New Self-Organizing Maps with Non-Conventional Metrics and Their Applications for Iris Recognition and Automatic Translation

VICTOR-EMIL NEAGOE
Department Electronics, Telecommunications and Information Technology
Polytechnic University of Bucharest
P.O. Box 16-37, Bucharest 16, Romania
ROMANIA

Abstract: - This paper has as aim the design and applications of two self-organizing maps using non-conventional metrics. First approach concerns the Levensthein Self-Organizing Map (LSOM). The LSOM is a SOM that uses a symbolic representation for both the input and also for the weight rows and it is based on the Levensthein metrics. The software implementation of the experimental LSOM model is designed for automatic Romanian-English translation of 1009 words and expressions. The second approach corresponds to the Hamming Self-Organizing Map (HSOM). The HSOM is a SOM that uses binary representation for input and weight vectors and is based on Hamming metric. We have implemented the HSOM for recognition of iris binary templates, and we have evaluated its performances for CASIA iris database with 108 subjects. A recognition score of 99.08% is obtained.

Key-Words: - Self-organizing map, Levensthein metric, Hamming metric, automatic translation, iris recognition

1 Introduction

In pattern recognition, the Self-Organizing Map (SOM) (often called Kohonen network [1], [2]) performs a high quality classification, assigning the similar input vectors to the same neuron or to neighbour neurons. Thus, this network transforms the relation of similarity between input vectors into a relation of neighbourhood of the neurons. The map uses the competition principle, by evaluating the distances between the input vector and the weight vectors corresponding to each neuron. Instead of using the classical Euclidean distance, there are recent approaches to use other metrics. The author of this paper proposed a syntactical SOM based on Levensthein metric [3]. There are also other approaches to apply the Hamming metric for a SOM with binary inputs and weights [4], [5].

2 Levensthein Self-Organizing Map (LSOM)

We further propose a syntactical Levensthein Self-Organizing Map (LSOM) that has significant differences by comparison to the conventional SOM.

2.1 Characteristics of LSOM

• Both the inputs and the weights of the LSOM are represented into a symbolic form, but not in a numerical one (like for SOM);
• For the LSOM, we eliminate the condition that the two representations used in the competition phase to have the same length. Instead of evaluating the Euclidean distance between two real vectors, belonging to the same space (for the case of conventional SOM), we evaluate the weighted Levensthein distance between two rows of symbols with different lengths, for the LSOM. For computing the above distance, we have used the Wagner-Fisher algorithm. The application of Levensthein distance is perfectly adequate for our task, where we compare words (expressions) of different lengths.
• The LSOM uses the competition principle (like SOM). One computes the Levensthein distances between the input row of symbols and all the rows of weights corresponding to the network neurons. The winner is the neuron that minimizes the above distances.

2.2 Levensthein metric

We further define the weighted Levensthein metrics ([3], [6], and [7]) in order to compute the distance between two rows of symbols of different lengths. Let α and β be two rows of symbols of lengths "m" and "n" defined as

$$\alpha = a_1 a_2 \ldots a_m$$
$$\beta = b_1 b_2 \ldots b_n$$

where a_i, b_j are symbols belonging to the same alphabet. We define the transformation

$$M : \alpha \to \beta$$

as being the set of ordered pairs (i, j), where $1 \le i \le m, 1 \le j \le n$. We denote

$$I = \{i \mid (i, j) \in M\}$$
$$J = \{j \mid (i, j) \in M\}$$

One can prove that the M transformation satisfies the following interpretations:

(1) If $a_i \ne b_j$, for $(i, j) \in M$, then b_j is substituted by a_i;

(2) If $j \notin J$, then b_j is inserted;

(3) If $i \notin I$, then a_i is eliminated.

Let M be the set of all the transformations from α to β. Then, the minimum cost of a transformation from α to β, denoted by D (α, β) is defined as follows:

$$D(\alpha, \beta) = \min_{M \in \tilde{M}} \left[\sum_{(i,j) \in M} p(i, j) + (n - |J|) * q + (m - |I|) * r \right]$$

where $|I|, |J|$ are the cardinals of I, respectively of J, q is the insertion cost, r is the elimination cost, and

$$p(i, j) = \begin{cases} 0, a_i = b_j \\ p, a_i \ne b_j \end{cases}$$

is the cost of substitution of symbol a_i by b_j.

The above-defined distance is called weighted Levensthein distance if the following two conditions are fulfilled for any pair $(i_1, j_1), (i_2, j_2)$:

(1) $i_1 = i_2 \Leftrightarrow j_1 = j_2$

(2) $i_1 < i_2 \Leftrightarrow j_1 < j_2$

2.3 Wagner-Fisher algorithm

In order to iteratively compute the Levensthein distance, we have used the Wagner-Fisher algorithm (given in [3], [6], and [7]), having the following steps:

Step 1. D (0, 0) = 0;

Step 2. For i from 1 to n, one computes

D (i, 0)=D (i-1, 0)+ $c_1(a_i)$;

Step 3. For j from 1 to m, one computes

D (0, j)=D (0, j-1)+ $c_2(b_j)$;

Step 4. For i from 1 to n and j from 1 to m, one computes:

$$A_1 = D(i-1, j-1) + c(a_i, b_j);$$
$$A_2 = D(i-1, j) + c_1(a_i);$$
$$A_3 = D(i, j-1) + c_2(b_j);$$
$$D(i, j) = \min(A_1, A_2, A_3);$$

where $c_1(a_i), c_2(b_j)$, and $c(a_i, b_j)$ are the costs of insertion, elimination, and respectively substitution.

Step 5. The distance between the two words is D (n, m).

2.4 LSOM Training

Training of the LSOM (refinement of the weights) is performed by analogy to the procedure of conventional SOM, in a neighbourhood of the winner neuron using the Levensthein distance between the input row and the row of weights characterizing the winner neuron (or one of its neighbours). One tends to change the weight row in order to minimize the Levensthein distance. To solve this task, one uses the following operations: insertion, deletion and/or substitution.

The training algorithm for LSOM is the following:

Step 1.
Initialize the weights of the LSOM

Step 2.

• Apply one by one the words (expressions) belonging to the training set.

• For each of them, compute the winner neuron, by minimizing the Levenshein distance between the input word and all the weight rows corresponding to the LSOM neurons.

• Make identical the weight row of the winner neuron with that of the input word (corresponding row).

• Refine the weight rows of the neurons belonging to the neighbourhood of the winner by performing the elementary operations of substitution, insertion, and deletion, in order to reduce their Levensthein distances to the input word (but not to make them zero). The reduction of the Levensthein distance is a function of the neuron position regarding the winner; this reduction increases when the Euclidean distance (in the map co-ordinates) between the corresponding neuron and the winner neuron decreases.

Step 3.
• Compute the classification error as a sum of all the minimum Levenshein distances of the words (expressions) belonging to the training lot. Such a distance is the minimum of the distances between the corresponding input word and the weight rows of the LSOM neurons.
Step 4.
• Test the stop condition (if the classification error is zero).

2.5 LSOM Test

The test (operational) phase consists of computing the Levensthein distance between the input word and all the weight rows of the LSOM. The minimum distance leads to the winner neuron; its label corresponds to the class label.

3 Hamming Self-Organizing Map (HSOM)

We shall present a SOM variant based on Hamming metric, called Hamming Self-Organizing Map (HSOM). It can be considered as a particular case of the previous LSOM.

3.1 Characteristics of HSOM

• Both the input vector elements and the weights of the HSOM are represented as binary integers "0" or "1";
• The HSOM is based on Hamming distance. Assuming two binary vectors,

$\mathbf{A}=(a_1, a_2, \ldots, a_n)^t$
$\mathbf{B}=(b_1, b_2, \ldots, b_n)^t$

where $a_i, b_i \in \{0, 1\}$, the Hamming distance between **A** and **B** is

$$d_H(\mathbf{A}, \mathbf{B}) = \left(\sum_{i=1}^{n} |a_i - b_i| \right)$$

Consequently, the Hamming distance between the input vector **X** and the weight vector $\mathbf{W_j}$ of the j-th neuron in the competitive layer is calculated by the equation

$d_H(\mathbf{X}, \mathbf{W_j}) = \text{bitcount}\{ x_i \text{ XOR } w_{ji} \}$

$d_H(\mathbf{X}, \mathbf{W_j}) = \text{bitcount}\{(\overline{x_i} \wedge w_{ji}) \vee (x_i \wedge \overline{w_{ji}}) \mid i = 1, \ldots, n\}$, j=1, …, M; M = number of output neurons.

• The HSOM uses the competition principle (like SOM). One computes the Hamming distances between the binary input vector and all the binary weight vectors. The winner is the neuron that minimizes the above distances

$$c = \arg\min_j \{d_H(\mathbf{X}, \mathbf{W_j})\}$$

3.2 HSOM Training

To update the binary weight vectors of HSOM, we firstly compute exclusive-OR (XOR) of each element of **X** and $\mathbf{W_j}$. If XOR (x_i, w_{ji}) = 1, then w_{ji} is a candidate for inversion. The number of inverted bits (belonging to the weight vector $\mathbf{W_j}$) is defined as a learning rate; it gradually decreases as learning progresses.

4 LSOM for Automatic Translation

There are many automatic translation systems using very sophisticated techniques, most of them being non-neural. They take into account both the dictionary of correspondence between the words of the two languages and also the syntax rules. The design of such a system is laborious and usually it is specific for the two languages.

We further present the experimental results of the software implementation of the LSOM model, corresponding to a Romanian-English translation application, using a set of 1009 Romanian words and expressions. One chooses a circular architecture [8] of the LSOM (Fig. 1). Denoting by "*n*" the number of inputs (representing the maximum number of symbols belonging to an input row) and by *"M"* he number of outputs (neurons), the software experiment corresponds to*: n =20* and *M= 1050.* The weight matrix is *w(i,j), i= 1, 2, ..., n; j=1, 2, ...,M.*

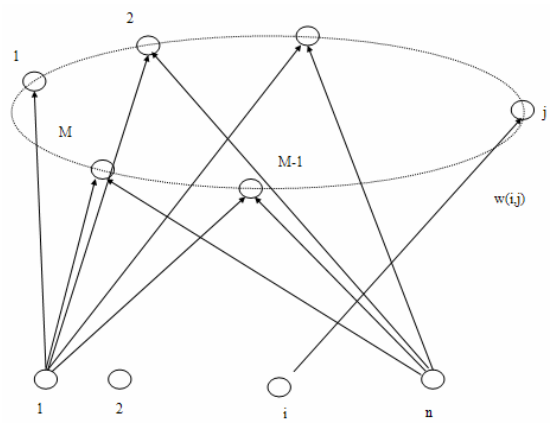

Fig. 1. The circular architecture of the LSOM.

0 5 3 2 2 3 4 5 7 6 7 8 7 6 8 9 1 3 1 4 6 4 1 2 5 1 3
5 0 2 3 4 2 1 1 4 2 3 4 2 1 5 5 6 3 4 2 2 1 5 3 2 4 1
3 2 0 1 2 1 2 3 5 4 5 6 4 3 6 7 4 2 2 2 4 1 3 1 3 2 1
2 3 1 0 1 1 2 3 5 4 5 6 5 4 6 7 3 1 1 2 4 2 2 1 3 2 2
2 4 2 1 0 2 3 4 5 5 6 7 6 5 6 7 2 1 1 2 4 3 1 2 3 2 3
3 2 1 1 2 0 1 2 4 3 4 5 4 3 5 6 4 1 2 1 3 1 3 2 2 3 2
4 1 2 2 3 1 0 1 3 2 3 4 3 2 4 5 6 2 3 1 2 1 4 3 1 4 2
5 1 3 3 4 2 1 0 2 1 2 3 2 1 3 4 6 3 4 1 1 2 5 4 1 5 2
7 4 5 5 5 4 3 2 0 1 1 1 2 2 1 2 7 4 5 3 1 4 6 6 2 7 3
6 2 4 4 5 3 2 1 1 0 1 2 1 1 2 3 7 4 5 3 1 3 6 5 2 6 2
7 3 5 5 6 4 3 2 1 1 0 1 1 2 1 2 8 5 6 4 2 4 7 6 3 7 2
8 4 6 6 7 5 4 3 1 2 1 0 2 3 1 1 9 6 7 5 3 5 8 7 4 8 2
7 2 4 5 6 4 3 2 2 1 1 2 0 1 2 3 8 5 6 4 2 3 7 5 3 6 1
6 1 3 4 5 3 2 1 2 1 2 3 1 0 3 4 7 4 5 3 2 2 6 4 2 5 1
9 5 6 6 6 5 4 3 1 2 1 1 2 3 0 1 8 5 7 4 2 5 7 7 3 8 3
9 5 7 7 7 6 5 4 2 3 2 1 3 4 1 0 9 6 8 5 3 6 8 8 4 9 3
1 6 4 3 2 4 6 6 7 7 8 9 8 7 8 9 0 3 2 4 6 5 1 3 5 2 3
3 3 2 1 1 1 2 3 4 4 5 6 5 4 5 6 3 0 2 1 3 2 2 2 2 3 3
1 4 2 1 1 2 3 4 5 5 6 7 6 5 7 8 2 2 0 3 5 3 1 1 4 1 2
4 2 2 2 2 1 1 1 3 3 4 5 4 3 4 5 4 1 3 0 2 2 3 3 1 4 3
6 2 4 4 4 3 2 1 1 1 2 3 2 2 2 3 6 3 5 2 0 3 5 5 1 5 3
4 1 1 2 3 1 1 2 4 3 4 5 3 2 5 6 5 2 3 2 3 0 4 2 2 3 1
1 5 3 2 1 3 4 5 6 6 7 8 7 6 7 8 1 2 1 3 5 4 0 2 4 2 3
2 3 1 1 2 2 3 4 6 5 6 7 5 4 7 8 3 2 1 3 5 2 2 0 4 1 1
5 2 3 3 3 2 1 1 2 2 3 4 3 2 3 4 5 2 4 1 1 2 4 4 0 5 3
1 4 2 2 2 3 4 5 7 6 7 8 6 5 8 9 2 3 1 4 5 3 2 1 5 0 2
3 1 1 2 3 2 2 2 3 2 2 2 1 1 3 3 3 3 2 2 3 3 1 3 1 3 2 0

Fig. 2. The "27 x 27" matrix of substitution costs

The input alphabet has 27 symbols, namely 26 letters and the "space" symbol (for expression translation):

A={ a b c d e f g h i j k l m n o p q r s t u v w x y z }.

For using the Levensthein metrics, we have chosen the substitution costs, insertion costs, and deletion costs. The substitution costs are defined as a function of the distance between the corresponding letters on the keyboard, in order to correct the punch errors (for example, c (a, b)=5, c (a, c) =3, c (a, d) =2). Taking into account the above principle, we have defined the corresponding 27 x 27 matrix of substitution costs. (Fig. 2). The insertion, respectively the deletion costs are taken all equal to 2.

For training, we used a variable size neighbourhood, where the radius of neighbourhood decreases with the epoch index, according to the following sequence:

{ 256, 256, 256, 128, 128, 128, 32, 32, 32, 8, 8, 8, 2, 2, 2 }.

To perform an automatic translation, we associate a table to the output labelled neurons giving the correspondence between the two languages: Romanian and English. At the same time, taking into account the LSOM design, one can correct some input (typing) errors. In Table 1, we give an example.

Table 1. Example of automatic translation and error correction.

Original Word (Ro)	Erroneous Input (Ro)	Output (En)	Error Correction
abil	Abi	Skilful	Yes
abur	Abuv	Steam	Yes
lemn	Lenm	Wood	Yes
admite	Dmit	Bribe	No

5 HSOM for Iris Recognition

Since 1987, when Flom and Safir [9] observed the stability of iris morphology over human lifetime and estimated the probability for the existence of two similar irises at 1 in 1072, the use of iris biometrics has been increasingly encouraged by both government and private entities. The iris is commonly recognized as one of the most reliable biometric traits[10], [11], [12]. We shall further apply the HSOM for binary iris template recognition.

5.1 CASIA Iris Database

The Chinese Academy of Sciences – Institute of Automation (CASIA) eye image database contains 756 grey scale eye images with 108 unique eyes or classes and 7 different images of each unique eye. Images from each class are taken from two sessions with one month interval between sessions. The images were captured especially for iris recognition research using specialised digital optics developed by the National Laboratory of Pattern Recognition, China. The eye images are mainly from persons of Asian decent, whose eyes are characterised by irises that are densely pigmented, and with dark eyelashes. Due to specialized imaging conditions using near infra-red light, features in the iris region are highly visible and there is good contrast between pupil, iris and sclera regions.

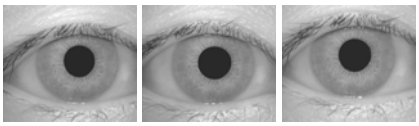

Fig.3. CASIA images of the first session.

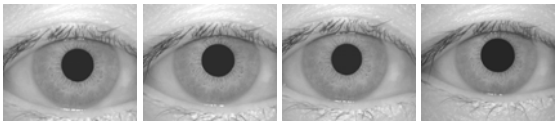

Fig.4. Corresponding images of the second session.

5.2 The experimental system for iris recognition

The experimental software implemented system for iris recognition has the following stages (Fig. 5):

a) Iris detection
b) Normalisation
c) Feature extraction and encoding
d) Hamming Self-Organizing Map

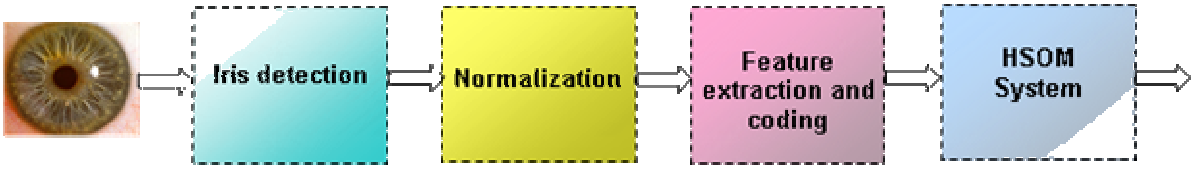

Fig. 5. Architecture of the implemented software for iris recognition including HSOM

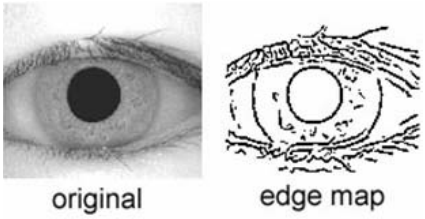

Fig. 6. a) an eye image from the CASIA database; b) corresponding edge map

1. Iris detection implies three steps:
a) edge detection
b) detection of the borders of iris region
c) eyelash and noise isolation

For *edge detection*, an edge map is generated by calculating the first derivatives of intensity values in an eye image and then thresholding the result (Fig. 6). From the edge map, votes are cast in Hough space for the parameters of circles passing through each edge point. The circular Hough transform is employed to deduce the radius and centre coordinates of the pupil and iris regions. The results are shown in Fig. 7.

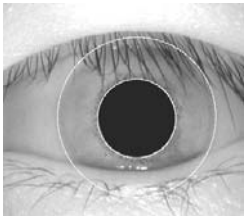

Fig. 7. Detection of iris borders using Hough transform.

Eyelids were isolated by first fitting a line to the upper and lower eyelid using the linear Hough transform. A second horizontal line is then drawn, which intersects with the first line at the iris edge that is closest to the pupil. This process is illustrated in Fig. 8 and is done for both the top and bottom eyelids.

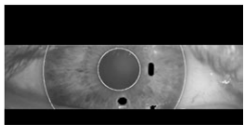

Fig. 8. Isolation of eye lids using a linear Hough transform.

Separable eyelashes are detected using 1D Gabor filters, since the convolution of a separable eyelash with the Gaussian smoothing function results in a low output value. Thus, if a resultant point is smaller than a threshold, it is noted that this point belongs to an eyelash (Fig. 9)

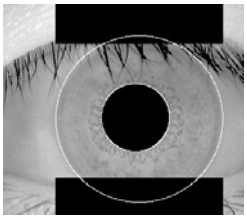

Fig. 9. Eye lash and noise detection.

For normalisation, the homogenous rubber sheet model devised by Daugman [1] remaps each point within the iris region to a pair of polar coordinates (r,θ) where r is on the interval [0,1] and θ is angle $[0,2\pi]$ (Fig. 10).

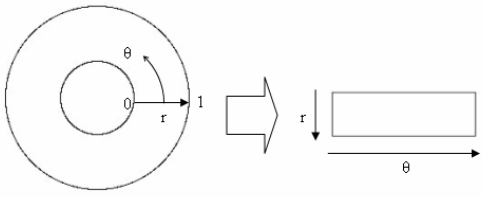

Fig.10. Daugman's rubber sheet model.

The output of the normalisation stage is given in Figs. 11 and 12.

Fig. 11. Polar normalisation of iris region.

Fig.12. Noise mask corresponding to polar normalisation.

Feature encoding was implemented by convolving the normalised iris pattern with 1D Log-Gabor wavelets. The 2D normalised pattern is broken up into a number of 1D signals, and then these 1D signals are convolved with 1D Gabor wavelets. The output of filtering is then phase quantized to four levels using the Daugman method [10], with each filter producing two bits of data for each phasor. The output of phase quantization is chosen to be a grey code, so that when going from one quadrant to another, only one bit changes.

The encoding process produces a bitwise template containing a number of bits of information, and a corresponding noise mask which corresponds to corrupt areas within the iris pattern, and marks bits in the template as corrupt (figs. 13 ands 14).

Fig. 13. Binary iris template.

Fig. 14. Binary noise mask.

For each subject (belonging to the set of 108 of the database CASIA), the three images of the first session have been considered for training, while three of the four images of the second session have been included in the test set. Thus, the learning set contained 324 images and the test set had 324 as well. Using HSOM, one obtains a correct recognition score of 98.15%.

To improve the recognition performance, we have chosen a decision fusion; namely, for each subject we performed a classification for each of the three images of the test set corresponding to this subject, then we combined the results by decision fusion. Thus, one obtains a recognition score of 99.08%.

6 Conclusions

1. We propose a Levensthein Self-Organizing Map (LSOM), where input rows and the SOM weights are represented by rows of symbols of different lengths, by eliminating the necessity of normalizing the length of input . One uses the specific weighted Levensthein distance, which is computed by applying the Wagner-Fisher algorithm.
2. The syntactical way of representation of the input words and expressions is naturally adequate to the automatic translation. Any substitution cost given in the matrix of Fig. 1 is defined as a function of the Euclidean distance between the corresponding letters on the keyboard, in order to correct the input (typing) errors.
3. An important advantage of the model is that its design does not depend on the specific two languages, characterizing the translation. Consequently, it can be trained and retrained by the user for any pair of languages.
4. The second approach has as target the Hamming Self-Organizing Map (HSOM). Both the input vector elements and the weights of the HSOM are represented as binary integers “0” or “1”. One uses the specific Hamming distance in the competition phase.
5. We have implemented the HSOM for recognition of iris binary templates, and we have evaluated its performances for CASIA iris database with 108 subjects. A recognition score of 99.08% is obtained.

References:

[1] T. Kohonen, “The Self-Organizing Map”, *Proceedings IEEE*, Vol. 78, No. 9, Sept. 1990, pp. 1464-1479.

[2] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, 1995.

[3] V. E. Neagoe, “A Neural Error Correcting Approach Based on Levensthein Metrics for Automatic Romanian-English Translation”, In: *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, July 22-25, 2001*, Orlando, Florida, USA, Vol. XIII, pp. 14-17, ISBN: 980-07-7553-6.

[4] R. Kubota, K. Horio and T. Yamakawa, Reproduction Strategy Based on Self-Organizing Map for Genetic Algorithms, *International Journal of Innovative Computing, Information and Control (ICIC International)*, Vol. 1, No. 4, Dec. 2005, pp. 595—607, ISSN 1349-4198.

[5] T. Yamakawa, K. Horio, T. Hiratsuka, Advanced self-organizing maps using binary weight vector and its digital hardware design, *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP)*, 18-22 Nov. 2002, Vol. 3, pp. 1330 – 1335.

[6] G. Ferrate et al. (eds.), *Syntactic and Structural Pattern Recognition*, NATO ASI Series, Vol. F45, Springer-Verlag, Berlin, 1988.

[7] V. Neagoe, O. Stanasila, *Teoria recunoasterii formelor (Pattern Recognition Theory)*, Romanian Academy Publishing House, Bucharest, 1992.

[8] V. E. Neagoe, "A Circular Kohonen Network for Image Vector Quantization", *Parallel Computing: State of the Art and Perspectives* (E. H. D'Hollander, G. R Joubert, F. J. Peters and D. Trystram eds.), Vol. 11, Amsterdam-New York: Elsevier, 1996, pp. 677-680.

[9] L. Flom and A. Safir, "*Iris Recognition System*", US Patent 4 641 394, 1987.

[10] J. G. Daugman, "High Confidence Visual Recognition of Persons by a Test of Statistical Independence," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 25, No. 11, pp. 1148-1161, Nov. 1993.

[11] R. P. Wildes, "Iris Recognition: An Emerging Biometric Technology", *Proc. IEEE*, Vol. 85, No. 9, pp. 1348-1363, Sept. 1997.

[12] L. Masek, *Recognition of Human Iris Patterns for Biometric Identification*, B.S. thesis, School of Computer Science and Software Engineering, The University of Western Australia, 2003.