EURASIP Journal on Applied Signal Processing 2005:13, 2054–2060
© 2005 Hindawi Publishing Corporation

Fast Pattern Detection Using Normalized Neural Networks and Cross-Correlation in the Frequency Domain

Hazem M. El-Bakry
Multimedia Devices Laboratory, University of Aizu, Aizu Wakamatsu 965-8580, Japan
Email: d8071106@u-aiza.ac.jp

Qiangfu Zhao
Multimedia Devices Laboratory, University of Aizu, Aizu Wakamatsu 965-8580, Japan
Email: qf-zhao@u-aizu.ac.jp

Received 12 January 2004; Revised 20 December 2004

Neural networks have shown good results for detection of a certain pattern in a given image. In our previous work, a fast algorithm for object/face detection was presented. Such algorithm was designed based on cross-correlation in the frequency domain between the input image and the weights of neural networks. Our previous work also solved the problem of local subimage normalization in the frequency domain. In this paper, the effect of image normalization on the speedup ratio of pattern detection is presented. Simulation results show that local subimage normalization through weight normalization is faster than subimage normalization in the spatial domain. Moreover, the overall speedup ratio of the detection process is increased as the normalization of weights is done offline.

Keywords and phrases: fast pattern detection, neural networks, cross-correlation, image normalization.

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 [1, 2, 3]. But the problem with neural networks is that the computational complexity is very high because the networks have to process many small local windows in the images [4, 5]. Some authors tried to speed up the detection process of neural networks [6, 7, 8]. They proposed a multilayer perceptron (MLP) algorithm for fast object/face detection. They claimed that applying cross-correlation in the frequency domain between the input image and the neural weights is much faster than using conventional neural networks. They stated this without any conditions and introduced formulas for the number of computation steps needed by conventional neural networks and their proposed fast neural networks. Then, they established an equation for the speedup ratio. It was proved in [9] that their equations contain many errors which lead to an invalid speedup ratio. Moreover, a symmetry condition is necessary and must be found either in the input image or in the neural weights so that those fast neural networks can give the same correct results as conventional neural networks for detecting a certain pattern in a given image. Recently, we succeeded in accelerating the behavior of neural networks during the search process [10]. The speedup of the detection process is achieved by converting the input image into a symmetric one and applying cross-correlation in the frequency domain between the new symmetric image and the neural weights. Mathematical proof and simulation results for fast testing of the new proposed symmetric image using Matlab are given.

The problem of subimage (local) normalization in the Fourier space was presented in [11]. This problem was solved in [12]. We proved that the number of computation steps required for weight normalization is less than that needed for image normalization. But, we did not discuss the effect of normalization on the speedup ratio. Here, the effect of weight normalization on the speedup 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 speedup ratio. This is because weight normalization requires fewer computation steps than subimage normalization. Moreover, for neural networks, normalization of weights can be easily doneoff line before starting the search process.

In Section 2, fast neural networks for pattern detection are described. A new symmetric form for the input image to speed up the detection process is presented in Section 3. Subimage normalization in the frequency domain through normalization of weights is introduced in Section 4. The effect of weight normalization on the speedup ratio is presented in Section 5.

2. THEORY OF FAST NEURAL NETWORKS BASED ON CROSS-CORRELATION IN THE FREQUENCY DOMAIN FOR PATTERN DETECTION

Finding a certain pattern in the input image is a search problem. Each subimage in the input image is tested for the presence or absence of the required pattern. At each pixel position in the input image, each subimage is multiplied by a window of weights, which has the same size as the subimage. The outputs of neurons in the hidden layer are multiplied by the weights of the output layer. A high output implies that the tested subimage contains the required pattern and vice versa. Thus, we may conclude that this searching problem is a cross-correlation between the image under test and the weights of the hidden neurons.

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 the 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 speedup in an order of magnitude can be achieved during the detection process [1, 2, 9, 10, 12].

In the detection phase, a subimage I of size $m\times n$ (sliding window) is extracted from the tested image, which has a size $P \times T$, and fed to the neural network. Let X_i be the vector of weights between the input subimage and the hidden layer. This vector has a size of $m\times n$ and can be represented as $m\times n$ matrix. The output of hidden neurons h_i can be calculated as follows:

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

where g is the activation function and b_i is the bias of each hidden neuron i. Equation (1) represents the output of each hidden neuron for a particular subimage I. It can be obtained from image Z as follows:

$$h_i(u,v) = g\left(\sum_{j=-m/2}^{m/2}\sum_{k=-n/2}^{n/2} X_i(j,k)Z(u+j,v+k)+b_i\right). \tag{2}$$

Equation (2) represents a cross-correlation operation. Given any two functions f and d, their cross-correlation can be obtained by

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

Therefore, (2) may be written as

$$h_i = g\left(X_i \otimes Z + b_i\right), \tag{4}$$

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) and $(u,v)\in[P-m+1, T-n+1]$.

Now, the above cross-correlation can be expressed in terms of the Fourier transform as follows:

$$Z\otimes X_i = F^{-1}\left(F(Z)\bullet F*(X_i)\right). \tag{5}$$

Hence, by evaluating this cross-correlation, a speedup ratio can be obtained comparable to conventional neural networks. Also, the final output of the neural network can be evaluated as

$$O(u,v) = g\left(\sum_{i=1}^{q} w_o(i)h_i(u,v)+b_o\right), \tag{6}$$

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 is located at the position (u,v) in the input image Z.

The complexity of cross-correlation in the frequency domain can be analyzed as follows [9].

(1) For a tested image of $N\times N$ pixels, the 2D FFT requires a number equal to $O(N^2\log_2 N^2)$ of complex computation steps. Also, the same number of complex computation steps is required for computing the 2D FFT of the weight matrix for each neuron in the hidden layer.

(2) At each neuron in the hidden layer, the inverse 2D FFT is computed. So, q backward and $(1+q)$ forward transforms have to be computed. Therefore, for an image under test, the total number of the 2D FFT to compute is $O((2q+1)N^2\log_2 N^2)$.

(3) The input image and the weights should be multiplied in the frequency domain. Therefore, a number of complex computation steps equal to $O(qN^2)$ should be added.

(4) The number of computation steps required by fast neural networks is complex and must be converted into a real version. It is known that the 2D fast Fourier transform requires $O((N^2/2)\log_2 N^2)$ complex multiplications and $O(N^2\log_2 N^2)$ 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. So, the total number of computation steps required to obtain the 2D FFT of an $N\times N$ image is [9]

$$\rho = O\left(6\left(\left(\frac{N^2}{2}\right)\log_2 N^2\right)+2\left(N^2\log_2 N^2\right)\right) \tag{7}$$

which may be simplified to

$$\rho = O\left(5N^2\log_2 N^2\right). \tag{8}$$

Performing complex dot product in the frequency domain also requires $O(6qN^2)$ real operations.

(5) In order to perform cross-correlation in the frequency domain, the weight matrix must have the same size as the input image. So, a number of zeros equal to $(N^2 - n^2)$ must be added to the weight matrix. This requires a total real number of computation steps equal to $O(q(N^2 - n^2))$ for all neurons. Moreover, after computing the FFT2 for the weight matrix, the conjugate of this matrix must be obtained. So, a real number of computation steps equal to $O(qN^2)$ 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 $O(N)$ are 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 FFT2. To create a complex number requires two real floating-point operations. So, the total number of computation steps required for fast neural networks becomes [9]

$$\sigma = O\big((2q+1)(5N^2 \log_2 N^2)+6qN^2+q(N^2-n^2)+qN^2+N\big) \tag{9}$$

which can be reformulated as

$$\sigma = O\big((2q+1)(5N^2 \log_2 N^2) + q(8N^2 - n^2) + N\big). \tag{10}$$

(6) Using a sliding window of size $n \times n$ for the same image of $N \times N$ pixels, $O(q(2n^2-1)(N-n+1)^2)$ computation steps are required when using traditional neural networks for pattern detection process. The theoretical speedup factor η can be evaluated as follows [9]:

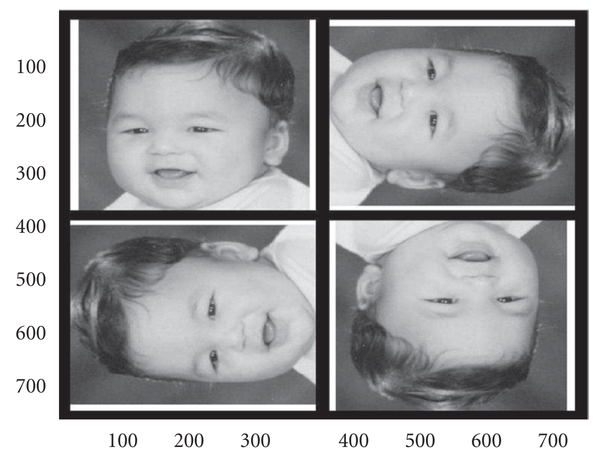
$$\eta = O\left(\frac{q(2n^2 - 1)(N - n + 1)^2}{(2q + 1)(5N^2 \log_2 N^2) + q(8N^2 - n^2) + N}\right). \tag{11}$$

(7) But as proved in [10], this cross-correlation in frequency domain (fast neural networks) gives the same results as conventional cross-correlation (conventional neural networks) only in two cases. Either the weights are symmetric or the input image is symmetric. It is very complex to allow the weights to be symmetric in the required form which needs to be as follows [9]:

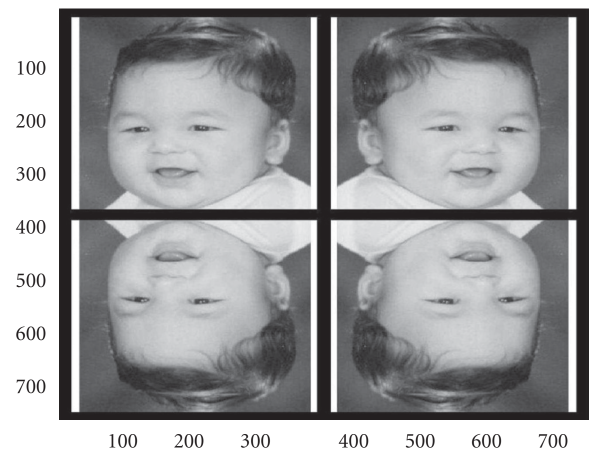
$$W = \begin{bmatrix} w & 0 \\ 0 & w_d \end{bmatrix}. \tag{12}$$

Adding this constraint to the learning rules will cause many well-known problems during the training process of the neural network. Another solution is to convert the input image into one of the required symmetric forms as shown in Figure 1. As the input image has a dimension of (N), the new symmetric image will have a length of $(2N)$. In this case, the number of computation steps required for fast neural networks can be calculated as follows [9]:

$$\sigma_{2N} = O\big((2q+1)(5(2N)^2 \log_2 (2N)^2) + q(8(2N)^2 - n^2) + 2N\big). \tag{13}$$



(a)



(b)

Figure 1: Image conversion from nonsymmetric to symmetric one.

But converting the nonsymmetric input image into a symmetric one will slow down the proposed fast neural networks more compared to conventional neural networks. In this case, for any size of the input image, dividing the number of operations required for conventional neural networks by those needed by fast neural networks (11) gives a lower speedup ratio than the one listed in Table 1 [9].

3. A NEW SYMMETRIC FORM FOR THE INPUT IMAGE TO SPEED UP THE PROCESS OF NEURAL NETWORKS

In this section, another new symmetric form for the input image is presented. This form helps in reducing the number of computation steps required by neural networks for pattern detection. As shown in Figure 2, the input image will be converted into symmetric form by rotating it 180 degrees. Then, both the up and down images are tested as a single (symmetric) image consisting of two images. In this case, the new symmetric image will have $(2N \times N)$ dimensions.

TABLE 1: A comparison between the number of computation steps (in millions) required for conventional and fast neural networks to manipulate images shown in Figure 1 with different sizes ($n = 20$).

Image size	Conventional neural networks	Fast neural networks ($2N$)	Speedup ratio
100 × 100	157,267170	196,098091	0.802
200 × 200	785,281170	882,028364	0.890
300 × 300	1892,695170	2113,036584	0.896
400 × 400	3479,509170	3918,549456	0.888
500 × 500	5545,723170	6319,116413	0.877
600 × 600	8091,337170	9330,582337	0.867
700 × 700	11116,351170	12965,856005	0.857
800 × 800	14620,765170	17235,856005	0.848
900 × 900	18604,579170	21986,745146	0.846
1000 × 1000	23067,793170	27716,501654	0.832

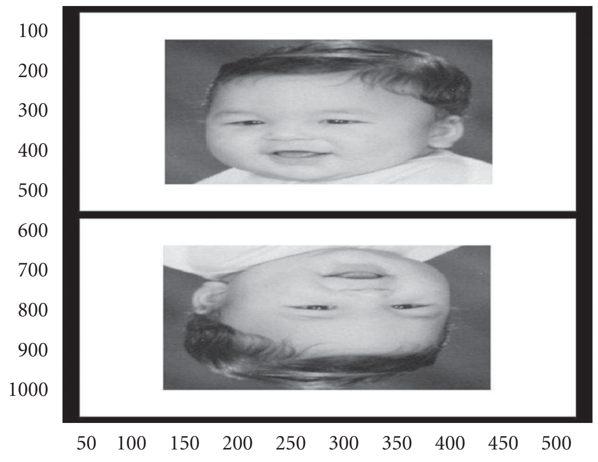


FIGURE 2: Image conversion from nonsymmetric to symmetric one through rotation into down direction.

TABLE 2: The theoretical speedup ratio in case of converting an image into symmetric one through rotation into down direction.

Image size	Speedup ratio ($n = 20$)	Speedup ratio ($n = 25$)	Speedup ratio ($n = 30$)
100 × 100	1.71	2.36	2.96
200 × 200	1.88	2.79	3.79
300 × 300	1.89	2.85	3.96
400 × 400	1.86	2.85	3.99
500 × 500	1.84	2.82	3.98
600 × 600	1.82	2.80	3.96
700 × 700	1.80	2.77	3.93
800 × 800	1.78	2.74	3.90
900 × 900	1.76	2.72	3.87
1000 × 1000	1.74	2.70	3.84

TABLE 3: Simulation results for speedup ratio in case of converting an image into symmetric one through rotation into down direction.

Image size	Speedup ratio ($n = 20$)	Speedup ratio ($n = 25$)	Speedup ratio ($n = 30$)
100 × 100	5.29	6.74	11.16
200 × 200	4.46	6.24	9.94
300 × 300	4.17	5.08	8.66
400 × 400	3.59	4.78	7.45
500 × 500	3.40	4.34	6.87
600 × 600	3.30	4.42	6.16
700 × 700	3.12	4.20	5.74
800 × 800	2.60	3.58	4.76
900 × 900	2.97	4.10	5.38
1000 × 1000	2.57	3.47	4.63

By substituting in (9) for the new dimensions, the number of computation steps required for cross-correlating this new image with the weights in the frequency domain can be calculated as follows [10]:

$$\begin{aligned}\sigma = O\big((2q+1)\big(5\big(2N^2\log_2 N + 2N^2\log_2 2N\big)\big)\\ + q6\big(2N^2\big) + q\big(2N^2 - n^2\big) + q\big(2N^2\big) + 2N\big),\end{aligned} \tag{14}$$

which can be simplified to

$$\sigma = O\big((2q+1)\big(10N^2\big(\log_2 2N + \log_2 N\big)\big) + q\big(16N^2 - n^2\big) + 2N\big). \tag{15}$$

So, the speedup ratio in this case can be calculated as

$$\eta = O\left(\frac{q\big(2n^2-1\big)(N-n+1)^2}{(2q+1)\big(10N^2\big(\log_2 2N + \log_2 N\big)\big) + q\big(16N^2 - n^2\big) + 2N}\right). \tag{16}$$

The theoretical speedup ratio in this case with different sizes of the input image and different in size weight matrices is listed in Table 2. Practical speedup ratio for manipulating images of different sizes and different in size weight matrices is also listed in Table 3 using 700 MHz processor and Matlab.

This new configuration is useful for reducing the number of patterns that the neural network will learn. This is because the image is rotated down as shown in Figure 2. 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 when using the presented configuration for the input image. 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.

4. SUBIMAGE CENTERING AND NORMALIZATION IN THE FREQUENCY DOMAIN

In [11], the authors 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. Centering and normalizing the image can be obtained by centering and normalizing the weights as follows [12].

Let $\bar{X}_{rc}$ be the zero-mean centered subimage located at (r, c) in the input image ψ as follows:

$$\bar{X}_{rc} = X_{rc} - \bar{x}_{rc}, \tag{17}$$

where $\bar{x}_{rc}$ is the mean value of the subimage located at (r, c). We are interested in computing the cross-correlation between the subimage $\bar{X}_{rc}$ and the weights W_i, that is,

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \bar{x}_{rc} \otimes W_i, \tag{18}$$

where

$$\bar{x}_{rc} = \frac{X_{rc}}{n^2}. \tag{19}$$

Combining (18) and (19), we get the following expression:

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - \frac{X_{rc}}{n^2} \otimes W_i, \tag{20}$$

which is the same as

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes W_i - X_{rc} \otimes \frac{W_i}{n^2}. \tag{21}$$

The centered zero-mean weights are given by

$$\overline{W}_i = W_i - \frac{W_i}{n^2}. \tag{22}$$

Also, (21) can be written as

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes \left(W_i - \frac{W_i}{n^2} \right). \tag{23}$$

Table 4: The speedup ratio of the normalization process for images of different sizes ($n = 20, q = 100$).

Image size	Speedup ratio
100 × 100	62
200 × 200	328
300 × 300	790
400 × 400	1452
500 × 500	2314
600 × 600	3376
700 × 700	4638
800 × 800	6100
900 × 900	7762
1000 × 1000	9624

So, we may conclude that

$$\bar{X}_{rc} \otimes W_i = X_{rc} \otimes \overline{W}_i \tag{24}$$

which means that cross-correlating a centered image with the weight matrix is equal to the cross-correlation of the noncentered image with the centered weight matrix.

5. EFFECT OF WEIGHT NORMALIZATION ON THE SPEEDUP RATIO

Normalization of subimages 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 subimages 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 in the spatial one. By using weight normalization, the speedup ratio for image normalization Γ can be calculated as

$$\Gamma = O\left(\frac{(N - n + 1)^2}{q} \right). \tag{25}$$

The speedup ratio of the normalization process for images of different sizes is listed in Table 4. As a result, we may conclude the following.

(1) Using this technique, normalization in the frequency domain can be done through normalizing the weights in spatial domain.

(2) Normalization of an image through normalization of weights is faster than normalization of each subimage.

(3) Normalization of weights can be done offline. So, the speedup ratio in the case of weight normalization can be calculated as follows.

(a) For conventional neural networks

The speedup ratio equals the number of computation steps required by conventional neural networks with image normalization divided by the number of computation steps

Table 5: Simulation results for the speedup ratio in case of image normalization by normalizing the input weights.

Image size	Speedup ratio ($n = 20$)	Speedup ratio ($n = 25$)	Speedup ratio ($n = 30$)
100×100	37.05	41.61	46.73
200×200	43.80	58.72	71.93
300×300	53.57	55.84	70.42
400×400	55.61	61.93	75.29
500×500	53.14	56.01	82.72
600×600	51.13	64.24	78.60
700×700	58.80	60.24	73.66
800×800	52.99	55.83	57.77
900×900	60.26	62.13	64.31
1000×1000	33.93	36.88	39.39

needed by conventional neural networks with weight normalization, which is done offline. The speedup ratio η_c in this case can be given by

$$\eta_c = O\left(\frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{q(2n^2-1)(N-n+1)^2}\right), \tag{26}$$

which can be simplified to

$$\eta_c = O\left(1 + \frac{2n^2}{q(2n^2-1)}\right). \tag{27}$$

(b) For fast neural networks

The overall speedup 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 offline. The overall speedup ratio η_o can be given by

$$\eta_o = O\left(\frac{q(2n^2-1)(N-n+1)^2 + 2n^2(N-n+1)^2}{((2q+1)(10N^2(\log_2 2N + \log_2 N)) + q(16N^2-n^2) + 2N)}\right), \tag{28}$$

which can be simplified to

$$\eta_o = O\left(\frac{(N-n+1)^2(q(2n^2-1) + 2n^2)}{((2q+1)(10N^2(\log_2 2N + \log_2 N)) + q(16N^2-n^2) + 2N)}\right). \tag{29}$$

For the relation between the speedup ratio before (η) and after (η_o), the normalization process can be summed up as

$$\eta_o = O\left(\eta + \frac{2n^2(N-n+1)^2}{((2q+1)(10N^2(\log_2 2N + \log_2 N)) + q(16N^2-n^2) + 2N)}\right). \tag{30}$$

The overall speedup ratio with images of different sizes and different sizes of windows is listed in Table 5. We can easily note that the speedup ratio in case of image normalization through weight normalization is larger than the speedup ratio (without normalization) listed in Table 3. 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.

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 converting the input image into the presented symmetric form and normalizing the weights of the neural networks. Simulation results have confirmed this by using Matlab. Moreover, by using this new symmetric configuration for the input image, the number of neurons in the hidden layer has been reduced. As a result, fast neural networks, based on cross-correlation in the frequency domain, have become faster than conventional neural networks. Furthermore, we have generally proved that the speedup ratio in the case of image normalization through normalization of weights is faster than without normalization.

REFERENCES

[1] H. M. El-Bakry, "Human iris detection using fast cooperative modular neural networks and image decomposition," *Machine Graphics & Vision Journal*, vol. 11, no. 4, pp. 498–512, 2002.

[2] H. M. El-Bakry, "Automatic human face recognition using modular neural networks," *Machine Graphics & Vision Journal*, vol. 10, no. 1, pp. 47–73, 2001.

[3] S. Baluja, H. A. Rowley, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 1, pp. 23–38, 1998.

[4] S. Srisuk and W. Kurutach, "A new robust face detection in color images," in *Proc. 5th IEEE International Conference on Automatic Face and Gesture Recognition (AFGR '02)*, pp. 291–296, Washington, DC, USA, May 2002.

[5] Y. Zhu, S. Schwartz, and M. Orchard, "Fast face detection using subspace discriminate wavelet features," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, vol. 1, pp. 636–642, Hilton Head Island, SC, USA, June 2000.

[6] S. Ben-Yacoub, B. Fasel, and J. Luettin, "Fast face detection using MLP and FFT," in *Proc. 2nd International Conference on Audio and Video-Based Biometric Person Authentication (AVBPA '99)*, pp. 31–36, Washington, DC, USA, March 1999.

[7] B. Fasel, "Fast multi-scale face detection," IDIAP-Com 98-04, IDIAP, Eurecom, Sofia-Antipolis, France, 1998.

[8] S. Ben-Yacoub, "Fast object detection using MLP and FFT," IDIAP-RR 11, IDIAP, Martigny, Switzerland, 1997.

[9] H. M. El-Bakry, "Comments on using MLP and FFT for fast object/face detection," in *Proc. International Joint Conference on Neural Networks (IJCNN '03)*, vol. 2, pp. 1284–1288, Portland, Ore, USA, July 2003.

[10] H. M. El-Bakry and Q. Zhao, "Fast object/face detection using neural networks and fast Fourier transform," *International Journal on Signal Processing*, vol. 1, no. 3, pp. 182–187, 2004.

[11] R. Feraud, O. Bernier, J. E. Viallet, and M. Collobert, "A fast and accurate face detector for indexation of face images," in *Proc. 4th IEEE International Conference on Automatic Face and Gesture Recognition (AFGR '00)*, pp. 77–82, Grenoble, France, March 2000.

[12] H. M. El-Bakry, "Face detection using fast neural networks and image decomposition," *Neurocomputing*, vol. 48, no. 1-4, pp. 1039–1046, 2002.

Hazem M. El-Bakry received the B.S. degree in electronics engineering and the M.S. degree 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 the University of Aizu, Japan. 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 35 papers as a single author in major international journals and conferences. He is the first author in 11 refereed international journal papers and more than 56 refereed international conference papers. Engineer Hazem El-Bakry has the Patent no. 2003E 19442 DE HOL/NUR, Magnetic Resonance, SIEMENS Company, Erlangen, Germany, 2003. He is a referee for the International Journal of Machine Graphics & Vision and many different major international conferences. He was selected as a Chairman for sessions in many different international conferences. He was invited for a talk in the Biometric Consortium, Orlando, Fla, USA, 12–14 September 2001, which was cosponsored by the United States National Security Agency (NSA) and the National Institute of Standards and Technology (NIST).

Qiangfu Zhao received the Ph.D. degree from Tohoku University, Japan, in 1988. He joined the Department of Electronic Engineering, Beijing Institute of Technology, China, in 1988, first as a Postdoctoral Fellow and then as an Associate Professor. He was an Associate Professor in October 1993 at the Department of Electronic Engineering, Tohoku University, Japan. He joined the University of Aizu, Japan, in April 1995 as an Associate Professor, and became a tenure Full Professor in April 1999. His research interests include image processing, pattern recognition and understanding, computational intelligence, neurocomputing, and evolutionary computation.