Fast Normalized Neural Processors for Pattern Detection Based on Cross Correlation Implemented in the Frequency Domain

Hazem M. El-Bakry and Qiangfu Zhao

University of Aizu, Aizu Wakamatsu, Japan 965-8580
Phone: +81-242-37-2519, Fax: +81-242-37-2743,
E-mail: d8071106@u-aizu.ac.jp

Neural networks have shown good results for detecting a certain pattern in a given image. In this paper, fast neural networks for pattern detection are presented. Such neural processors are designed based on cross correlation in the frequency domain between the input image and the weights of neural networks. New general formulas for fast cross correlation as well as the speed up ratio are given. Also, commutative cross correlation is achieved. Furthermore, an approach to reduce the computation steps required by these fast neural networks for the searching process is presented. The principle of divide and conquer strategy is applied through image decomposition. Each image is divided into small in size sub-images and then each one is tested separately by using a single fast neural processor. Compared with conventional and fast neural networks, experimental results show that a speed up ratio is achieved when applying this technique to locate different patterns automatically in cluttered scenes. Furthermore, faster pattern detection is obtained by using parallel processing techniques to test the resulting sub-images at the same time using the same number of fast neural networks. In contrast to fast neural networks, the speed up ratio is increased with the size of the input image when using fast neural networks and image decomposition. Our previous work solved the problem of local sub-image normalization in the frequency domain. Here, the effect of image normalization on the speed up ratio of pattern detection is presented. Simulation results show that local sub-image normalization through weight normalization is faster than sub-image normalization in the spatial domain. Moreover, the overall speed up ratio of the detection process is increased as the normalization of weights is done off line.

Keywords—Fast Pattern Detection, Neural Networks, Cross Correlation, Image Normalization
ACM Classification: I.5

1. INTRODUCTION

Pattern detection is a fundamental step before pattern recognition. Its reliability and performance have a major influence in a whole pattern recognition system. Nowadays, neural networks have shown very good results for detecting a certain pattern in a given image (Baluja *et al*, 1998). But the problem with neural networks is that the computational complexity is very high. This is because the networks have to process many small local windows in the images as stated by Zhu *et al* (2000), and Srisuk (2002). Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999) tried to speed up the

Manuscript received: 12 May 2005
Communicating Editor: Geoff McLachlan

detection process of neural networks. They proposed a multilayer perceptron (MLP) algorithm for fast object/face detection. The same authors claimed incorrect equation for cross correlation between the input image and the weights of the neural networks. They introduced formulas for the number of computation steps needed by conventional and fast neural networks. Then, they established an equation for the speed up ratio. Unfortunately, these formulas contain many errors which lead to invalid speed up ratio. Other authors developed their work based on these incorrect equations (El-Bakry, 2001). Therefore, the fact that these equations are not valid must be clarified to all researchers. It is not only very important but also urgent to notify other researchers not to waste their time and effort doing research based on wrong equations. The main objective of this paper is to correct the formulas of cross correlation as well as the equations which describe the computation steps required by conventional and fast neural networks presented by Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999). Some of these wrong equations were corrected in previous publications (El-Bakry, 2003). Here, all of these errors are corrected. Mathematical proof and simulation results for fast testing of the new proposed algorithm using Matlab are given. Furthermore, more results have been obtained since those presented in our previous paper (El-Bakry, 2005).

The problem of sub-image (local) normalization in the Fourier space was presented by Feraud *et al* (2000). This problem was solved by El-Bakry (2002). It was proven that the number of computation steps required for weight normalization is less than that needed for image normalization. However, the effect of normalization on the speed up ratio was not discussed. Here, the effect of weight normalization on the speed up ratio is theoretically and practically discussed. Mathematical calculations prove that the new idea of weight normalization, instead of image normalization, provides good results and increases the speed up ratio. This is because weight normalization requires fewer computation steps than sub-image normalization. Moreover, for neural networks, normalization of weights can be easily done off line before starting the search process.

In Section 2, fast neural networks for pattern detection are described. Comments on conventional neural networks, fast neural networks, and the speed up ratio of pattern detection are given. A faster searching algorithm for pattern detection that reduces the number of the required computation steps through image decomposition is presented in Section 3. Accelerating the new approach using parallel processing techniques is also introduced. Sub-image normalization in the frequency domain through normalization of weights is introduced in Section 4. The effect of weight normalization on the speed up ratio is presented in Section 5.

2. FAST PATTERN DETECTION USING MLP AND FFT

A fast algorithm for object/face detection based on two dimensional cross correlations that take place between the tested image and the sliding window (20x20 pixels) was described by Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999). Such window is represented by the neural network weights situated between the input unit and the hidden layer. The convolution theorem in mathematical analysis says that a convolution of f with h is identical to the result of the following steps: let F and H be the results of the Fourier transformation of f and h in the frequency domain. Multiply F and H in the frequency domain point by point and then transform this product into spatial domain via the inverse Fourier transform. As a result, these cross correlations can be represented by a product in the frequency domain. Thus, by using cross correlation in the frequency domain a speed up in an order of magnitude can be achieved during the detection process (Ben Yakoub, 1997; Fasel, 1998 and Ben Yakoub *et al*, 1999).

In the detection phase, a sub-image I of size mxn (sliding window) is extracted from the tested image, which has a size PxT, and fed to the neural network. Let W_i be the vector of weights between

the input sub-image and the hidden layer. This vector has a size of mxn and can be represented as an mxn matrix. The output of hidden neurons h_i can be calculated as follows:

$$h_i = g\left(\sum_{j=1}^{m}\sum_{k=1}^{n} W_i(j,k)I(j,k) + b_i\right) \quad (1)$$

where g is the activation function and b_i is the bias of each hidden neuron (i). Eq.1 represents the output of each hidden neuron for a particular sub-image I. It can be computed for the whole image Ψ as follows:

$$h_i(u,v) = g\left(\sum_{j=-m/2}^{m/2}\sum_{k=-n/2}^{n/2} W_i(j,k)\ \Psi(u+j, v+k) + b_i\right) \quad (2)$$

Eq.2 represents a cross correlation operation. Given any two functions f and g, their cross correlation can be obtained by [2]:

$$f(x,y)\otimes g(x,y) = \left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(m,n)g(x+m,y+n)\right) \quad (3)$$

Therefore, Eq.2 may be written as follows (Ben Yakoub, 1997 and Fasel, 1998):

$$h_i = g\left(W_i \otimes \Psi + b_i\right) \quad (4)$$

where h_i is the output of the hidden neuron (i) and h_i (u,v) is the activity of the hidden unit (i) when the sliding window is located at position (u,v) in the input image Ψ and (u,v) $\in$[P-m+1,T-n+1].

Now, the above cross correlation can be expressed in terms of the Fourier Transform:

$$\Psi \otimes W_i = F^{-1}\left(F(\Psi)\bullet F^{*}\left(W_i\right)\right) \quad (5)$$

(*) means the conjugate of the FFT for the weight matrix. Hence, by evaluating this cross correlation, a speed up ratio can be obtained comparable to conventional neural networks. Also, the final output of the neural network can be evaluated as follows:

$$O(u,v) = g\left(\sum_{i=1}^{q} w_O(i)\, h_i(u,v) + b_O\right) \quad (6)$$

where q is the number of neurons in the hidden layer. O(u,v) is the output of the neural network when the sliding window located at the position (u,v) in the input image Ψ.

Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999) analyzed their proposed fast neural network as follows: For a tested image of NxN pixels, the 2D-FFT requires $O(N^2(\log_2 N)^2)$ computation steps. For the weight matrix W_i, the 2D-FFT can be computed off line since these are constant parameters of the network independent of the tested image. The 2D-FFT of the tested image must be computed. As a result, q backward and one forward transforms have to be computed. Therefore, for a tested image, the total number of the 2D-FFT to compute is $(q+1)N^2(\log_2 N)^2$. In addition, the input image and the weights should be multiplied in the frequency domain. Therefore, computation steps of (qN^2) should be added. This yields a total of $O((q+1)N^2(\log_2 N)^2+qN^2)$ computation steps for the fast neural network.

Using sliding window of size nxn, for the same image of NxN pixels, qN^2n^2 computation steps are required when using traditional neural networks for the face detection process. The theoretical

speed up factor η can be evaluated as follows (Ben Yakoub, 1997):

$$\eta = \frac{qn^2}{(q+1)\log^2 N} \tag{7}$$

The speed up factor introduced by Ben Yakoub (1997) and given by Eq.7 is not correct for the following reasons:

1. The number of computation steps required for the 2D-FFT is $O(N^2\log_2 N^2)$ and not $O(N^2\log^2 N)$ as presented by Ben Yakoub (1997) and Fasel (1998). Also, this is not a typing error as the curve in Figure 2 in Ben Yakoub (1997) realizes Eq.7, and the curves in Figure 15 in Fasel (1998) realizes Eq.31 and Eq.32 in Fasel (1998).

2. Also, the speed up ratio presented by Ben Yakoub (1997) not only contains an error but also is not precise. This is because for fast neural networks, the term $(6qN^2)$ corresponds to a complex dot product in the frequency domain and must be added. Such term has a great effect on the speed up ratio. Adding only qN^2 as stated by Fasel (1998) is not correct since a one complex multiplication requires six real computation steps.

3. For conventional neural networks, the number of operations is $(q(2n^2-1)(N-n+1)^2)$ and not (qN^2n^2). The term n^2 is required for multiplication of n^2 elements (in the input window) by n^2 weights which results in another new n^2 elements. Adding these n^2 elements, requires another (n^2-1) steps. So, the total computation steps needed for each window is $(2n^2-1)$. The search operation for a face in the input image uses a window with nxn weights. This operation is done at each pixel in the input image. Therefore, such process is repeated $(N-n+1)^2$ times and not N^2 as stated by Ben Yakoub (1997) and Ben Yakoub *et al* (1999).

4. Before applying cross correlation, the 2D-FFT of the weight matrix must be computed. Because of the dot product, which is done in the frequency domain, the size of weight matrix should be increased to be the same as the size of the input image. Computing the 2D-FFT of the weight matrix off line as stated by Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999) is not practical. In this case, all of the input images must have the same size. As a result, the input image will have only a one fixed size. This means that, the testing time for an image of size 50x50 pixels will be the same as that image of size 1000x1000 pixels and of course, this is unreliable. So, another number of complex computation steps to perform 2D-FFT for (NxN) matrix should be added to the complex number of computation steps (σ) required by the fast neural networks as follows:

   $$\sigma=((2q+1)(N^2\log_2 N^2) + 6qN^2) \tag{8}$$

   This will increase the computation steps required for the fast neural networks especially when q is more than one neuron.

5. It is not valid to compare the number of complex computation steps by another of real computation steps directly. The number of computation steps given by Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999) for conventional neural networks is for real operations while that which is required by the fast neural networks is for complex operations. To obtain the speed up ratio, Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999) have divided the two formulas directly without converting the number of computation steps required by the fast neural networks into a real version. It is known that the two dimensions Fast Fourier Transform

requires $(N^2/2)\log_2 N^2$ complex multiplications and $N^2\log_2 N^2$ complex additions. Every complex multiplication is realized by six real floating point operations and every complex addition is implemented by two real floating point operations. Therefore, the total number of computation steps required to obtain the 2D-FFT of an NxN image is:

$$\rho=6((N^2/2)\log_2 N^2) + 2(N^2\log_2 N^2) \quad (9)$$

which may be simplified to:

$$\rho=5(N^2\log_2 N^2) \quad (10)$$

6. For the weight matrix to have the same size as the input image, a number of zeros = (N^2-n^2) must be added to the weight matrix. This requires a total real number of computation steps = $q(N^2-n^2)$ for all neurons. Moreover, after computing the 2D-FFT for the weight matrix, the conjugate of this matrix must be obtained. So, a real number of computation steps =qN^2 should be added in order to obtain the conjugate of the weight matrix for all neurons. Also, a number of real computation steps equal to N is required to create butterflies complex numbers ($e^{-jk(2\Pi n/N)}$), where 0<K<L. These (N/2) complex numbers are multiplied by the elements of the input image or by previous complex numbers during the computation of 2D-FFT. To create a complex number requires two real floating point operations. Thus, the total number of computation steps required by the fast neural networks is:

$$\sigma=((2q+1)(5N^2\log_2 N^2) +6qN^2+q(N^2-n^2)+qN^2 +N) \quad (11)$$

which can be reformulated as:

$$\sigma=((2q+1)(5N^2\log_2 N^2) +q(8N^2-n^2) +N) \quad (12)$$

Therefore, the correct speed up ratio is as follows:

$$\eta = \frac{q(2n^2-1)(N-n+1)^2}{(2q+1)(5N^2\log_2 N^2) + q(8N^2 - n^2) + N} \quad (13)$$

The correct theoretical speed up ratio (Eq.13) with different sizes of the input image and different in size weight matrices is listed in Table 1. Practical speed up ratio for manipulating images of different sizes and different in size weight matrices is listed in Table 2 using a 700 MHz processor and Matlab ver 5.3.

For general fast cross correlation the speed up ratio comes in the following form:

$$\eta = \frac{q(2n^2-1)N^2}{(2q+1)(5(N+\tau)^2\log_2 (N+\tau)^2) + q(8(N+\tau)^2 - n^2) + (N+\tau)} \quad (14)$$

where τ is a small number depends on the size of the weight matrix. General cross correlation means that the process starts from the first element in the input matrix. The theoretical speed up ratio for general fast cross correlation (Eq.14) is shown in Table 3. Compared with MATLAB cross correlation function (xcorr2), experimental results show that the proposed algorithm is faster than this function as shown in Table 4.

7. Furthermore, there are critical errors in Eq.3 and Eq.4 (which is Eq.4 in Ben Yakoub (1997) and also Eq.13 in Fasel (1998). Eq.3 is not correct because the definition of cross correlation is:

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	3.67	5.04	6.34
200x200	4.01	5.92	8.05
300x300	4.00	6.03	8.37
400x400	3.95	6.01	8.42
500x500	3.89	5.95	8.39
600x600	3.83	5.88	8.33
700x700	3.78	5.82	8.26
800x800	3.73	5.76	8.19
900x900	3.69	5.70	8.12
1000x1000	3.65	5.65	8.05
1100x1100	3.62	5.60	7.99
1200x1200	3.58	5.55	7.93
1300x1300	3.55	5.51	7.93
1400x1400	3.53	5.47	7.82
1500x1500	3.50	5.43	7.77
1600x1600	3.48	5.43	7.72
1700x1700	3.45	5.37	7.68
1800x1800	3.43	5.34	7.64
1900x1900	3.41	5.31	7.60
2000x2000	3.40	5.28	7.56

Table 1: The theoretical speed up ratio for images with different sizes.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	7.88	10.75	14.69
200x200	6.21	9.19	13.17
300x300	5.54	8.43	12.21
400x400	4.78	7.45	11.41
500x500	4.68	7.13	10.79
600x600	4.46	6.97	10.28
700x700	4.34	6.83	9.81
800x800	4.27	6.68	9.60
900x900	4.31	6.79	9.72
1000x1000	4.19	6.59	9.46
1100x1100	4.24	6.66	9.62
1200x1200	4.20	6.62	9.57
1300x1300	4.17	6.57	9.53
1400x1400	4.13	6.53	9.49
1500x1500	4.10	6.49	9.45
1600x1600	4.07	6.45	9.41
1700x1700	4.03	6.41	9.37
1800x1800	4.00	6.38	9.32
1900x1900	3.97	6.35	9.28
2000x2000	3.94	6.31	9.25

Table 2: Practical Speed up ratio for images with different sizes Using MATLAB ver 5.3.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	5.39	8.36	11.95
200x200	4.81	7.49	10.75
300x300	4.51	7.03	10.16
400x400	4.32	6.73	9.68
500x500	4.18	6.52	9.37
600x600	4.07	6.35	9.13
700x700	3.99	6.21	8.94
800x800	3.91	6.10	8.77
900x900	3.84	6.00	8.63
1000x1000	3.78	5.91	8.51
1100x1100	3.74	5.85	8.43
1200x1200	3.70	5.78	8.33
1300x1300	3.66	5.72	8.24
1400x1400	3.62	5.66	8.16
1500x1500	3.59	5.61	8.08
1600x1600	3.56	5.57	8.02
1700x1700	3.53	5.52	7.95
1800x1800	3.51	5.48	7.89
1900x1900	3.48	5.44	7.84
2000x2000	3.45	5.40	7.79

Table 3: The theoretical speed up ratio for the general fast cross correlation algorithm.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	10.14	13.05	16.49
200x200	9.17	11.92	14.33
300x300	8.25	10.83	13.41
400x400	7.91	9.62	12.65
500x500	6.77	9.24	11.77
600x600	6.46	8.89	11.19
700x700	5.99	8.47	10.96
800x800	5.48	8.74	10.32
900x900	5.31	8.43	10.66
1000x1000	5.91	8.66	10.51
1100x1100	5.77	8.61	10.46
1200x1200	5.68	8.56	10.40
1300x1300	5.62	8.52	10.35
1400x1400	5.58	8.47	10.31
1500x1500	5.54	8.43	10.26
1600x1600	5.50	8.39	10.22
1700x1700	5.46	8.33	10.18
1800x1800	5.42	8.28	10.14
1900x1900	5.38	8.24	10.10
2000x2000	5.34	8.20	10.06

Table 4: Simulation results of the speed up ratio for the general fast cross correlation compared with the MATLAB cross correlation function (xcorr2).

$$f(x,y)\otimes g(x,y)=\left(\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}f(x+m,y+n)g(m,n)\right) \tag{15}$$

and then Eq.4 must be written as follows:

$$h_i = g\left(\Psi \otimes W_i + b_i\right) \tag{16}$$

Therefore, the cross correlation in the frequency domain given by Eq.5 does not represent Eq.4. This is because the operation of cross correlation is not commutative ($W\otimes\Psi \neq \Psi\otimes W$). As a result, Eq.4 does not give the same correct results as conventional neural networks. This error leads other researchers (El-Bakry, 2004) who consider the references (Ben Yakoub, 1997; Fasel, 1998 and Ben Yakoub *et al*, 1999) to think about how to modify the operation of cross correlation so that Eq.4 can give the same correct results as conventional neural networks. Therefore, errors in these equations must be cleared to all the researchers. El-Bakry (2003) proved that a symmetry condition must be found in input matrices (images and the weights of neural networks) so that fast neural networks can give the same results as conventional neural networks. In case of symmetry $W\otimes\Psi=\Psi\otimes W$, the cross correlation becomes commutative and this is a valuable achievement. In this case, the cross correlation is performed without any constrains on the arrangement of matrices. As presented by El-Bakry (2004), this symmetry condition is useful for reducing the number of patterns that neural networks will learn. This is because the image is converted into symmetric shape by rotating it down and then the up image and its rotated down version are tested together as one (symmetric) image. If a pattern is detected in the rotated down image, then, this means that this pattern is found at the relative position in the up image. So, if conventional neural networks are trained for up and rotated down examples of the pattern, fast neural networks will be trained only to up examples. As the number of trained examples is reduced, the number of neurons in the hidden layer will be reduced and the neural network will be faster in the test phase compared with conventional neural networks.

8. Moreover, Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999) stated that the activity of each neuron in the hidden layer (Eq.4) can be expressed in terms of convolution between a bank of filter (weights) and the input image. This is not correct because the activity of the hidden neuron is a cross correlation between the input image and the weight matrix. It is known that the result of cross correlation between any two functions is different from their convolution. As proved by El-Bakry (2003), the two results will be the same, only when the two matrices are symmetric or at least the weight matrix is symmetric.

9. Images are tested for the presence of a face (object) at different scales by building a pyramid of the input image which generates a set of images at different resolutions. The face detector is then applied at each resolution and this process takes much more time as the number of processing steps will be increased. Ben Yakoub (1997), Fasel (1998) and Ben Yakoub *et al* (1999) stated that the Fourier transforms of the new scales do not need to be computed. This is due to a property of the Fourier transform. If z(x,y) is the original and a(x,y) is the subsampled by a factor of 2 in each direction image then:

$$a(x,y) = z(2x,2y) \tag{17}$$

$$Z(u,v) = FT(z(x,y)) \tag{18}$$

$$FT(a(x,y)) = A(u,v) = \frac{1}{4} Z\left(\frac{u}{2}, \frac{v}{2}\right) \tag{19}$$

This implies that we do not need to recompute the Fourier transform of the subsampled images, as it can be directly obtained from the original Fourier transform. But experimental results have shown that Eq.17 is valid only for images in the following form:

$$\Psi = \begin{bmatrix} A\ A\ B\ B\ C\ C..... \\ A\ A\ B\ B\ C\ C..... \\ . \\ . \\ . \\ . \\ S\ S\ X\ X\ Y\ Y..... \\ S\ S\ X\ X\ Y\ Y..... \end{bmatrix} \tag{20}$$

Ben Yakoub (1997) claimed that the processing needs $O((q+2)N^2\log^2 N)$ additional number of computation steps. Thus the speed up ratio will be (Ben Yakoub, 1997):

$$\eta = \frac{qn^2}{(q+2)\log^2 N} \tag{21}$$

Of course this is not correct, because the inverse of the Fourier transform is required to be computed at each neuron in the hidden layer (for the resulted matrix from the dot product between the Fourier matrix in two dimensions of the input image and the Fourier matrix in two dimensions of the weights, the inverse of the Fourier transform must be computed). So, the term (q+2) in Eq.21 should be (2q+1) because the inverse 2D-FFT in two dimensions must be done at each neuron in the hidden layer. In this case, the number of computation steps required to perform 2D-FFT for the fast neural networks will be:

$$\varphi=(2q+1)(5N^2\log_2 N^2)+(2q)5(N/2)^2\log_2(N/2)^2 \tag{22}$$

In addition, a number of computation steps equal to $6q(N/2)^2+q((N/2)^2-n^2)+q(N/2)^2$ must be added to the number of computation steps required by the fast neural networks.

3. A NEW FASTER ALGORITHM FOR PATTERN DETECTION BASED ON IMAGE DECOMPOSITION

In this section, a new faster algorithm for pattern detection is presented. The number of computation steps required for fast neural networks with different image sizes is listed in Table 5. From this table, we may notice that as the image size is increased, the number of computation steps required by fast neural networks is much increased. For example, the number of computation steps required for an image of size (50x50 pixels) is much less than that needed for an image of size (100x100 pixels). Also, the number of computation steps required for an image of size (500x500 pixels) is much less than that needed for an image of size (1000x1000 pixels). As a result, for example, if an image of size (100x100 pixels) is decomposed into 4 sub-images of size (50x50 pixels) and each sub-image is tested separately, then a speed up factor for pattern detection can be achieved. The number of computation steps required by fast neural networks to test an image after decomposition can be calculated as follows:

1. Assume that the size of the image under test is (NxN pixels).
2. Such image is decomposed into α (LxL pixels) sub-images. So, α can be computed as:

$$\alpha=(N/L)^2 \quad (23)$$

Image size	No. of computation steps in case of using FNN	Image size	No. of computation steps in case of using FNN
25x25	1.9085e+006	1050x1050	7.0142e+009
50x50	9.1949e+006	1100x1100	7.7476e+009
100x100	4.2916e+007	1150x1150	8.5197e+009
150x150	1.0460e+008	1200x1200	9.3306e+009
200x200	1.9610e+008	1250x1250	1.0180e+010
250x250	3.1868e+008	1300x1300	1.1070e+010
300x300	4.7335e+008	1350x1350	1.1998e+010
350x350	6.6091e+008	1400x1400	1.2966e+010
400x400	8.8203e+008	1450x1450	1.3973e+010
450x450	1.1373e+009	1500x1500	1.5021e+010
500x500	1.4273e+009	1550x1550	1.6108e+010
550x550	1.7524e+009	1600x1600	1.7236e+010
600x600	2.1130e+009	1650x1650	1.8404e+010
650x650	2.5096e+009	1700x1700	1.9612e+010
700x700	2.9426e+009	1750x1750	2.0861e+010
750x750	3.4121e+009	1800x1800	2.2150e+010
800x800	3.9186e+009	1850x1850	2.3480e+010
850x850	4.4622e+009	1900x1900	2.4851e+010
900x900	5.0434e+009	1950x1950	2.6263e+010
950x950	5.6623e+009	2000x2000	2.7716e+010
1000x1000	6.3191e+009	2050x2050	2.9211e+010

Table 5: The number of computation steps required by fast neural networks (FNN) for images of sizes (25x25 – 1050x1050 pixels), q=30, n=20.

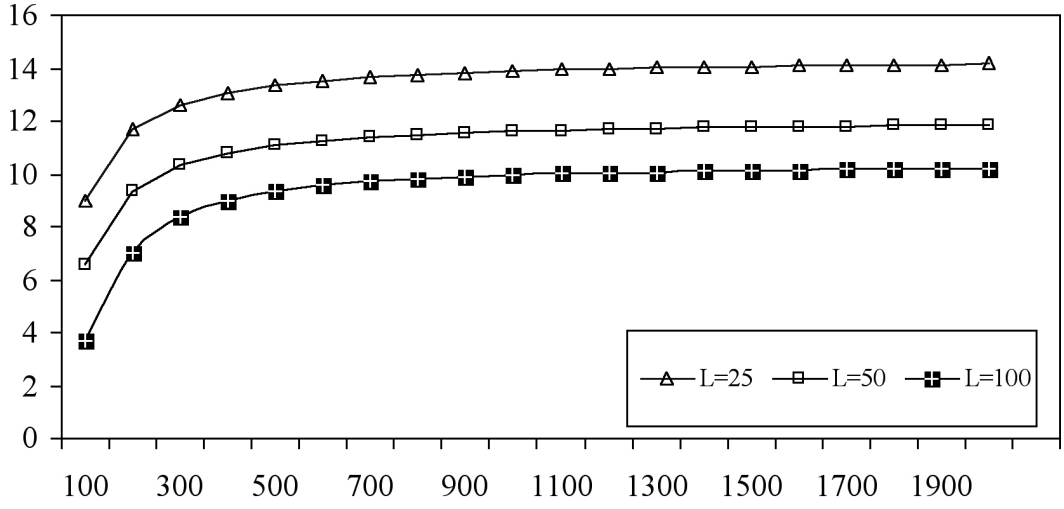


Figure 1: The speed up ratio for images decomposed into different in size sub-images (L).

3. Assume that, the number of computation steps required for testing one (LxL pixels) sub-image is β. So, the total number of computation steps (T) required for testing these sub-images resulting after the decomposition process is:

$$T = \alpha \beta \tag{24}$$

The speed up ratio in this case (η_d) can be computed as follows:

$$\eta_d = \frac{q(2n^2 - 1)(N - n + 1)^2}{(q(\alpha + 1) + \alpha)(5N_s^2 \log_2 N_s^2) + \alpha q(8N_s^2 - n^2) + N_s + \Delta} \tag{25}$$

where,

Ns: is the size of each small sub-image.

Δ: is a small number of computation steps required to obtain the results at the boundaries between sub-images and depends on the size of the sub-image.

To detect a pattern of size 20x20 pixels in an image of any size by using fast neural networks after image decomposition into sub-images, the optimal size of these sub-images must be computed. From Table 5, we may conclude that, the most suitable size for the sub-image which requires the smallest number of computation steps is 25x25 pixels. Also, the fastest speed up ratio can be achieved using this sub-image size (25x25) as shown in Figure 1. It is clear that the speed up ratio is reduced when the size of the sub-image (L) is increased. A comparison between the speed up ratio for fast neural networks and fast neural networks after image decomposition with different sizes of the tested images is listed in Tables 6 and 7. We may notice that the speed up ratio is increased with

Image size	Speed up ratio in case of using (FNN)	Speed up ratio in case of using FNN after image decomposition
50x50	2.7568	4.5871
100x100	5.0439	8.9997
150x150	5.6873	10.7600
200x200	5.9190	11.6707
250x250	6.0055	12.2228
300x300	6.0301	12.5923
350x350	6.0254	12.8565
400x400	6.0059	13.0547
450x450	5.9790	13.2088
500x500	5.9483	13.3320
550x550	5.9160	13.4328
600x600	5.8833	13.5168
650x650	5.8509	13.5878
700x700	5.8191	13.6487
750x750	5.7881	13.7014
800x800	5.7581	13.7475
850x850	5.7292	13.7881
900x900	5.7013	13.8243
950x950	5.6744	13.8566
1000x1000	5.6484	13.8857

Table 6: The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (25x25 pixels) for images of different sizes (from N=50 to N=1000, n=25, q=30).

the size of the input image when using fast neural networks and image decomposition. This is in contrast to using only fast neural networks and this is an important achievement. Moreover, as shown in Figure 2, the speed up ratio in case of image decomposition and different window size (n). It is clear that the speed up ratio is increased with the size of the weight/window matrix (n).

Image size	Speed up ratio in case of using (FNN)	Speed up ratio in case of using FNN after image decomposition
1050x1050	5.6234	13.9120
1100x1100	5.5994	13.9359
1150x1150	5.5762	13.9577
1200x1200	5.5538	13.9777
1250x1250	5.5322	13.9961
1300x1300	5.5113	14.0131
1350x1350	5.4912	14.0288
1400x1400	5.4717	14.0434
1450x1450	5.4528	14.0570
1500x1500	5.4345	14.0696
1550x1550	5.4168	14.0815
1600x1600	5.3996	14.0926
1650x1650	5.3830	14.1030
1700x1700	5.3668	14.1129
1750x1750	5.3511	14.1221
1800x1800	5.3358	14.1309
1850x1850	5.3209	14.1391
1900x1900	5.3064	14.1470
1950x1950	5.2923	14.1544
2000x2000	5.2786	14.1615

Table 7: The speed up ratio in case of using FNN and FNN after image decomposition into sub-images (25x25pixels) for images of different sizes (from N=1050 to N=2000, n=25, q=30).

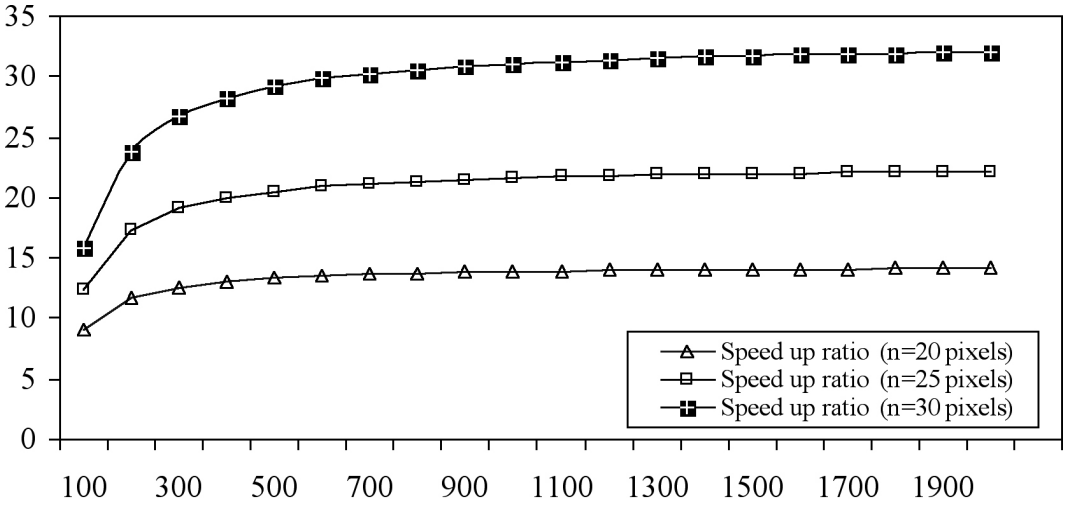


Figure 2: The speed up ratio in case of image decomposition and different window size (n), (L=25x25).

To further reduce the running time as well as increase the speed up ratio of the detection process, a parallel processing technique is used. Each sub-image is tested using a fast neural network simulated on a single processor or a separated node in a clustered system. The number of operations (ω) performed by each processor / node (sub-images tested by one processor/node) =

$$\omega = \frac{\text{The total number of sub} - \text{images}}{\text{Number of Processors / nodes}} \tag{26}$$

$$\omega = \frac{\alpha}{\Pr} \tag{27}$$

where, Pr is the number of processors or nodes.

The total number of computation steps (γ) required to test an image by using this approach can be calculated as:

$$\gamma = \omega b \tag{28}$$

By using this algorithm, the speed up ratio in this case (η_{dp}) can be computed as follows:

$$\eta_{dp} = \frac{q(2n^2 - 1)(N - n + 1)^2}{\text{ceil}(((q(\alpha + 1) + \alpha)(5N_s^2 \log_2 N_s^2) + \alpha q(8N_s^2 - n^2) + N_s)/pr)} \tag{29}$$

ceil(x) is a MATLAB function round the elements of x to the nearest integers towards infinity. As shown in Tables 8 and 9, using a symmetric multiprocessing system with 16 parallel processors or 16 nodes in either a massively parallel processing system or a clustered system, the speed up ratio (with respect to conventional neural networks) for pattern detection is increased. A further reduction in the computation steps can be obtained by dividing each sub-image into groups. For each group, the neural operation (multiplication by weights and summation) is performed for each group by using a single processor. This operation is done for all of these groups as well as other groups in all of the sub-images at the same time. The best case is achieved when each group consists of only one element. In this case, one operation is needed for multiplication of the one element by its weight and also a small number of operations (ε) is required to obtain the overall summation for each sub-image. If the sub-image has n^2 elements, then the required number of processors will be n^2. As a result, the number of computation steps will be $\alpha q(1+\varepsilon)$, where ε is a small number depending on the value of n. For example, when n=20, then ε=6 and if n=25, then ε=7. The speed up ratio can be calculated as:

$$\eta = (2n^2-1)(N-n+1)^2/\alpha(1+\varepsilon) \tag{30}$$

Moreover, if the number of processors = αn^2, then the number of computation steps will be $q(1+\varepsilon)$, and the speed up ratio becomes:

$$\eta = (2n^2-1)(N-n+1)^2/(1+\varepsilon) \tag{31}$$

Furthermore, if the number of processors = $q\alpha n^2$, then the number of computation steps will be $(1+\varepsilon)$, and the speed up ratio can be calculated as:

$$\eta = q(2n^2-1)(N-n+1)^2/(1+\varepsilon) \tag{32}$$

In this case, as the length of each group is very small, there is no need to apply cross correlation between the input image and the weights of the neural network in frequency domain.

Image size	Speed up ratio
50x50	73.3934
100x100	143.9953
150x150	172.1592
200x200	186.7311
250x250	195.5652
300x300	201.4760
350x350	205.7032
400x400	208.8745
450x450	211.3405
500x500	213.3126
550x550	214.9255
600x600	216.2689
650x650	217.4051
700x700	218.3786
750x750	219.2219
800x800	219.9596
850x850	220.6102
900x900	221.1883
950x950	221.7054
1000x1000	222.1707

Table 8: The speed up ratio in case of using FNN after image decomposition into sub-images (25x25 pixels) for images of different sizes (from N=50 to N=1000, n=25, q=30) using 16 parallel processors or 16 nodes.

Image size	Speed up ratio
1050x1050	222.5916
1100x1100	222.9740
1150x1150	223.3232
1200x1200	223.6432
1250x1250	223.9375
1300x1300	224.2091
1350x1350	224.4606
1400x1400	224.6941
1450x1450	224.9114
1500x1500	225.1142
1550x1550	225.3039
1600x1600	225.4817
1650x1650	225.6487
1700x1700	225.8059
1750x1750	225.9541
1800x1800	226.0940
1850x1850	226.2263
1900x1900	226.3517
1950x1950	226.4706
2000x2000	226.5836

Table 9: The speed up ratio in case of using FNN after image decomposition into sub-images (25x25 pixels) for images of different sizes (from N=1050 to N=2000, n=25, q=30) using 16 parallel processors or 16 nodes.

4. SUBIMAGE CENTRING AND NORMALIZATION IN THE FREQUENCY DOMAIN

Feraud *et al* (2000) stated that image normalization to avoid weak or strong illumination could not be done in the frequency space. This is because the image normalization is local and not easily computed in the Fourier space of the whole image. Here, a simple method for image normalization is presented. Centring and normalizing the image can be obtained by centring and normalizing the weights as follows (El-Bakry, 2002):

Let $\overline{X}_{rc}$ be the zero-mean centred sub-image located at (r,c) in the input image ψ:

$$\overline{X}_{rc} = X_{rc} - \bar{x}_{rc} \tag{33}$$

where, $\overline{X}_{rc}$ is the mean value of the sub-image located at (r,c). We are interested in computing the dot product between the sub-image $\overline{X}_{rc}$ and the weights W_i that is:

(34)

Where,

$$\bar{x}_{rc} = \frac{\sum_{i,j=1}^{n} X_{rc}(i,j)}{n} \tag{35}$$

Combining (34) and (35), we get the following expression:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet W_i - \frac{\sum_{k,j=1}^{n} X_{rc}(k,j)}{n^2} \bullet W_i \tag{36}$$

For any two matrices with the same size, multiplying the first matrix dot by the mean of the second and summing the results is the same as multiplying the second matrix dot by the mean of the first one and summing the results of multiplication. Therefore, Eq. (36) can be written as:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet W_i - X_{rc} \bullet \frac{\sum_{k,j=1}^{n} W_i(k,j)}{n^2} \tag{37}$$

The zero mean weights are given by:

$$\overline{W}_i = W_i - \frac{\sum_{k,j=1}^{n} W_i(k,j)}{n^2} \tag{38}$$

Also, Eq. (37) can be written as:

$$\overline{X}_{rc} \bullet W_i = X_{rc} \bullet \left(W_i - \frac{\sum_{k,j=1}^{n} W_i(k,j)}{n^2} \right) \tag{39}$$

So, we may conclude that:

(40)

which means that multiplying a normalized sub-image with a non-normalized weight matrix dot multiplication is equal to the dot multiplication between the non-normalized sub-image and the normalized weight matrix.

5. EFFECT OF WEIGHT NORMALIZATION ON THE SPEED UP RATIO

Normalization of sub-images in the spatial domain (in case of using traditional neural networks) requires $2n^2(N-n+1)^2$ computation steps. On the other hand, normalization of sub-images in the frequency domain through normalizing the weights of the neural networks requires $2qn^2$ operations. This proves that local image normalization in the frequency domain is faster than that

3. Normalization of weights can be done off line. So, the speed up ratio in the case of weight normalization can be calculated as follows:

a) For Conventional Neural Networks:

The speed up ratio equals the number of computation steps required by conventional neural networks with image normalization divided by the number of computation steps needed by conventional neural networks with weight normalization, which is done off line. The speed up ratio η_o in this case can be given by:

$$\eta_c = \frac{q(2n^2 - 1)(N - n + 1)^2 + 2n^2(N - n + 1)^2}{q(2n^2 - 1)(N - n + 1)^2} \tag{42}$$

which can be simplified to:

$$\eta_c = 1 + \frac{2n^2}{q(2n^2 - 1)} \tag{43}$$

b) For Fast Neural Networks:

The overall speed up ratio equals the number of computation steps required by conventional neural networks with image normalization divided by the number of computation steps needed by fast neural networks with weight normalization, which is done off line. The overall speed up ratio η_o can be given by:

$$\eta_o = \frac{q(2n^2 - 1)(N - n + 1)^2 + 2n^2(N - n + 1)^2}{(2q + 1)(5N^2\log_2 N^2) + q(8N^2 - n^2) + N)} \tag{44}$$

which can be simplified to:

$$\eta_o = \frac{(N - n + 1)^2(q(2n^2 - 1) + 2n^2)}{(2q + 1)(5N^2\log_2 N^2) + q(8N^2 - n^2) + N} \tag{45}$$

The relation between the speed up ratio before (η) and after (η_o) the normalization process can be summed up as:

$$\eta_o = \eta + \frac{2n^2(N - n + 1)^2}{(2q + 1)(5N^2\log_2 N^2) + q(8N^2 - n^2) + N} \tag{46}$$

The overall speed up ratio (Eq.45) with images of different sizes and different sizes of windows is listed in Table 11. We can easily note that the speed up ratio in case of image normalization through weight normalization is larger than the speed up ratio (without normalization) listed in Table 1. This means that the search process with normalized fast neural networks is done faster than conventional neural networks with or without normalization of the input image. The overall practical speed up ratio (Eq.45) after normalization of weights off line is listed in Table 12.

6. CONCLUSION

Normalized neural networks for fast pattern detection in a given image have been presented. It has been proved mathematically and practically that the speed of the detection process becomes faster than conventional neural networks. This has been accomplished by applying cross correlation in the frequency domain between the input image and the normalized input weights of the neural networks. New general formulas for fast cross correlation as well as the speed up ratio have been

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	3.7869	5.2121	6.5532
200x200	4.1382	6.1165	8.3167
300x300	4.1320	6.2313	8.6531
400x400	4.0766	6.2063	8.7031
500x500	4.0152	6.1467	8.6684
600x600	3.9570	6.0796	8.6054
700x700	3.9039	6.0132	8.5334
800x800	3.8557	5.9502	8.4603
900x900	3.8120	5.8915	8.3891
1000x1000	3.7723	5.8369	8.3212
1100x1100	3.7360	5.7862	8.2568
1200x1200	3.7027	5.7391	8.1961
1300x1300	3.6719	5.6952	8.1389
1400x1400	3.6434	5.6542	8.0849
1500x1500	3.6169	5.6158	8.0340
1600x1600	3.5922	5.5798	7.9858
1700x1700	3.5690	5.5458	7.9403
1800x1800	3.5472	5.5138	7.8971
1900x1900	3.5266	5.4835	7.8560
2000x2000	3.5072	5.4547	7.8169

Table 11: Theoretical results for the speed up ratio in case of image normalization by normalizing the input weights.

Image size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	74.10	83.22	93.46
200x200	87.60	117.44	143.86
300x300	107.14	111.68	140.84
400x400	111.22	123.86	150.58
500x500	106.28	112.02	165.44
600x600	102.26	128.48	157.2
700x700	117.6	120.48	147.32
800x800	105.98	111.66	115.54
900x900	120.52	124.26	128.62
1000x1000	67.86	73.76	78.78

Table 12: Simulation results for the speed up ratio in case of image normalization by normalizing the input weights.

given. Also, commutative cross correlation has been achieved. A faster neural network approach foe pattern detection has been introduced. Such approach has decomposed the input image under test into many small in size sub-images. Furthermore, a simple algorithm for fast pattern detection based on cross correlations in the frequency domain between the sub-images and the weights of the neural net has been presented in order to speed up the execution time. Simulation results have shown that, using a parallel processing technique, large values of speed up ratio could be achieved. Moreover, by using fast neural networks and image decomposition, the speed up ratio has been increased with

the size of the input image. Also, the problem of local sub-image normalization in the frequency space has been solved. It has been generally proved that the speed up ratio in the case of image normalization through normalization of weights is faster than sub-image normalization in the spatial domain. This speed up ratio is faster than the one obtained without normalization. Simulation results have confirmed theoretical computations by using Matlab. The proposed approach can be applied to detect the presence/absence of any other object in an image.

REFERENCES

BALUJA, S., ROWLEY, H. A. and KANADE, T. (1998): Neural network–based face detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20, (1): 23–38.

BEN-YACOUB, S. (1997): Fast object detection using MLP and FFT, *IDIAP-RR 11, IDIAP*.

BEN-YACOUB, S., FASEL, B. and LUETTIN, J. (1999): Fast face detection using MLP and FFT, in *Proc. Second International Conference on Audio and Video-based Biometric Person Authentication* (AVBPA'99).

EL-BAKRY, H. M., ABO-ELSOUD, M. A. and KAMEL, M. S. (2000): Fast modular neural networks for human face detection, *Proc. of IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Como, Italy, III: 320–324, 24-27 July.

EL-BAKRY, H. M. (2000): Fast iris detection using cooperative modular neural nets, *Proc. of the 6th International Conference on Soft Computing*, 1–4 Oct., Japan.

EL-BAKRY, H. M. (2001): Automatic human face recognition using modular neural networks, *Machine Graphics & Vision Journal* (MG&V) 10 (1): 47–73.

EL-BAKRY, H. M. (2001): Fast iris detection using cooperative modular neural networks, *Proc. of the 5th International Conference on Artificial Neural Nets and Genetic Algorithms*, 22–25 April, Sydney, Czech Republic, 201–204.

EL-BAKRY, H. M. (2001): Fast iris detection using neural nets, *Proc. of the 14th Canadian Conference on Electrical and Computer Engineering*, 13–16 May, Canada: 1409–1415.

EL-BAKRY, H. M. (2001): Human iris detection using fast cooperative modular neural nets, *Proc. of INNS-IEEE International Joint Conference on Neural Networks*, 14–19 July, Washington, DC, US : 577–582.

EL-BAKRY, H. M. (2001): Human iris detection for information security using fast neural nets, *Proc. of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, 22–25 July, Orlando, Florida, USA.

EL-BAKRY, H. M. (2001): Human iris detection for personal identification using fast modular neural nets, *Proc. of the 2001 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, 25–28 July, Monte Carlo Resort, Las Vegas, Nevada, USA: 112–118.

EL-BAKRY, H. M. (2001): Human face detection using fast neural networks and image decomposition, *Proc. the fifth International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, 6–8 September, Osaka-kyoiku University, Kashiwara City, Japan: 1330–1334.

EL-BAKRY, H. M. (2001): Fast iris detection for personal verification using modular neural networks, *Proc. of the International Conference on Computational Intelligence*, 1–3 Oct., Dortmund, Germany: 269–283.

EL-BAKRY, H. M. (2001): Fast cooperative modular neural nets for human face detection, *Proc. of IEEE International Conference on Image Processing*, 7–10 Oct., Thessaloniki, Greece.

EL-BAKRY, H. M. (2001): Fast face detection using neural networks and image decomposition, *Proc. of the 6th International Computer Science Conference, Active Media Technology*, Dec. 18–20, Hong Kong – China: 205–215.

EL-BAKRY, H. M. (2002): Face detection using fast neural networks and image decomposition, *Neurocomputing Journal*, 48: 1039–1046.

EL-BAKRY, H. M. (2002): Face detection using fast neural networks and image decomposition, *Proc. of INNS-IEEE International Joint Conference on Neural Networks*, 14–19 May, Honolulu, Hawaii, USA.

EL-BAKRY, H. M. (2002): Human iris detection using fast cooperative modular neural networks and image decomposition, *Machine Graphics & Vision Journal* (MG&V), 11 (4): 498–512.

EL-BAKRY, H. M. (2003): Comments on using MLP and FFT for fast object/face detection, *Proc. of IEEE IJCNN'03*, Portland, Oregon, 20–24 July: 1284–1288,

EL-BAKRY, H. M. and STOYAN, H. (2004): Fast neural networks for object/face detection, *Proc. of the 30th Anniversary SOFSEM Conference on Current Trends in Theory and Practice of Computer Science*, 24–30 January, Hotel VZ MERIN, Czech Republic.

EL-BAKRY, H. M. and STOYAN, H. (2004): Fast neural networks for sub-matrix (object/face) detection, *Proc. of IEEE International Symposium on Circuits and Systems*, Vancouver, Canada, 23–26 May.

EL-BAKRY, H. M. (2004): Fast sub-image detection using neural networks and cross correlation in frequency domain, *Proc. of IS 2004: 14th Annual Canadian Conference on Intelligent Systems*, Ottawa, Ontario, 6–8 June.

EL-BAKRY, H. M. and STOYAN, H. (2004): Fast neural networks for code detection in a stream of sequential data, *Proc. of CIC 2004 International Conference on Communications in Computing*, Las Vegas, Nevada, USA, 21–24 June.

EL-BAKRY, H. M. (2004): Fast neural networks for object/face detection, *Proc. of 5th International Symposium on Soft Computing for Industry with Applications of Financial Engineering*, 28 June–4 July, Sevilla, Andalucia, Spain.

EL-BAKRY, H. M. and STOYAN, H. (2004): A fast searching algorithm for sub-image (object/face) detection using Neural networks, *Proc. of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, 18–21 July, Orlando, USA.

EL-BAKRY, H. M. and STOYAN, H. (2004): Fast neural networks for code detection in sequential data using neural networks for communication applications, *Proc. of the First International Conference on Cybernetics and Information Technologies, Systems and Applications*: CITSA 2004, 21–25 July, Orlando, Florida, USA, 4:150–153.

EL-BAKRY, H. M. and ZHAO, Q. (2005) : Speeding-up normalized neural networks for face/object detection, *Machine Graphics & Vision Journal* (MG&V) 14 (1): 29–59.

EL-BAKRY, H. M. and ZHAO, Q. (2005): Fast pattern detection using normalized neural networks and cross correlation in the frequency domain, *EURASIP Journal on Applied Signal Processing* (13): 2054–2060 .

FASEL, B. (1998): Fast multi-scale face detection, *IDIAP-Com 98-04*.

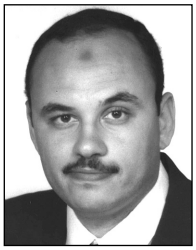
FERAUD, R., BERNIER, O., VIALLET, J. E. and COLLOBERT, M. (2000): A fast and accurate face detector for indexation of face images, *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, 28–30 March.

SRISUK, S. and KURUTACH, W. (2002): A new robust face detection in color images, *Proc. of IEEE Computer Society International Conference on Automatic Face and Gesture Recognition*, Washington D.C., USA, May 20–21, 2002: 306–311.

ZHU, Y., SCHWARTZ, S. and ORCHARD, M. (2000): Fast face detection using subspace discriminate wavelet features, *Proc. of IEEE Computer Society International Conference on Computer Vision and Pattern Recognition (CVPR'00)*, South Carolina, 13–15 June, 1: 1636–1643.

BIOGRAPHICAL NOTES

Hazem Mokhtar El-Bakry received his BSc degree in electronics engineering, and MSc in electrical communication engineering from the Faculty of Engineering, Mansoura University, Egypt, in 1992 and 1995 respectively. Since 1997, he has been an assistant lecturer at the Faculty of Computer Science and Information Systems, Mansoura University, Egypt. Currently, he is a doctoral student at the multimedia device laboratory, University of Aizu, Japan. In 2004, he got a research scholarship from the Japanese Government based on a recommendation from University of Aizu.



Hazem Mokhtar El-Bakry

His research interests include neural networks, pattern recognition, image processing, biometrics, cooperative intelligent systems and electronic circuits. In these areas, he has published more than 39 papers as a single author in major international journals and conferences. He is the first author in 12 refereed international journal papers and more than 70 refereed international conference papers.

Hazem El-Bakry has the patent No. 2003E 19442 DE HOL / NUR, Magnetic Resonance, SIEMENS Company, Erlangen, Germany, 2003. He is a referee for IEEE Transactions on Signal Processing, the International Journal of Machine Graphics & Vision, and many IEEE international conferences.

Dr Qiangfu Zhao received the PhD degree from Tohoku University of Japan in 1988. He joined the Department of Electronic Engineering, Beijing Institute of Technology, China in 1988, first as a post doctoral fellow and then associate professor. He was associate professor from October 1993 at the Department of Electronic Engineering, Tohoku University, Japan. He joined the University of Aizu, Japan from April 1995 as associate professor, and became tenure full professor in April 1999. Professor Zhao's research interests include image processing, pattern recognition and understanding, computational intelligence, neurocomputing and evolutionary computation.

Qiangfu Zhao