Appeared in Proc. of International Conference on Pattern Recognition (ICPR), (Tampa, USA), December 2008.

Indexing Iris Images*

Rajiv Mukherjee and Arun Ross
West Virginia University, Morgantown, WV 26506
rajiv.mukherjee@gmail.com, arun.ross@mail.wvu.edu

Abstract

Given a query iris image, the goal of indexing is to identify and retrieve a small subset *of candidate irides from the database in order to determine a possible match. This can significantly improve the response time of iris recognition systems operating in the identification mode. In this work, we analyze two different approaches to iris indexing. The first technique is based on the analysis of IrisCodes (post-encoding indexing); the second technique is based on the analysis of features extracted from the iris texture (pre-encoding indexing). Experiments on a subset of the publicly available CASIA-IrisV3 database compare the two approaches and illustrate the potential of the proposed indexing methods for large scale iris identification.*

1. Introduction

The goal of an iris biometric system is to extract the textural content of the iris and to authenticate an individual on the basis of the ensuing features [1]. During enrollment, the features of an iris image are extracted and stored in the database as a template. During authentication, the features extracted from a query image are compared against the templates in the database in order to perform identification or verification. An iris indexing (or filtering) technique will help reduce the number of candidate identities to be considered by an iris *identification* system when searching for a match in a large repository of irides [2, 3]. Current iris identification systems perform exhaustive matching based on the hamming distance between IrisCodes [1]. This paper introduces two iris indexing techniques. The first technique is based on an analysis of IrisCodes extracted from an individual's iris. The second technique examines the textural content of the image using the Signed Pixel Level Difference Histogram (SPLDH) of the raw pixel intensities.

2. Iris Indexing

The problem of automatic iris identification involves comparing a query iris image, q, with iris entries, $D = \{d_1, d_2, d_3, ...d_n\}$, in a database in order to determine the identity y of the iris. Each entry d_j, $j = 1, 2, \ldots n$, is assumed to be associated with an identity, y_j. The computational complexity of the identification process is primarily dictated by the number of entries, $\mid D \mid= n$, in the database. In order to reduce the number of matching operations, a filtering procedure can be invoked to identify a subset R of irides ($R \subset D$) such that $|R| = m << n$. An indexing scheme will assign an index value (scalar or vector) to each iris thereby allowing the query image to be compared against only those irides in the database that have comparable index values. Currently deployed commecial iris recognition systems can perform *rapid* one-to-many matching. Therefore, the need to explore iris indexing schemes may not be of immediate appeal. However, as database size increases and the need for processing non-ideal iris images emerges, the computational demand on iris identification systems is likely to increase. Thus, iris indexing schemes will play a pivotal role in organizing irides in diverse databases to facilitate the rapid retrieval of identities.

2.1. Approach to the problem

Two approaches are considered in this work. Since most commercial systems use IrisCodes to represent the iris texture, the first approach focuses on indexing IrisCodes. However, to accommodate iris images acquired using multiple acquisition devices and encoded using methods other than IrisCodes, a second indexing scheme that directly examines the textural content of the iris is

*The project was supported by the Center for Identification Technology Research (CITeR) and NSF CAREER Grant number IIS 0642554.

discussed. Thus we have two methods: indexing IrisCodes (post-encoding) or indexing iris texture (pre-encoding).

3. Indexing IrisCodes

The process of generating an IrisCode typically involves the following stages: (a) iris segmentation, where the iris is localized and isolated from the other structures in the vicinity such as sclera, pupil, eyelids and eyelashes; (b) geometric normalization, where the annular structure of the iris is mapped to the polar domain via an "unwrapping" procedure resulting in a rectangular entity; and (c) feature extraction, where this rectangular entity is projected onto a Gabor wavelet and the resulting phasor information quantized into an IrisCode. The IrisCode is a binary template whose spatial extent (and dimensionality) corresponds to the dimensions of the unwrapped iris structure. In order to organize the IrisCodes pertaining to multiple eyes (identities), we use a clustering scheme to create groups of IrisCodes. However, there are two factors that significantly impact this process: (a) the dimensionality of the raw IrisCodes can be very high (e.g., 2048); and (b) correlations may exist between certain dimensions. Therefore, the IrisCodes are first projected onto a lower dimension utilizing one of the following three methods prior to the application of a clustering scheme. In the first method, each row in the IrisCode is reduced to a single entity by merely averaging its entries; in the second method, the average of the entries in a column is used to represent that particular column; and in the third method, a linear transformation scheme (the Principal Component Analysis (PCA)) is applied to transform the IrisCode into a low-dimensional subspace. The k-means clustering scheme is then used to partition the dimension-reduced IrisCodes into multiple groups. The value of k dictates the number of clusters that can be generated from the IrisCodes. Thus, every IrisCode in the database is assigned to one of k clusters.

4. Indexing Iris Texture

The second iris indexing technique proposed in this work is motivated by the statistical analysis of pixel intensities and the positions of blocks (i.e., a local group of pixels) in the iris texture. This analysis is done using the Signed Pixel Level Difference Histogram (SPLDH). The rationale of this approach is to model the relationship between multiple sub-blocks *within a cropped region* in the unwrapped iris image and to assign an index value to each block indicating a list of other sub-blocks whose textural characteristics are similar to it. To facilitate this, the following sequence of steps is executed:

1. For a given segmented and normalized iris image I of size m by n, a cropped portion, I_c, of fixed size say $k \times l$ is extracted. I_c is divided into sub-blocks of size $r \times r$. Assuming k and l to be multiples of r, the number of sub-blocks in the cropped image will be $b_{tot} = (k_i * l_i)/r^2$. Each sub-block is assigned a unique position number in the integral interval $[1, b_{tot}]$.
2. The SPLDH is computed by examining the pixel intensities of two non-overlapping sub-blocks and computing the histogram of the signed differences in pixel intensities of the corresponding pixels in the two sub-blocks.
3. The entropy of the histogram is calculated in order to determine the 'similarity' of one sub-block with respect to the first one. For each sub-block, the absolute position of n other sub-blocks within the cropped iris image that are similar is used in order to generate an index vector for that sub-block.
4. For a given segmented, enhanced and cropped query iris image Q of size $k \times l$, the above procedure is repeated for all sub-blocks.

5. Iris Retrieval

The previous section examined the process of generating index values (or vectors) from iris images. In this section, we describe techniques to retrieve iris images from a database based on these indices. The performance of the retrieval system can then be characterized by two parameters: (a) hit rate, indicating the probability that at least one of the retrieved images corresponds to the correct identity; (b) penetration rate, indicating the fraction of user identities retrieved by the system. We first describe the database that was used in our work.

5.1. Database Used

The CASIA Iris Image Database V3.0 was used in the experimental evaluation of the iris indexing techniques proposed here. This database is divided into three parts: the CASIA-IrisV3-Interval dataset, the CASIA-IrisV3-Lamp dataset and the CASIA-IrisV3-Twins dataset. In this work, the CASIA-IrisV3-Interval dataset was used since it contains images captured in two sessions, separated by an interval of at least one month. This database contains images of resolution 320×280 pixels captured from 249 subjects. In most cases, both the left and right eye images of a subject are present in the database. However all users do not have the same number of image samples per eye. Only those users having at least six sample images per eye were considered in this work[1]. Since the left and right eyes are assumed to be independent, they can be treated as distinct 'users'. Thus, 143 'users' corresponding to the left eye and 139 'users' corresponding to the right eye, were identified. The training and test set each contain (randomly chosen) three samples per user.

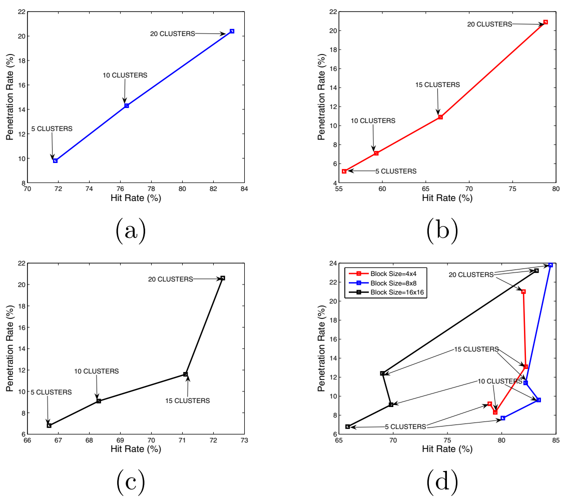

Figure 1. Indexing performance using IrisCodes: (a) PCA analysis on IrisCode. (b) Analysis of row-based features extracted from the IrisCode. (c) Analysis of column-based features extracted from the IrisCode. (d) PCA analysis of block-based features extracted from the IrisCode.

5.2. Retrieval Based on IrisCodes

Experiment 1: PCA is performed on the IrisCodes in the training set and dimensionality reduction is achieved according to the number of significant principal components. An unsupervised clustering (k-means) is applied on the reduced dimension set and the training dataset is split into k classes, with each class being represented by its centroid. Each class has a set of user ids associated with it. For a given query image Q, the IrisCode is generated and transformed via PCA to obtain the reduced-dimension index vector. Q is then assigned to a class according to its Euclidean distance from the centroid of the clusters.

Experiment 2: The index vector is constructed based on the mean of each row in the IrisCode. Unsupervised clustering (k-means) is performed on these index vectors, and the training dataset is split into k classes. For a given query image from the test set, the IrisCode is generated and the index feature vector is extracted. The query image is then assigned a class according to the Euclidian distance of the index vector from the cluster centroid. The user ids associated with the cluster represent the reduced candidate set for matching. The indexing result using the row based features of the IrisCode is reported in Figure 1. The indexing accuracy can be improved by extending the search to the next nearest neighboring class at the cost of increased penetration into the database.

Experiment 3: This is similar to Experiment 2 above, except that the index vector extracted is based on the mean of each column in the IrisCode present in the training database. Columns of the IrisCode correspond to radial direction in the iris. It is observed that row-based features perform better than column-based features.

Experiment 4: Each IrisCode in the training database is split into blocks of size m by m, where $m \in 4, 8, 16$. The first order statistic (mean) computed for each block serves as an index value. The index vector is extracted from all IrisCodes in the training database and an unsupervised clustering (k-means) is performed on the index vectors, and the training database is partitioned into k classes. The classification performance is reported by varying the values of k and m. The results for $m = 4, 8, 16$ are shown in Figure 1. Results indicate that $m = 4$ and $m = 8$ perform better than $m = 16$. This can be attributed to the fact that as the block size increases the local information is lost.

5.3. Retrieval Based on Iris Texture

The database management and the retrieval scheme is explained using a simple illustration. Assume that the cropped portion of each iris image can be divided into $b_{tot} = 16$ blocks (4 per row) of equal width and height. For each of these sub-blocks, p_i, $i = 1, 2, \ldots 16$, an index vector is first computed by determining the top n similar sub-blocks. For the sake of this illustration, assume $n = 2$. Then for each sub-block, p_i, an index vector $B(p_i)$ can be defined where $B(p_i) = (p_j, p_k)$. Assume, in this example, that $B(p_i) = (1, 2)$. Also, each sub-block has a 'tolerance zone' that includes a maximum of eight neighbors. For $b_{tot} = 16$, the 'tolerance zone' of sub-block 1 will include sub-block positions $2, 5, 6$. Similarly, the 'tolerance zone' of sub-block position 11 will include

[1]This was necessary in order to have sufficient training and test samples.

$6, 7, 8, 10, 12, 14, 15, 16$.

In Figure 2, the first level represented by squares (ranging from 1 to 16) denotes p_j. Based on the value of p_j, the first node, represented here by p_1, is further split into all possible sub-block positions according to the 'tolerance zone'. These block positions are represented by hexagons below level 1. Each node in level 1 is further split according to p_k into level 2 nodes. The level 2 node is represented by pentagons with index values (β_1, β_2), where β_1 represents the level 1 node and β_2 represents the sub-block positions in the tolerance zone of p_2. The nodes below level 2 are leaves. Each leaf stores the list of user IDs corresponding to a particular block index vector ('tolerance zone' included). In this illustration, since the splitting of the node in level 1 takes place according to $\beta_1 = 1$, the splitting of the node in level 2 results in $\beta_2 = \{2, 3, 5, 6, 7\}$. Thus, the retrieval scheme for index vector $(1, 2)$ will examine all nodes with indices θ_1, θ_2, where $\theta_1 = 1, 2, 5, 6$ and $\theta_2 = 1, 2, 3, 5, 6, 7$. Thus a total of 24 leaves will be examined out of a possible $b_{tot}^2 = 16^2 = 256$ possible leaves. An example of a traversal path for the block index vector $1, 2$ is indicated by the shaded blocks. After having visited all potential leaves according to the block index vector, a list of user IDs defining the set of potential matching candidates is retrieved. A voting scheme may be employed to determine user IDs that occur frequently in this list. (Since the user ID corresponding to a single user can occur in multiple leaves).

It is observed that on an average 2.3% of the users in the CAISA-V3 database occupy each leaf of the tree. In our experiments we set $b_{tot} = 96$ and $n = 2$. A hit rate of 84% was achieved at a penetration rate of 30% of the CASIA-V3 database. We anticipate that by considering more complex intra-block features and better 'tolerance' zones, the performance can be significantly increased.

6. Analysis of iris indexing techniques

The first technique is based on IrisCodes and performs unsupervised clustering on index vectors extracted from the IrisCode. In the PCA-based indexing of IrisCode the average penetration for a 80% hit rate is 17%, whereas the column-based scheme results in a hit rate of 80% for an average penetration rate of $\sim 21\%$. However, when indexing is based on the block-based statistics of the IrisCode (8x8 blocks), the average penetration for a 80% hit rate is only 8%. The second technique based on raw texture analysis results in a hit rate of 84% for an average penetration rate of 30%. Since a hit rate of 100% was not attained using these techniques, we are currently examining the sources of error.

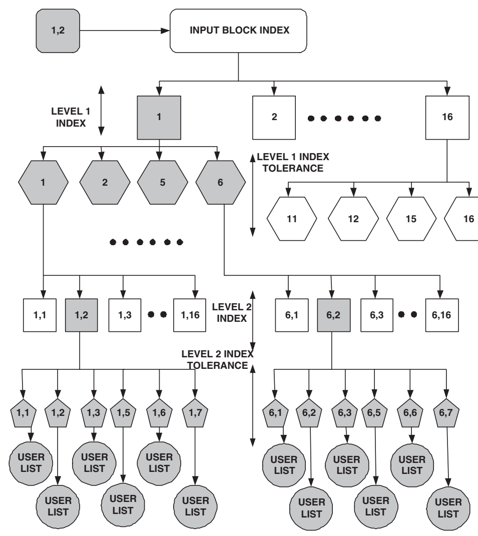

Figure 2. Block based indexing retrieval scheme for a given input block index.

7. Summary and Future Work

The purpose of this work was to highlight the possibility of indexing iris image databases. Two iris indexing approaches have been proposed based on the (a) analysis of IrisCodes and (b) analysis of iris texture. Since the indexing scheme based on SPLDH is block-based, there is an inherent sense of parallelism, where individual blocks can be indexed independently and the results consolidated thereafter. In addition, since the retrieval scheme is based on the *position* of blocks, the tree-like structure used for database management can be a fixed entity, thus simplifying the process of adding new users. However, there is plenty of scope for improvement: this work merely highlights the possibility of indexing large-scale iris databases.

References

[1] J. Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Trans on PAMI*, 15(11):1148–1160, November 1993.

[2] X. Qiu, Z. Sun, and T. Tan. Global texture analysis of iris images for ethnic classification. In *Int'l Conf on Biometrics*, pp. 411–418, 2006.

[3] L. Yu, D. Zhang, K. Wang, and W. Yang. Coarse iris classification using box-counting to estimate fractal dimensions. *Pat. Rec.*, 38(11):1791–1798, November 2005.