International Journal of Pattern Recognition and Artificial Intelligence

Iris Recognition Using AdaBoost and Levenshtein Distances

Joan Climent
Computer Eng. & Automatic Control Dept.
Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
juan.climent@upc.edu

Roberto A Hexsel
Depto. de Informática
Universidade Federal do Paraná (UFPR)
Curitiba, Brazil

This paper presents an efficient IrisCode classifier, built from phase features which uses AdaBoost for the selection of Gabor wavelets bandwidths. The final iris classifier consists of a weighted contribution of weak classifiers. As weak classifiers we use 3-split decision trees that identify a candidate based on the Levenshtein distance between phase vectors of the respective iris images. Our experiments show that the Levenshtein distance has better discrimination in comparing IrisCodes than the Hamming distance. Our process also differs from existing methods because the wavelengths of the Gabor filters used, and their final weights in the decision function, are chosen from the robust final classifier, instead of being fixed and/or limited by the programmer, thus yielding higher iris recognition rates. A pyramidal strategy for cascading filters with increasing complexity makes the system suitable for real-time operation. We have designed a processor array to accelerate the computation of the Levenshtein distance. The processing elements are simple basic cells, interconnected by relatively short paths, which makes it suitable for a VLSI implementation.

Keywords: iris recognition; AdaBoost; biometrics; Levenshtein distance; string matching; processor array.

1. Introduction

Biometric systems are becoming popular methods for personal identification. Each biometric technology has its set of advantages, considering their usability and security. The human iris, located between the pupil and the sclera, has a complex pattern determined by the chaotic morphogenetic processes during embryonic development. The iris pattern is unique to each person and to each eye, and is essentially stable during an entire lifespan. Furthermore, an iris image is typically captured using a non-contact imaging device, of great importance in practical applications. These reasons make iris recognition a robust technique for personal identification[33].

The first automatic iris recognition system was developed by Daugman[10]. He applied Gabor filters to the iris image for extracting phase features, known as the

IrisCode. While Daugman continued refining his algorithm[12], several researchers also worked on iris recognition. Wildes[34] uses a Laplacian pyramid to represent the iris texture, and classify the iris images by means of normalised correlation. Boles *et al.*[4] use an 1D wavelet transform at various resolution levels of concentric circles on the iris image. They characterise the texture of the iris with a zero-crossing representation. Ma *et al.*[19] employ a bank of spatial filters, with kernels that are suitable for iris recognition to represent the local texture features of the iris. Ma *et al.*[18] use as features a position sequence of local shape variation points. Sun *et al.*[27] use the histogram of local binary pattern for global iris texture representation and graph matching for structural classification; this method achieves high discriminability only in rich textured iris images. Lim *et al.*[17] decompose the iris image into four levels using 2D Haar wavelet transform, and use a modified competitive learning neural network (LVQ) as a classifier. Ali *et al.*[1] describe a support vector machine is used for classification. Bae *et al.*[3] project the iris signals onto a bank of basis vectors derived by independent component analysis and quantise the resulting projection coefficients as features. Correlation has also been used to recognise iris patterns. Vijaya Kumar *et al.*[16] describe a correlation filter for each class that employs 2D Fourier transforms of training images. Proença *et al.*[23] propose an iris classification method that divides the segmented and normalised iris image into six regions; an independent feature extraction and comparison is used for each region, and each of the dissimilarity values is combined through a classification rule.

Some authors have used Adaboost to obtain a strong classifer by means of boosting weaker ones. The results presented in these works show that the boosting algorithm can effectively improve the recognition accuracy. Wang *et al.*[32] use wavelet probabilistic neural networks as weak classifiers. Their published results for EER, at 3.34%, are distant from those obtained by using classical Gabor wavelets. Tian *et al.*[28] propose a zero-crossing detection method for iris feature extraction, and boost these instead of the Gabor filters. They show that the correct classification rate is not as accurate as the one obtained with Gabor filters, but the computation takes less time. They report a 99% recognition rate using the CASIA database.

Chen *et al.*[7] characterize the iris patterns by using an edge-type descriptor. Each edge-type flag is used to design a weak classifier. The edge-type descriptors, like the Gabor filters, are computed in a filter pyramid to obtain coarser level descriptions. The authors determine the edge-type flags with a derivative of Gaussian filter and the Laplacian of Gaussian filter. A set of DoG/LoG filters is then selected using the Adaboost algorithm.

Anoter work that makes use of boosting is that by He *et al.*[15]. Instead of Gabor phasors, ordinal measures are used for iris representation. There are however too many parameters that need tuning when using ordinal measures, and to construct and optimal classifier is a difficult problem. The authors suggest the use of similarity oriented boosting instead of AdaBoost. Ordinal measures are difficult to boost,

thus oriented boosting must be driven by a similarity rule because of the large amount of features. It should be noted that all these algorithms work with grey-level images, and colour information is not used since the most important information for recognition is the texture variation of the iris, which is the same in both grey and colour images.

In general, the iris recognition process consists of five stages: (i) image acquisition, (ii) iris localisation, (iii) iris normalisation, (iv) IrisCode extraction, and (v) iris pattern identification. In this paper we deal with the fifth stage, iris pattern identification. Our approach to iris pattern identification is to first, use Levenshtein distance to classify a given IrisCode, and then employ AdaBoost to decide similarity[8]. We designed an array processor to accelerate the computation of the Levenshtein distance between an input IrisCode and a set of previously recorded IrisCodes.

Existing methods for iris identification are based on the comparison of IrisCodes. There are several approaches to measure the angular distances between IrisCode vectors, but the most widely used is the normalised Hamming distance. We use Levenshtein distance to measure the differences between IrisCodes. We show that this is a more accurate metric for deciding if two IrisCodes belong to the same person. Also, whatever the metric used, some parameters must be fixed by the programmer, such as the minimum distance needed to consider that two IrisCodes belong to the same person, the number of wavelengths being used, and their values. We present an iris identification process based on AdaBoost classification. The boosting process selects the most significant wavelengths for the identification, and also determines the threshold distances between phase vectors to decide if they belong to the same iris. The boosting is implemented as a cascade of filters as that greatly speeds up the decision process.

Section 2 briefly presents the iris recognition technique using the five steps mentioned above. Section 3 explains our choice of the Levenshtein distance over the Hamming distance. Section 4 describes the AdaBoost algorithm we implemented. In Section 5 we show that the proposed approach improves accuracy in the iris identification stage. In Section 6 we present an array processor designed to accelerate the computation of the Levenshtein distance. Our conclusions are stated in Section 7.

2. Iris image encoding

In general, iris recognition systems are composed of five stages: acquisition, localisation, normalisation, encoding, and identification. Figure 1 shows the results obtained after the four initial stages.

Prior to obtaining the IrisCode, the pupil must be located and the iris segmented. We employ a standard technique to segment the iris[34]. The iris can be located at the region between two concentric circles, one for the iris-sclera boundary and another for the iris-pupil boundary, as shown in Figure 2.b. The pupil is

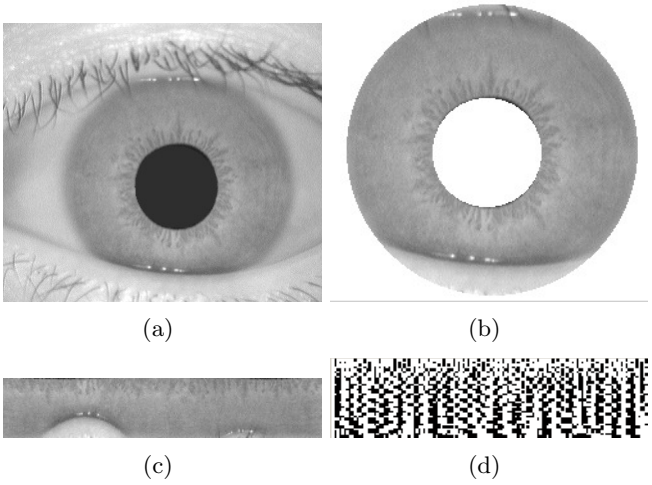


Figure 1. Four initial stages of iris recognition: (a) acquired eye image; (b) segmented iris; (c) normalised iris; (d) IrisCode.

detected with the integro-differential operator shown in Equation 1, as proposed by Daugman[10].

$$\max_{r,x_0,y_0} \left| G_\sigma(r) \cdot \frac{\partial}{\partial r} \oint_{r,x_0,y_0} \frac{I(x,y)}{2\pi r} ds \right| \tag{1}$$

The iris region is then transformed into a rectangle. This Cartesian to polar transform, known as normalisation, is based on the Daugman's rubber sheet model[10]. As shown in Figure 2.c, each point of the iris image is mapped to a pair of polar coordinates (r, θ), where radius $r \in [0, 1]$ and angle $\theta \in [0, 2\pi]$. Regions with high occlusions are not considered, and the amount of occlusion free areas can be used as a quality measure[35].

Iris encoding is implemented using Gabor filters, which are a combination of Gaussian and sinusoidal functions. Since they are bandpass filters, the effects of high-frequency noise and low-frequency illumination non-uniformity can be minimised. Equation 2 describes a Gabor filter, where λ is the wavelength, σ the Standard Deviation, γ the aspect ratio, and θ the filter orientation.

$$\begin{aligned} \Psi(x,y) &= e^{-\frac{x'^2+(\gamma y')^2}{2\sigma^2}} \cdot \cos\frac{2\pi x'}{\lambda} \\ &\quad + i \cdot e^{-\frac{x'^2+(\gamma y')^2}{2\sigma^2}} \cdot \sin\frac{2\pi x'}{\lambda} \\ x' &= x\cos\theta + y\sin\theta \\ y' &= -x\sin\theta + x\cos\theta \end{aligned} \tag{2}$$

Because phase information is robust to illumination changes, only this is used from Equation 2. The real component of Equation 2 corresponds to the symmetric part

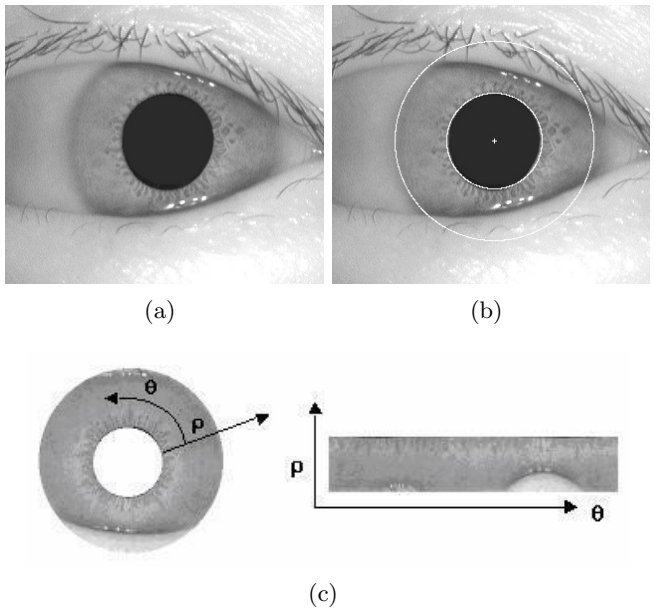


Figure 2. Iris localisation and normalisation: (a) original image; (b) iris localisation; (c) iris normalisation.

of the filter, while the imaginary corresponds to the asymmetric. Figure 3 shows an example of the iris encoding process.

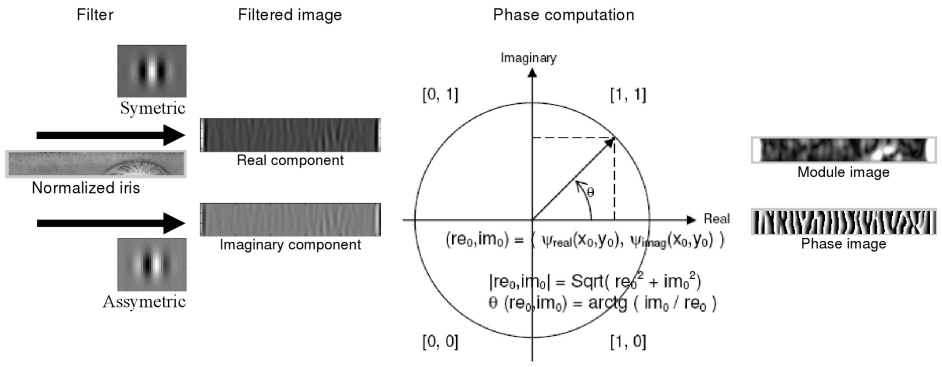


Figure 3. Iris encoding process.

Once the iris image has been encoded, it must be compared to others to verify identity. The usual metric to compare angular distances between IrisCodes is the normalised Hamming distance, with the phase angle encoded into 2 or 3 bits. If the

distance between two IrisCodes is smaller than a fixed threshold, the two images are taken as belonging to the same iris. We investigate this comparison next.

3. The Edit Distance

A novel approach for comparing IrisCodes, which uses *Levenshtein distance*, is presented in this paper. The Levenshtein distance[5], also called *edit distance*, is employed for measuring the difference between two strings. The distance is given as the minimum number of operations needed to transform one string into the other, where an 'operation' is an insertion, deletion, or substitution of a single character. This metric is useful in a wide range of applications, and there is a large body of work concerning string comparison using Levenshtein distance in recent literature – see, for example, Ref. 25. The usual way to compute the Levenshtein distance is with an $(m+1) \times (n+1)$ cost matrix $\mathcal{L}$, where m and n are the lengths of the two strings.

Let $\mathcal{L}$ be the cost matrix for strings S_A and S_B. The value $\mathcal{L}(i,j)$ represents the distance between substrings $S_A[1,i]$ and $S_B[1,j]$. The cost matrix values are computed using a dynamic programming algorithm, where the cost $\mathcal{L}(i,j)$ is determined from previously computed costs, according to Equation 3, where $C_i(S_B(j))$, $C_d(S_A(i))$ are the insertion and deletion costs, and $C_S(S_A(i), S_B(j))$ is the cost of substitution of the i-th character of string A by the j-th character of string B. The final edit distance between strings A and B is determined by the last value in the cost matrix: $d(A,B) = \mathcal{L}(m,n)$.

$$
\begin{aligned}
\mathcal{L}(0,0) &= 0 \\
\mathcal{L}(0,j) &= \mathcal{L}(0,j-1) + C_i(S_B(j)), \quad 1 \le j \le n \\
\mathcal{L}(i,0) &= \mathcal{L}(i-1,0) + C_d(S_A(i)), \quad 1 \le i \le m \\
\mathcal{L}(i,j) &= \min \begin{cases} \text{for all } \; 1 \le i \le m\,, 1 \le j \le n \\ \mathcal{L}(i-1,j) + C_d(S_A(i)) \\ \mathcal{L}(i,j-1) + C_i(S_B(j)) \\ \mathcal{L}(i-1,j-1) + C_S(S_A(i), S_B(j)) \end{cases}
\end{aligned} \tag{3}
$$

We set the insertion and deletion costs both to 1, and the substitution cost to half the Hamming distance between characters. The characters are the 2-bit coded phasors, and the strings are IrisCode segments corresponding to a given wavelength.

For a given pair of strings, the Hamming distance is equivalent to the Levenshtein distance if the only operation considered is substitution, and insertions and deletions are ignored. Thus, the Levenshtein distance can be a more accurate metric than Hamming distance for comparing IrisCode segments. In fact, the Hamming distance is an upper bound of the Levenshtein distance, as proved by Navarro[22]. The insertion and deletion operations account for the elastic rotations found in iris patterns. This means that the pixel shifts are not homogeneous around the iris pattern, because of the eccentricity of the pupil, the segmentation results, and the acquisition process. With the use of the deletion/insertion we can handle such

rotations, which is not possible with the standard Hamming distance algorithm. In the standard approach, IrisCodes are compared multiple times, shifting the patterns, in order to compensate the effect of iris rotation. Just shifting the patterns is equivalent to pure geometric rotations of the input image, however, the rotations found in iris patterns are not geometric but elastic rotations.

We compared the performance of the two methods, namely Levenshtein distance and Hamming distance, by measuring the intra-class, and inter-class, distances. IrisCodes from CASIA database from 100 different eyes, taken in 7 different illuminations, were used. Over 10,000 comparisons between IrisCodes were made, of the same iris under different conditions, using Levenshtein and Hamming distances. The intra-class means and standard deviations were thus obtained. Another set of 10,000 comparisons were made between IrisCodes from different persons to obtain the inter-class means and standard deviations.

With this experiment we attempted to determine which metric is more suitable for iris identification. Hence, the distances were computed using the raw IrisCode, prior to the boosting stage. For two-choice decision tasks, such as in biometric decision making, the decidability index δ is one measure of how well separated the two distributions are, since recognition errors would be caused if these overlap[11]. The performance of any biometric technology can be calibrated by its δ score.

If the two means are μ_1 and μ_2, and their respective standard deviations are σ_1 and σ_2, then δ is defined by Equation 4.

$$\delta = \frac{\mu_1 - \mu_2}{\sqrt{\frac{(\sigma_1^2+\sigma_2^2)}{2}}} \tag{4}$$

This measure of decidability is independent of how conservative the acceptance threshold is. Rather, by measuring separation, δ reflects the degree to which any improvement in the false-matching error-rate is attained at the expense of a worsening in the failure-to-match error-rate. The decidabilities measured on our data sets are $\delta = 3.64$ using Levenshtein distance, and $\delta = 1.61$ for the Hamming distance. Thus, Levenshtein distance is a more discriminative measure to compare IrisCodes.

A direct application of the dynamic algorithm for Levenshtein distance computation has a computational cost of $O(mn)$, where m and n are the length of the strings. Ukkonen[29] presents an algorithm that can check whether the edit distance is below a given threshold $\mathcal{K}_\lambda$ in $O(\mathcal{K}_\lambda^2)$. This is called a "diagonal transition algorithm", because the diagonals of the dynamic programming matrix (from the upper-left to the lower-right cells) are monotonically increasing. The algorithm computes, in constant time, the positions where the values along the diagonals are incremented. Only $\mathcal{K}_\lambda^2$ such positions need be computed to reach the lower right decisive cell. The classifier described in Section 4 does not use the distance value, but it makes a decision based on the fact that distance is smaller than a given threshold $\mathcal{K}_\lambda$. Thus, the Ukkonen approach is an efficient solution for our application. Even though the computation of the Hamming distance has a lower computational cost, the computation of Levenshtein distance with Ukkonen cost reduction, and a cascading

strategy shown in next section, can give a solution in a few milliseconds using a database such as CASIA. Results for computation times are given in Section 5. For a really large database, perhaps a specific hardware architecture would have to be used, if so needed. There are proposals for such architectures in the literature, as that by Climent *et al.*[9], and their respective low cost hardware implementation[2], as a dedicated processor implemented in a FPGA.

The edit distances are computed for IrisCode segments obtained by using different wavelength filters. In a classical approach, distances computed at different scales have the same weight in the classification, ignoring the fact that some distances might be more significative than others. This problem can be effectively solved with a boosting algorithm. We propose AdaBoost to obtain a classifier based on a weighted function of simple decisions taken from the edit distances computed by comparing the segments. Filter wavelengths, effective for discriminating different iris patterns, have a higher weight in the robust classifier, thus improving the recognition rate. Also, these weights and decision thresholds are determined automatically by the algorithm. The algorithm is described in next section.

4. Iris classification using AdaBoost

Boosting is a meta-algorithm for automatic learning that builds a robust classifier by a combination of a set of weak classifiers. A classifier is considered weak if it has a correct classification ratio slightly better than chance. Consider a set of weak classifiers $h_t(x)$, then the strong classifier $sign(f(x))$ is defined by Equation 5.

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{5}$$

The AdaBoost meta-algorithm obtains the strong classifier iteratively. In each iteration, a weak classifier is added in, weighted by its predictive capacity α_t. Each training pattern is given a weight which determines its probability of being selected to the training set for a weak classifier. If a training pattern is correctly classified, then the probability of being used again in a subsequent weak classifier is reduced. Conversely, if the pattern is not well classified, then the probability of being used again is raised. The AdaBoost algorithm[14] is detailed next.

(1) Given
   (a) a training set: $\{ (x_1, y_1), \cdots, (x_m, y_m) \}$;
   where x_i are IrisCode segments, and y_i the label of their corresponding class;
   (b) a set of weak classifiers $h_j( )$;
(2) Initialise a distribution $D_1(i) = 1/m$ for $i = 1 \cdots m$
(3) For $t \leftarrow 1 \cdots T$:
   (a) find the weak classifier with minimum error: $h_t = \arg\min_{h_j} \epsilon_j$,
   where $\epsilon_j = \sum_{i=1 \ldots m} D_t(i)$ and $y_i \neq h_j(x_i)$;
   (b) compute the coefficient α_t: $\alpha_t = 0.5 \ln(1 - \epsilon_t)/\epsilon_t$;

(c) update, for normalisation factor Z_t,

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t} \text{ if } h_t(x_i) = y_i \\ e^{\alpha_t} \quad \text{otherwise} \end{cases}$$

(4) Return the strong classifier

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{6}$$

As weak classifiers $h_j(\,)$ we use size-parameterisable decision trees. For a threshold $\mathcal{K}_\lambda$, that minimises the error of the weak classifier for a given wavelength λ, every split is a simple decision:
if $d_\lambda(i,j) < \mathcal{K}_\lambda$ *then* $h_\lambda(i,j) = +1$ *else* $h_\lambda(i,j) = -1$,
where $d_\lambda(i,j)$ is the edit distance between two IrisCode lines i, j, computed for a wavelength λ.

In the training stage, the distances are pre-computed for all possible wavelengths. N distance matrices d_λ are built corresponding to N different wavelengths, where $d_\lambda(i,j)$ is the Levenshtein distance, obtained with the algorithm defined by Equation 3. The wavelength range is limited at 24. Labels are assigned to each distance value:
if (i and j are samples from the same iris) then $y_\lambda(i,j) = +1$ *else* $y_\lambda(i,j) = -1$.

Thus, a robust classifier $H(x)$ is produced by the linear combination of the results obtained from the decision trees, as defined in Equation 6, where x are the values of the N matrices d_λ of Levenshtein distances between IrisCodes lines, α_t can be understood as the weight of the classifier h_t in the global decision, and $\sum_{t=1}^{T} \alpha_t h_t(x)$ is a measure of the confidence for the iris identification.

Figure 4 shows a plot of error × number of iterations of the AdaBoost algorithm. The error tends to stabilise beyond 100 iterations, so we chose 150 iterations to train the classifier. The decision-tree size S – the number of splits – must be fixed, and S was fixed at 3, which is a good trade-off between accuracy and processing time, as shown in Section 5.

In the recognition stage, we applied a technique of cascading the classifiers to reduce the computation time, as widely used in face detection applications[31]. The key idea is that smaller, therefore faster, boosted classifiers can be constructed, which reject many of the false candidates, while accepting all positive ones. The simpler classifiers are used to reject the majority of false candidates, before the complex classifiers are called upon to achieve a low rate of false-acceptation. Figure 5 contains a diagram of the identification process, known as cascade[24]. A positive identification from the first classifier triggers the evaluation of a second classifier, which in case of positive identification triggers a third one, and so on. A negative identification on any level causes the immediate rejection of the candidate.

The complete system consists of an 8-level cascade of classifiers. The first classifier is a single decision tree, while the last classifier consists of the weighted combination of 150 decision trees. Each matching attempt is processed at the various

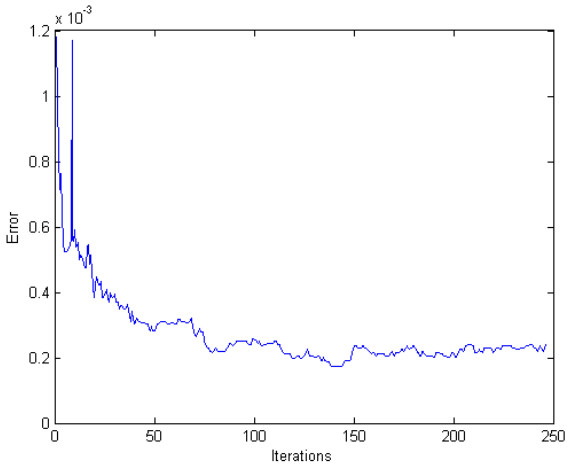


Figure 4. Error evolution vs. AdaBoost iterations.

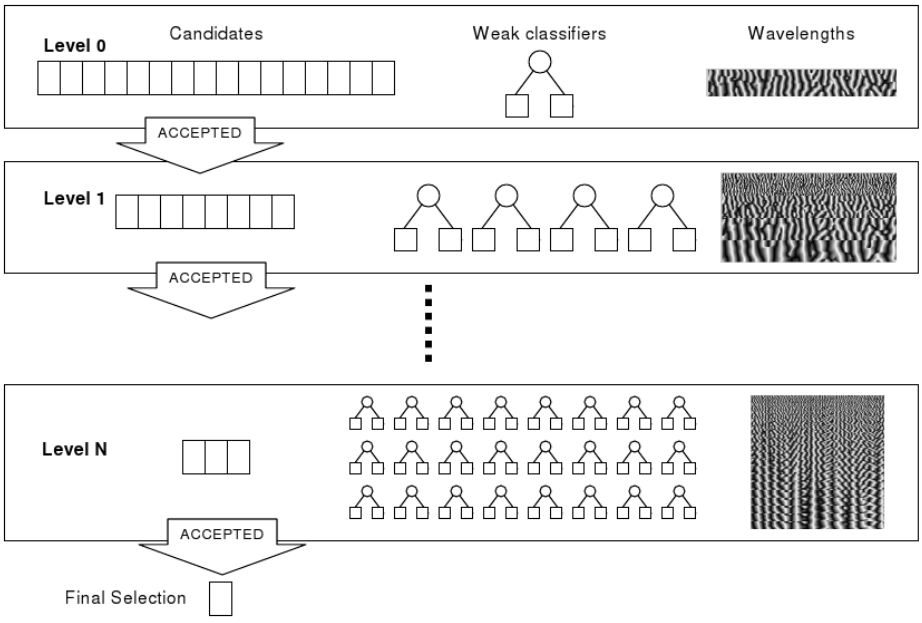


Figure 5. Cascade of classifiers.

levels, and if any classifier rejects the attempt, the processing terminates with a rejection. This cascading strategy allows a majority of non-matching attempts to be

quickly rejected – with fewer comparisons, while spending more computation time on potential true-matchings. The process is thus considerably accelerated, without losing accuracy in identification, and the choice of 8-levels is a tradeoff between speed and accuracy.

Using this cascading strategy there is no need to compute all distances for all possible wavelengths, since they are computed at the corresponding level if needed. In fact, not even single distances have to be totally computed because the weak classifiers need only to ascertain if a given distance is below the desired threshold $\mathcal{K}_\lambda$. Once the Levenshtein distance being computed is above the threshold, the candidate is rejected and there is no need to complete that computation.

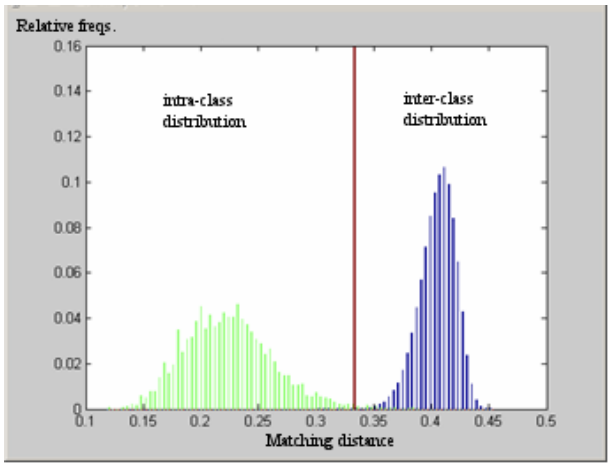
5. Results

The iris recognition system was tested with images from CASIA[6] and UPOL[30,13] databases: 756 monochrome images from 108 eyes from CASIA database; and 384 colour images from 64 persons from UPOL database. As described in Section 2, iris images were normalised to 256x32 pixels. Each iris image is coded using a 256x32 phasor array, and each phasor was encoded in Gray code with 2 bits. Wavelengths considered range from 2 to 24 pixels.

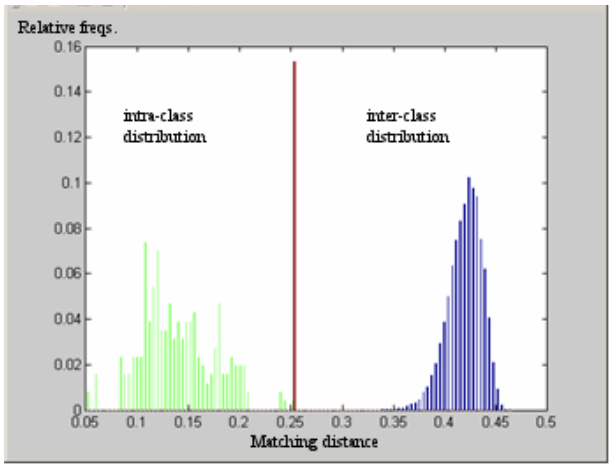
Error rates of 0% were easily achieved with images from the UPOL database. Images from this database are free of eyelids, eyelashes and other interferences. Figure 6 shows that the distance between the intra-class and the inter-class distributions is very large for the UPOL database, indicating a complete decidability. Hence, when using images from this database results are not realistic.

The CASIA database is the most commonly used iris image database for evaluation purposes, and several papers report their experimental results while making use of this database. Thus, we use CASIA for comparing our results to other published methods. The number of iris images from CASIA database is sufficiently large for an adequate performance comparison. Three images from each person were randomly chosen as a training set, whereas the remainder are the control set, without overlap. This yields 432 images for a single test. The test results should be independent on the training images randomly chosen. Thus, a new test is done with 3 new training images, and tested again with the other 4, which were not previously used for learning. This process is repeated 1,000 times. Error ratios are computed using the set of results obtained from the whole process.

To evaluate the performance of the identification system, we use the receiving operating characteristic (ROC), the equal error rate (EER) and the false-rejection rate (FRR) for false-acceptation rate (FAR) equal to zero. The ROC curve displays the FRR as a function of the FAR, that is, "the probability that an authorised person is falsely rejected" against "the probability that an unauthorised person is falsely accepted"[20]. The EER indicates the identification error rate for which the FAR and the FRR are equal. FRR for FAR=0 is a useful metric for access/entry validation since it guarantees that no unauthorised person can gain access to the



(a)



(b)

Figure 6. Distribution of intra-class and inter-class distances: (a) CASIA database; (b) UPOL database.

protected environment. Table 1 shows the error rates obtained using four different decision tree sizes. A decision tree with 4 splits gives a slightly lower ERR than one with 3 splits, but the latter is the best trade-off between speed and correct identification rate, and for this reason the number of splits of the decision trees has been fixed at 3. Our identification system achieves an $FRR_{FAR=0}$=1.03%.

Figure 7 shows the FRR×FAR ROC curve using 3-split trees as weak classifiers and the CASIA database. The results obtained with our system were compared to published methods. Table 2 shows the results reported in the literature that use

Table 1. Results as a function of tree splits

tree splits	EER [%]	$FRR_{FAR=0}$ [%]
1	0,045	3,51
2	0,017	2,18
3	0,004	1,03
4	0,003	1,03

CASIA database for test images. Results from the combination of two methods are reported by Sun *et al.*[27]. The Mizayawa method[21] has de best performance, followed by He's method[15]. We also include some intermediate results using Hamming distance instead of Levenshtein distance. This helps to evaluate the effect of the distance in the overall EER. We performed tests using the IrisCode lines, and also considering all possible +-3 pixels shifts corresponding to iris rotations.

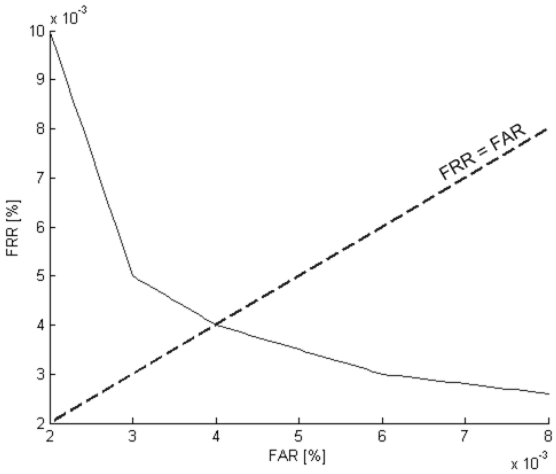


Figure 7. ROC curve with the EER line superimposed.

In the training process, IrisCodes are computed from images, and the wavelengths and their respective thresholds for the classifier are automatically determined by the boosting process. In the recognition process, an iris image is encoded and compared to all other pre-computed IrisCodes in the database. The input IrisCode is computed in 405 μs on a 2.13GHz Intel Core 2 Duo 6400. The Levenshtein distance between two IrisCode rows is computed in 240 μs. The time needed to decide if two iris images belong to the same eye, in a worst case situation, is

Table 2. Comparison with existing methods

Method	EER [%]
Boles *et al.*[4]	8.13
Wildes[34]	1.76
Proença *et al.*[23]	1.01
Sun *et al.*[27]	0.86
Ma *et al.*[19]	0.51
Daugman + Tan[27]	0.49
Daugman + Sun[27]	0.37
Tan + Sun (27)	0.32
ours using Hamming distance	0.08
Daugman[10]	0.08
Ma *et al.*[18]	0.07
ours using HD ±3 pixel rotations	0.05
Chen *et al.*[7]	0.023
He *et al.*[15]	0.01
Miyazawa *et al.*[21]	0.0099
ours using Levenshtein distance	0.004

72.5 ms, which is just an upper bound. Actual times are (much) shorter because the complete Levenshtein distance array is seldom computed: most candidates are refused once the partial distance is above a given threshold. Furthermore, with the cascading strategy, not all the 150 weak classifiers must be used. In fact, 42% of candidates are refused in the first cascading stage, 23% in the second, 19% in the third, 13% in the forth, and less than 3% go beyond the fifth stage. To determine the average computing time to decide if two iris images belong to the same eye, we selected 200 random images and compared them with the whole database. The average time for the verification is 1.6 ms.

6. Processor Architecture

We designed an array processor to speed up the computation of the Levenshtein distances between an input IrisCode line and k different IrisCode lines previously recorded.

A direct implementation of the dynamic programming algorithm in Equation 3 would require a large amount of circuitry, because the distances in the Levenshtein matrix can assume large values when comparing long strings. Therefore, a processor element would have to add and compare large data values, and the widths of the buses between adjacent processing elements would be rather wide, and the routing of the circuit connections would also be prohibitive.

If the elements of the distance matrix are represented as Sastry *et al.*[26] suggest, it is possible to bound the information length that is transmitted from one processing element to another, and avoid the use of large values. This encoding scheme consists of computing, for every element in the Levenshtein matrix, two incremental costs, instead of one absolute cost. One incremental cost is the difference between the matrix element and its top neighbour, namely the *incremental vertical cost* (C_v). The other incremental cost is the difference between the matrix element and its left neighbour, the *incremental horizontal cost* (C_h).

Using this representation, the algorithm to compute the Levenshtein distances constructs two different matrices, one for the incremental vertical costs, and another for the incremental horizontal costs. The values of the elements of these matrices for a given row i and column j are determined by Equation 7.

$$\begin{aligned} \Lambda &= C_h(i-1,j) + DelCost \\ \Gamma &= C_v(i,j-1) + InsCost \\ \Theta &= SubCost(i,j) \\ C_v(i,j) &= \min(\Lambda,\Gamma,\Theta) - C_h(i-1,j) \\ C_h(i,j) &= \min(\Lambda,\Gamma,\Theta) - C_v(i-1,j) \end{aligned} \tag{7}$$

In Equation 7, for all $i,j \leq n$, $C_h(i,0) = DelCost$ and $C_v(0,j) = InsCost$. The substitution cost $SubCost(i,j)$ of phasor i by phasor j is the exclusive OR between those characters.

The distance between two IrisCode lines, α and β, is determined by the two formulae in Equation 8.

$$\begin{aligned} d(\alpha,\beta) &= n \cdot InsCost + \sum_{i=1}^{n} C_v(i,n) \\ &= n \cdot DelCost + \sum_{j=1}^{n} C_h(n,j) \end{aligned} \tag{8}$$

Figure 8 shows the dependencies between matrix elements using the incremental approach: each element depends only on elements that are located above and to the left. Thus, all elements along the 45^o diagonal can be computed simultaneously.

Figure 9 shows a block diagram of the circuit that computes the incremental costs of a single element in a Levenshtein array. As insertion and deletion costs are considered to be the same and defined to be 1, the addition circuit may be implemented as an increment operation. Incremental costs C_v and C_h are input to the system and the substitution cost is computed by the XOR operation. The processing element computes new incremental costs according to Equation 7. The incremental vertical cost is transmitted to the adjacent processor to the right, while the incremental horizontal cost is transmitted down.

A block diagram of the array processor is shown in Figure 10. The array computes the distance between an input IrisCode line with k others previously registered

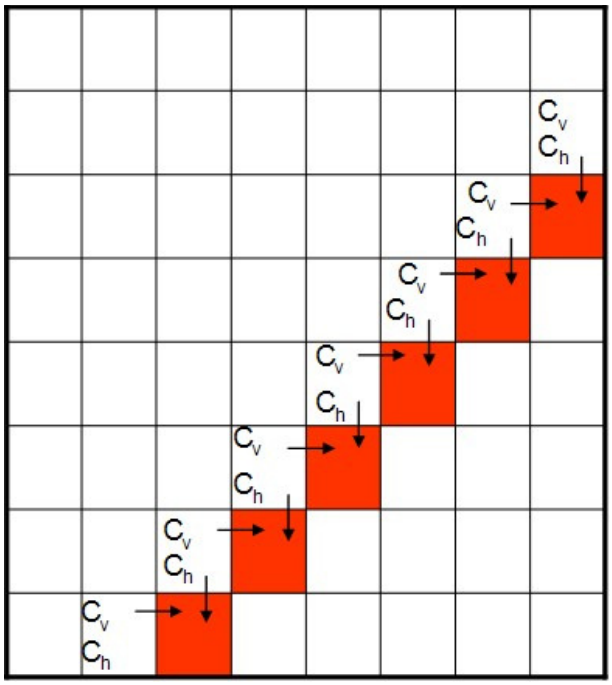


Figure 8. Dependencies between matrix elements.

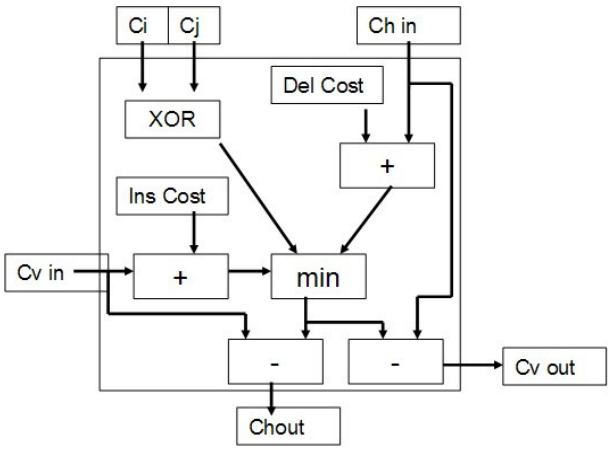


Figure 9. Processing element.

IrisCodes. Each processing element performs the computations along its respective column in the edit distance matrix. If n the IrisCode length, then n processing

elements are needed at every cycle to compute the incremental costs.

An accumulator is used to hold the edit distance. The output of the processing element at the last column is added to the accumulator. The width of the accumulator depends on the maximum value that the edit matrix can take, and is therefore dependent on the edit costs and the length of IrisCodes.

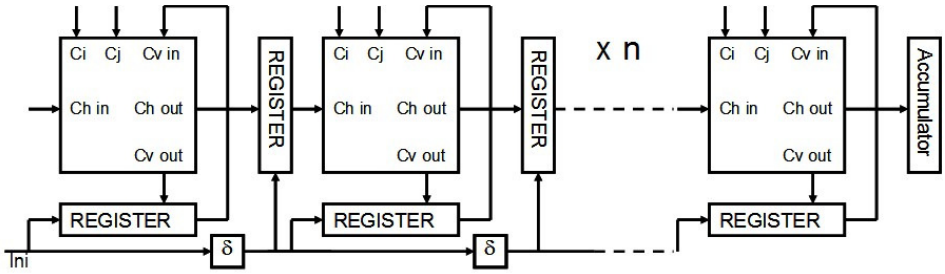


Figure 10. Array of processing elements.

On every clock cycle, substitution costs corresponding to a new 45^o diagonal of the cost matrix are computed by the corresponding processing elements. At the same time, incremental costs computed by each processing element are latched into the corresponding horizontal and vertical registers. The incremental vertical cost coming from the last processing element is also added in the same cycle. The accumulator is initialised with the value n ($n \cdot InsCost$). After the first $2n-1$ cycles, the accumulator contains the distance between the input IrisCode line and the first registered IrisCode. Then, every n cycles the distance to a new registered IrisCode line is computed.

This design uses just one single phase signal to control all data flow. Since k input registered sequences are being compared, the registers are initialised by the signal (Ini) each time a new sequence is input to the system. The delay elements δ are also controlled by the same clock signal.

Array implementation and preliminary results A prototype of the array was tested on a Xilinx Spartan-3E kit. The FPGA accommodates 256 processing elements, an array large enough for testing the IrisCode line matching. The timing of the circuit was analysed to find critical paths and the minimum clock cycle period. The critical paths between processing elements yield a worst case delay of 8 ns. Restrictions in the design tools at our disposal allow a maximum clock frequency of 50 Mhz. Only a single phase clocking signal is needed to control the array processor, and that minimises the use of interconnect lines, yielding short cycle times.

The implementation is in a early stage. More than one IrisCode line comparison can be performed in parallel if the array were implemented on a higher density device. Of course, clock speed can be increased with a faster FPGA and more effort

on fitting the design to the device. This is object of work that is being currently undertaken.

7. Conclusion

We present a novel iris recognition technique that minimises false identification rates. The two main contributions of our work are (i) the introduction of the AdaBoost algorithm in the iris pattern classification stage, and (ii) the use of Levenshtein distance instead of Hamming distance. This approach leads to an effective identification system and the error ratios obtained are considerably lower than those obtained using existing techniques.

The computation of the Levenshtein distance has a higher cost than the computation of of the Hamming distance. However, this cost is significantly reduced by the use of the diagonal transition algorithm. Efficiency is further increased by the cascade of classifiers, structured as a pyramid. The resulting classifier is computationally efficient since only a small number of decision trees need to be evaluated during run time.

We designed an array processor to accelerate the computation of the Levenshtein distance between an input IrisCode and a set of previously recorded codes. By changing the encoding of the distance between words in a code, the processing elements are small and simple basic cells, interconnected by relatively short paths.

Acknowledgements Portions of the research in this paper use the CASIA iris image database collected by the Institute of Automation, Chinese Academy of Sciences, and the UPOL database from the Palacky University in Olomuc. This research was partially supported by Consolider Ingenio 2010, project (CSD2007-00018) and CICYT project DPI2010-17112. One of the authors (RAH) was supported by grant BEX 1656-09-0, from CAPES, Ministry of Education, Brazil.

Bibliography

1. H S Ali and M J E Wahyudi. Iris recognition system by using support vector machines. In *Int Conf on Computer and Comunications Engineering (ICCCE'08)*, pages 516–521, 2008.
2. J Aranda, J Climent, A Grau, and A Sanfeliu. Low cost architecture for structure measure distance computation. In *Proc Int Conf on Pattern Recognition*, pages 1592–1594, 1998.
3. K Bae, S Noh, and J Kim. Iris feature extraction using independent component analysis. In *Proc 4th Int Conf on Audio- and Video-Based Biometric Person Authentication*, LNCS 2688, pages 838–844, 2003.
4. W W Boles and B Boashash. A human identification technique using images of the iris and wavelet transform. *IEEE Trans on Signal Processing*, 46(4):1185–1188, April 1998.
5. H Bunke. String matching for structural pattern recognition. In *Syntactic and Structural Pattern Recognition – Theory and Applications*, pages 381–414. World Scientific Publ, 1990. isbn 978-9971-5-0566-0.

6. CASIA. `http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp`. 2010.
7. K-R Chen, C-T Chou, S-W Shih, W-S Chen, and D-Y Chen. Feature selection for iris recognition with AdaBoost. In *Proc of the 3rd Intl Conf Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, volume 2 of *IIH-MSP '07*, pages 411–414, 2007.
8. J Climent, J D Blanco, and R A Hexsel. Approximate string matching for iris recognition by means of boosted Gabor wavelets. In *23rd Conference on Graphics, Patterns and Images (SIBGRAPI'10)*, pages 40–47, August 2010.
9. J Climent, A Grau, J Aranda, and A Sanfeliu. Clique-to-clique distance computation using a specific architecture. In *Proc Int Workshop on Structural and Syntactic Pattern Recognition*, pages 405–412, 1998.
10. J G Daugman. High confidence visual recognition of persons by a test of statistical independence. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 15(11):1148–1161, 1993.
11. J G Daugman. How iris recognition works. *IEEE Trans on Circuits and Systems for Video Technology*, 14(1):21–30, January 2004.
12. J G Daugman. New methods in iris recognition. *IEEE Trans on Systems, Man and Cybernetics, part B: Cybernetics*, 37(5):1167–1175, October 2007.
13. M Dobeš, J Martinek, D Skoupil, Z Dobešová, and J Pospíšil. Human eye localization using the modified Hough transform. *Optik – Int Journal for Light and Electron Optics*, 117(10):468–473, 2006.
14. Y Freund and R E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conf on Computational Learning Theory*, pages 23–37, 1995.
15. Z He, Z Sun, T Tan, X Qiu, C Zhong, and W Dong. Boosting ordinal features for accurate and fast iris recognition. *IEEE Conf on Computer Vision and Pattern Recognition*, pages 1–8, 2008.
16. B V K Vijaya Kumar, C Xie, and J Thornton. Iris verification using correlation filters. In *Proc 4th Int Conf on Audio- and Video-Based Biometric Person Authentication*, LNCS 2688, pages 697–705, 2003.
17. S Lim, K Lee, O Byeon, and T Kim. Efficient iris recognition through improvement of feature vector and classifier. *ETRI Journal*, 23(2):61–70, 2001.
18. 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, June 2004.
19. L Ma, Y Wang, and T Tan. Iris recognition using circular symmetric filters. In *Int Conf on Pattern Recognition (ICPR'02)*, volume 2, pages 414–417, 2002.
20. A J Mansfield and J Wayman. Best practice standards for testing and reporting performance of biometric devices, version 2.1. Technical report, U.K. National Physical Laboratory, Middlesex, August 2002. `http://www.cesg.gov.uk/site/ast/biometrics/media/BestPractice.pdf`.
21. K Miyazawa, K Ito, T Aoki, K Kobayashi, and H Nakajima. An effective approach for iris recognition using phase-based image matching. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 30(10):1741–1756, 2008.
22. G Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.
23. H Proença and L A Alexandre. Toward non-cooperative iris recognition: A classification approach using multiple signatures. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 29(4):607–612, 2007.
24. J R Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.
25. M Ritt, A M Costa, V M Orengo, and S Mergen. An integer linear programming ap-

proach for approximate string comparison. *European Journal of Operational Research*, 198(3):706–714, November 2009.
26. R Sastry, N Ranganathan, and K Remedios. CASM: a VLSI chip for approximate string matching. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 17(8):824–830, August 1995.
27. Z Sun, T Tan, and X Qiu. Graph matching iris image blocks with local binary pattern. *Advances in Biometrics*, 3832:366–372, January 2006.
28. Q-C Tian, X-L Zhao, X-J Wu, L-S Li, and L Liu. Iris classifier enhanced algorithm based on AdaBoost. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques*, volume 2 of *Communications in Computer and Information Science*, pages 1001–1009. Springer, 2007.
29. Esko Ukkonen. Algorithms for approximate string matching. *Inf. Control*, 64(1-3):100–118, 1985.
30. UPOL. `http://phoenix.inf.upol.cz/iris`. 2010.
31. P Viola and M J Jones. Robust real-time face detection. *Int Journal Computer Vision*, 57(2):137–154, 2004.
32. A Wang, Y Chen, X Zhang, and J Wu. Iris recognition based on 2D wavelet and AdaBoost neural network. In *Trends in Intelligent Systems and Computer Engineering*, volume 6 of *L N in Electrical Engineering*, pages 117–128. Springer US, 2008.
33. J L Wayman, A K Jain, D Maltoni, and D Maio (Eds.). *Biometric Systems*. Springer-Verlag, 2005. isbn 978-1-85233-596-0.
34. R P Wildes. Iris recognition: An emerging biometric technology. *Proc of the IEEE*, 85(9):1348–1363, September 1997.
35. J Zuo, N K Ratha, and J H Connell. A new approach for iris segmentation. In *Proc Computer Vision and Pattern Recognition Workshops (CVPRW'08)*, pages 1–6, 2008.