability of having early termination for varying BERs, based on (3). For a query with 15% bit errors, it is statistically guaranteed that the match is found by traversing no more than 50 beacon spaces. Fig. 9 plots the theoretical value of S_i/S_{128} $(1 \leq i \leq 128)$ as well as the experimental results. The two curves fit closely, which shows that the condensation due to early termination is not affected by the data correlation.

By design, our algorithm caters for cyclic rotations of a query. The common approach in past work was trial and error: shifting the query a few times, then searching the shifted queries [14]. The problem, though, is that a slight shift of the query would remove most benefits of early termination since failed trials will incur relatively long delays. We tackle this problem by incorporating the rotations into the algorithmic design so that the search performance is rotation invariant, as shown in Fig. 10(a). Fig. 10(b) shows a cost breakdown which indicates that the delay is dominated by the cost components 2 and 3 (see Table I), with 4 and 5 diminishing to be insignificant.

C. Comparison

To make the evaluation closer to the real-world situation, we enroll more iris samples under two different conditions: favorable and noisy. The histograms of the two additional datasets are shown in Fig. 11.

In the first experiment, ten samples were collected from 70 eyes each, using the same camera and at a fixed measurement

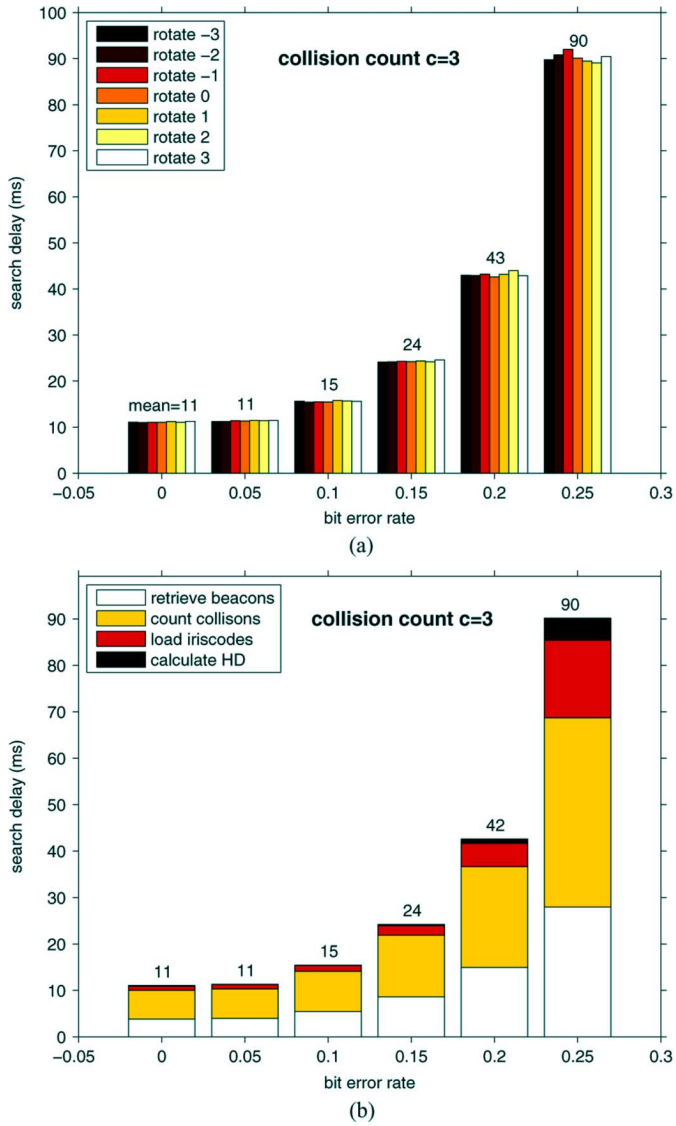

Fig. 10. Delays in BGS and their breakdown into various factors when seven cyclic rotations in iris codes are included in the search process to compensate for unknown tilts of the eye, head, or camera.

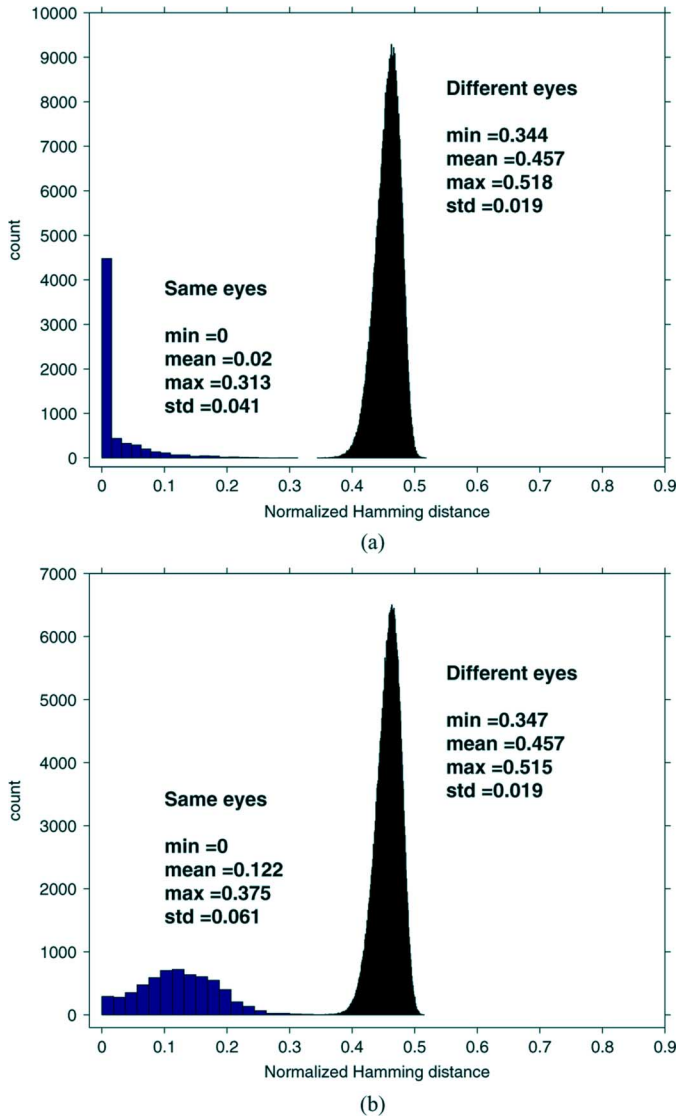

Fig. 11. Dissimilarity score distributions for same-eye and different-eye comparisons after seven rotations to allow for unknown tilts of the eye, head, or camera, plotted both in the case of ideal and nonideal imaging conditions. (a) Using a single type of camera. (b) When different types of cameras are used interoperably.

distance. We index the first sample from each eye (adding it to the UAE database), and use the remaining nine as queries. Since the enrollment setting is consistent, the mean intraeye Hamming distance is small: only 2% [see Fig. 11(a)]. Currently, the BGS algorithm only indexes the iris codes and ignores the masks. This has the effect of increasing the BERs. We find the BER of the beacons derived from the same eyes ranging from 4.56% up to 30.99%, with a mean of 14.94%. This error range can be well tolerated by BGS, as shown in the following.

Table III summarizes the experiment results for both ES and BGS. When $c = 3$, BGS reports an average delay of 30 ms, which is more than 300 times faster than ES. This is achieved as BGS checks only $41/N = 0.006\%$ of the database, while ES has to compare it with half of the records on average. The degradation of the false rejection rate—from 0.32% to 0.64%—is acceptable, given the great speedup factor. When $c = 4$, only two records are checked with a 99.04% success rate in finding the match. However, the gain in condensation is offset by the delayed early termination. As a result, the average search delay is 33 ms, slightly longer than that of $c = 3$. Note that in this evaluation, we took the data loading time into account. If we wish to hold all data in memory, we could achieve far more impressive performance by modifying BGS accordingly, but that will be less useful in practice.

TABLE III
COMPARISON BETWEEN ES AND BGS USING ONE TYPE OF CAMERA

Algorithm	FAR	FRR	Search size	Avg delay (ms)	max (ms)
ES	0	0.32%	317262	9,192	11,860
BGS (c=2)	0	0.48%	1087	133	6,488
BGS (c=3)	**0**	**0.64%**	**41**	**30**	**318**
BGS (c=4)	0	0.96%	2	33	177

In fact, BGS can be made to be much faster if we remove the seven-rotation requirement. In that case, the experiment shows that it takes merely 4 ms to find a match, provided that the query has less than 30% bit errors and no rotations. However, a slight

TABLE IV
COMPARISON BETWEEN ES AND BGS (UP TO THREE TRIES) MIXING TWO TYPES OF CAMERAS

Algorithm	FAR	FRR	Search size	Avg delay (ms)	max (ms)
ES	0	1.32%	320425	9,218	11,860
BGS (c=3)	0	0.55%	440	128	883

rotation of the query would cause the search to return no match and incur an upperbound delay of 28 ms. This may not be an issue if rotations are rare. However, in a noisy environment, as will be demonstrated, rotational shifts are fairly common.

The second experiment simulates a noisy environment, where there are two platforms using different cameras and measurement distances. On either platform, ten samples were collected from 61 eyes each. We index samples obtained on one platform, and use those acquired on the other platform as queries. The iris samples become fuzzier now: the mean intraeye Hamming distance increases to 12.2% [see Fig. 11(b)].

Under this noisy condition, ES reports a 1.32% false rejection rate. The BER of the beacons derived from the same eyes ranges from 12.81% up to 36.17%, with a mean of 25.58%. In addition, more than half (61%) of the queries are rotated due to head tilt, which makes noncyclic BGS unsuitable. More than 30% of the bit errors are more than what BGS can readily accommodate, but the rapid speed of BGS provides an option to query multiple scans of the same eye. The false rejection rate on the first attempt is 11.32%, and is reduced to 3.28%, 0.55% on the second and third attempts, respectively. While allowing three attempts, BGS reports an average delay of 128 ms, with the maximum of 883 ms (see Table IV). It is still significantly faster than ES; besides, it is more accurate because of the three attempts. (Though ES can also adopt the multiple-scan strategy—which reports 0.16% and 0.00% false rejection rates while allowing up to 2 and 3 tries, respectively—the maximum delay would increase linearly with the number of tries.)

Regardless of the enrollment condition, BGS uses much less memory than ES. The primary memory usage in BGS is an array that counts beacon collisions. If $c = 3$, only 2 bits are needed to record the collision count. To make it general, we use a byte, leading to $7 \times N$ bytes in total. In addition, BGS needs to store the retrieved IDs temporarily; the same space could be reused by subsequent retrievals (which is done automatically in Java through garbage collection). Hence, the total required memory in BGS is $(N)/(2^m)\times 4+7\times N \simeq 7\times N$ bytes. In comparison, a time-efficient implementation of ES requires the maximum use of the available memory; it uses $512 \times N$ bytes memory in our experiment (see Section III-B).

V. CONCLUSION

In this paper, we propose BGS for searching a large iriscode database efficiently. This algorithm works by indexing, adopting a "multiple colliding segments principle" and early termination strategy, so that the search range is reduced dramatically. It is evaluated using 632 500 real-world iris codes enrolled in the UAE border control since 2001. Two additional datasets are also included to study the variation of iris samples obtained from the same eyes under different conditions. The experiment shows that BGS is substantially faster than the current ES, with a negligible loss of precision. It requires much less memory and it does not depend on caching data in memory, hence obliterating the need for complex memory management. The preprocessing is simple and fast. It accommodates up to 30% bit errors in the query as well as up to seven cyclic rotations. The extra storage space is small and readily affordable–it supports dynamic maintenance, enabling easy indexing of new records. The rapid speed of BGS allows multiple acquisitions from the same eye, thus reducing the false rejection rate due to poor capture. Finally, we show that the empirical findings match the theoretical analysis. This makes BGS a useful technique for a wide range of fuzzy applications.

REFERENCES

[1] J. Daugman, "Probing the uniqueness and randomness of IrisCodes: Results from 200 billion iris pair comparisons," *Proc. IEEE*, vol. 94, no. 11, pp. 1927–1935, Nov. 2006.

[2] J. Daugman, "How iris recognition works," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 21–30, Jan. 2004.

[3] J. Daugman, "The importance of being random: Statistical principles of iris recognition," *Pattern Recognit.*, vol. 36, no. 2, pp. 279–291, 2003.

[4] F. Hao and C. W. Chan, "Online signature verification using a new extreme points warping technique," *Pattern Recognit. Lett.*, vol. 24, no. 16, pp. 2943–2951, 2003.

[5] J. D. R. Buchanan, R. P. Cowburn, A. V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. A. Allwood, and M. T. Bryan, "'Fingerprinting' documents and packaging," *Nature*, vol. 436, no. 28, p. 745, 2005.

[6] R. Pappu, B. Recht, J. Taylor, and J. Gershenfeld, "Physical one-way function," *Science*, vol. 297, no. 5589, pp. 2036–2030, 2002.

[7] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Similarity search: The metric space approach*. Berlin, Germany: Springer, 2006.

[8] B. Bustos, G. Navarro, and E. Chávez, "Pivot selection techniques for proximity searching in metric spaces," *Pattern Recognit. Lett.*, vol. 24, no. 14, pp. 2357–2366, 2003.

[9] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *Proc. 23th VLDB Conf.*, 1997, pp. 426–435.

[10] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory of Computing*, 1998, pp. 604–613.

[11] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th VLDB Conf.*, Edinburgh, Scotland, 1999, pp. 518–529.

[12] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," *Proc. 12th Symp. Computational Geometry*, pp. 253–262, 2004.

[13] A. Andoni and P. Indyk, E^2LSH User Manual Jun. 2005. [Online]. Available: http://web.mit.edu/andoni/www/LSH/manual.pdf.

[14] M. Miller, M. A. Rodriguez, and I. J. Cox, "Audio fingerprint: Nearest neighbor search in high dimensional binary spaces," in *Proc. IEEE Multimedia Signal Processing Workshop*, 2002, pp. 182–185.

[15] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. Int. Symp. Musical Information Retrieval*, 2002, pp. 144–148.

[16] The British Government website about the ID card scheme. [Online]. Available: http://www.identitycards.gov.uk/.

[17] The official site about the ration card scheme. [Online]. Available: http://hyderabad.ap.nic.in/rationcard.html.

[18] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "A real-time matching system for large fingerprint databases," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 799–813, Aug. 1996.

[19] M. Mattews, J. Cole, and J. D. Gradecki, *MySQL and Java Developer's Guide*. New York: Wiley, 2003.

[20] X. Qiu, Z. Sun, and T. Tan, "Texture analysis of iris images for ethnic classification," in *Proc. Int. Conf. Biometrics*, 2006, vol. 3832, pp. 411–418, Lecture Notes Comput. Sci.

[21] R. Mukherjee, "Indexing techniques for fingerprint and iris databases," M. S. dissertation, Lane Dept. Comput. Sci. Elect. Eng., West Virginia University, Morgantown, WV, 2007.

[22] L. Yu, D. Zhang, K. Wang, and W. Yang, "Coarse iris classification using box-counting to estimate fractal dimensions," *Pattern Recognit.*, vol. 38, no. 11, pp. 1791–1798, Nov. 2005.
[23] J. Thornton, M. Savvides, and B. V. K. Vijaya Kumar, "A Bayesian approach to deformed pattern matching of iris images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 4, pp. 596–606, Apr. 2007.
[24] P. J. Phillips, W. T. Scruggs, A. J. O'Toole, P. J. Flynn, K. W. Bowyer, C. L. Schott, and M. Sharpe, "FRVT 2006 and ICE 2006 large-scale results," National Inst. Standards Technol., NISTIR 7408, 2007 [Online]. Available: http://face.nist.gov.
[25] K. W. Bowyer, K. Hollingsworth, and P. J. Flynn, "Image understanding for iris biometrics: A survey," Univ. Notre Dame, Notre Dame, IN, Tech. Rep., 2007.

F. Hao, photograph and biography not available at the time of publication.

John Daugman, photograph and biography not available at the time of publication.

Piotr Zieliński, photograph and biography not available at the time of puiblication.