IEEE-International Conference on Signal processing, Communications and Networking
Madras Institute of Technology, Anna University, Chennai, India, Jan 4–6, 2008. Pp244-249

Effect of Finite Register Length on Bacterial Foraging Optimization based ICA and Constrained Genetic Algorithm based ICA Algorithm

D.P.Acharya[1], G.Panda[2], Senior Member IEEE, Y.V.S.Lakshmi[3]
[1 2] Department of Electronics & Communication Engineering, NIT, Rourkela -769 008.
[2]Center for Development of Telematics, Bangalore – 560 100.
d_p_acharya@rediffmail.com , ganapati.panda@gmail.com, lakshmi@cdotb.ernet.in

***Abstract*-Independent Component Analysis (ICA) technique separates mixed signals blindly without any information of the mixing system. Bacterial Foraging Optimization based ICA (BFOICA) and Constrained Genetic Algorithm based ICA (CGAICA) are two recently developed derivative free evolutionary computational ICA techniques. In BFOICA the foraging behavior of *E.coli* bacteria present in our intestine is mimicked for evaluation of independent components (IC) where as in CGAICA Genetic Algorithm is used for IC estimation in a constrained manner. The present work evaluates the error performance of BFOICA and CGAICA algorithm for its fixed-point implementation. Simulation study is carried on both fixed and floating point ICA algorithms. It is observed that the word length greatly influences the separation performance. A comparison of fixed-point error performance of both the algorithms is also carried out in this work.**

I. INTRODUCTION

Independent Component Analysis (ICA) is a statistical signal processing technique having emerging new practical application areas, such as blind separation of mixed voices or images, analysis of several types of data or feature extraction [1],[2],[3],[4],[5]. Though several algorithms for ICA have been reported in literature [6], [7] very few attempts have been made for successful practical implementation of any of these algorithms [8], [9]. An algorithm can be implemented either by 'fixed point arithmetic' or by 'floating point arithmetic'. The floating point method demands more computational overhead resulting in a large number of processing elements. Thus the system becomes speed limited with large chip area and consuming more power in the process. Therefore a traditional choice has been for implementing systems by using fixed point arithmetic.

In fixed point method of implementation every operation introduces an error (truncation/round up) due to finite register length. These errors propagate and appear as a noise at the output of the system. Such noise alters or degrades the desired system performance. Hence a detailed analysis of error and account for implementation noise introduced by the fixed-point operations becomes the essential before implementation of a system [10]. The analysis also helps to know the accuracy that can be expected out of the existing system and to design with the minimum cost a new system to meet the required accuracy specifications [11]. Such an analysis with fast ICA and algebraic ICA algorithms has been carried out in [12].

Bacterial Foraging based Independent Component Analysis (BFOICA) [13] and Constrained Genetic Algorithm based Independent Component Analysis (CGAICA) [14] are two recently developed derivative free evolutionary computational ICA techniques. Both BFO and GA being population search based optimization techniques, they have several commonalities. However, BFOICA is reported to have faster convergence as compared to CGAICA [13]. Therefore it is quite motivating to study the effect of finite register length implementation of both the algorithms. The present work focuses on the fixed-point performance evaluation of BFOICA algorithm and CGAICA algorithm when implemented with different register lengths.

II. INDEPENDENT COMPONENT ANALYSIS (ICA)

ICA is a computationally efficient statistical signal processing technique for revealing hidden factors that underlie sets of random variables, measurements or signals. A generative model for the observed multivariable data, which is typically given as a large database of samples is defined by ICA. The data variables in the model are assumed to be linear or non-linear mixtures of some unknown latent variables and the system of mixing is unknown. The extraction of source in this process is done based on the assumption that the latent variables are non-Gaussian and statistically independent [15].

Suppose a set of observations of random variables is $(x_1(t)\, x_2(t)...x_n(t))$ where 't' is the time or the sample index and they are generated from a linear mixture of sources that are statistically independent. This is expressed in the following form

$$[x_1(t)\, x_2(t)...x_n(t)] = A[s_1(t)\, s_2(t)...s_n(t)] \qquad (1)$$

where A is some unknown mixing matrix and ^{T} stands for the transpose of a matrix.

Independent Component Analysis estimates both A and '$s_i(t)$' when only the observation '$x_i(t)$' are at hand. Number of independent components here is assumed to be equal to the number of observed mixtures.

978-1-4244-1923-4/08/$25.00 ©2008 IEEE

A. The BFOICA Algorithm

Bacterial foraging is a new evolutionary computational method proposed by Passino [16].In this scheme, the foraging behavior of *E.coli* bacteria present in our intestines is mimicked. They undergo different stages such as chemotaxes, swarming, reproduction and elimination and dispersal. The detailed treatment of this new concept is presented in [16].

With BFO algorithm kurtosis [13] is used as the contrast function to be maximized and hence the requirement of minimization of the nutrient function J, defined as

$$J = \frac{1}{ContrastFunction} \quad (2)$$

In Bacteria Foraging based optimization random column vector w, which is represented as two bacteria, is used to find the linear transformation $w^T z$.

Step-1: Data Centering

The mean $X_m = (x_{1m}, x_{2m} ..., x_{nm})^T$ of the observed mixed signal data $X = (x_1, x_2 ..., x_n)^T$ is computed and the mean is subtracted from the observed data set to make it zero mean.

$$X_c = X - X_m \quad (3)$$

Step-2: Whitening

The covariance matrix $CovX$ of the centered data X_c is computed. The eigenvalue decomposition of $CovX$ is performed. If D is the eigenvalue matrix and E is the eigenvector matrix then

$$Z = D^{-1/2} E^* X_c \quad (4)$$

Step-3: Initialization

Number of parameters p to be optimized, number of bacteria S, swimming length N_s after which tumbling of bacteria will be undertaken in a chemotatic loop, number of iterations N_c to be under taken in a chemotatic loop $N_c > N_s$ number of reproduction steps N_{re}, the elimination and dispersal probability P_{ed}, the location of each bacterium $P(1-p,1-S,1)$ and the value of $C(i)$ are initialized for the optimization algorithm.

Step-4: Iterative algorithm for optimization

The section models the bacterial population chemotaxis, reproduction, elimination and dispersal (initially, $j = k = l = 0$). For the algorithm updating θ^i automatically results in updating of P

i) Elimination-dispersal loop: $l = l + 1$;

ii) Reproduction loop: $k = k + 1$;

iii) Chemotaxis loop: $j = j + 1$

a) For $i = 1,2,....., S$, calculate cost function value for each bacterium ***i*** as follows.

*Compute value of cost function $J(i, j, k, l)$.

*let $J_{last} = J(i, j, k, l,)$ to save his value since we may find a better cost via a run.; *End of for loop.

b) For $i = 1,2,....S$ take the tumbling /swimming decision

*Tumble: Generate a random vector $\Delta(i) \in \Re^p$ with each element $\Delta_m(i) m = 1,2,....p,$ a random number on [-1,1].

*Move:

Let. $$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (5)$$

Fixed step size in the direction of tumble for bacterium i is considered.

*Compute $J(i, j+1, k, l)$,

*Swim: i) Let $m = 0$;(counter for swim length)

ii) while $m < N_s$(have not climbed down too long)

*Let $m = m + 1$, *If $J(i, j+1, k, l) < J_{last}$ (if doing better),

let $J_{last} = J(i, j+1, k, l)$ and

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

(6) and use this $\theta^i(j+1,k,l)$ to compute the new $J(i, j+1, k, l)$.

*Else, let $m = N_s$. This is the end of the while statement.

c) Go to the next bacterium $(i+1)$ if $i \neq S$ (i.e go to b) to process the next bacterium.

iv) If $j < N_c$, go to (iii). In this case, continue chemotaxis since the life of bacteria is not over.

v) Reproduction: a) For the given **k** and ***l***, and for each $i = 1,2,....S$, let $J^i_{health} = \min_{j \in \{1...N_s\}} \{j\ (i, j, l)\}$ be the health of the bacterium ***I*** (a measure of how many nutrients it got over its life time and how successful it was at avoiding noxious substance). Sort bacteria in order of ascending cost J_{health} (higher cost means lower health).

b) The $S_r = S/2$ bacteria with highest J_{health} values die and other S_r bacteria with the best value split (and the copies that are made are placed at the same location as their parent)

vi) If $k < N_{re}$ go to (ii), in this case, we have not reached the number of specified reproduction steps, so we start the next generation in the chemotactic loop.

vii) Elimination & dispersal: $i = 1,2,...S$ with probability P_{ed} eliminate and disperse each bacterium to a random location on the optimization domain. The position of the bacteria w_1 at which global minimum value is obtained yields the first independent component.

Step-5: Evaluation of Second Independent Component

To estimate the other ICs step 3 of the algorithm is repeated for getting weight vectors $w_2,...,w_n$. To prevent different vectors from converging to the same optimum and hence the same IC, the weight vectors are decorrelated using Gram-Schmidt like orthogonaliztion. When p vectors $w_1,...,w_p$ have been estimated, step 3 is run for w_{p+1} and after every iteration step the following iteration steps are performed.

$$w_{p+1} = w_{p+1} - \sum_{j=1}^{p} (w_{p+1}^T w_j) w_j \quad (7)$$

$$w_{p+1} = \frac{w_{p+1}}{\sqrt{w_{p+1}^T w_{p+1}}} \quad (8)$$

Above equations constrain the bacteria foraging based optimization process.

B. *The CGAICA Algorithm*

With Genetic Algorithm based optimization [14] we use kurtosis as the contrast function to be optimized. A random column vector w which is represented as two chromosomes is used to find the linear transformation $w^T z$. The preprocessing of data is carried out as described for BFOICA algorithm above and then the data is presented for optimization.

The following iterative procedure is followed to implement GA based ICA algorithm.

1) An initial population $\{\hat{\theta}_i\}_{i=1}^{N}$ of size N is created from a random initial set of parameter. The encoding length of each parameter is 15 bits. By decoding the individual to get the parameter of the system, the fitness for each individual is evaluated.

2) Two mates are selected for reproduction with probabilities proportional to their using tournament selection.

3) The multipoint crossover operator with crossover operator with crossover probability P_c is applied to the two mates and a pair of offspring are generated.

4) The mutation operator with probability P_m is applied to the newly generated offspring.

5) The fitness value for the off spring are computed after they are decoded as the parameter sets of the parametric system.

6) Steps 2-5 are repeated until an entirely new population of individuals is generated.

7) The previous population is replaced with the new population with the addition of an elitist selection.

8) If the stopping criterion is satisfied, go to step 11.

9) If generation number is greater than a predetermined value go to step 2.

10) Reinitialize the population survival, go to step 2.

11) Output the individual with the best fitness value and terminate the iterative procedure.

Step-2: Evaluation of Second Independent Component

The second IC is computed as in step 5 of BFOICA and (7) and (8) constrain the GA based optimization process.

III. FIXED-POINT MODELS

Any algorithm performs various arithmetic operations such as additions, subtractions, multiplications and divisions. When any of these operations is carried out with a fixed number of bit's then either truncation or round operation is performed. Hence for every operation an error signal is introduced at the location of that operation which appears at the output as noise. The fixed-point model introduces these truncation and rounding operations at each step of the algorithm so that one gets the fixed-point output of the system which can be compared with the corresponding output with full precision value. For each fixed-point addition and multiplication truncation and rounding operations are carried out respectively. Such a choice has been reported to introduce less mean square error in actual implementation. In the following algorithm $[.]_F$ denotes the fixed-point value of the parameter [.] and't' denotes the finite register length or number of bits without the sign bit. With sign bit the actual register length becomes t+1. For example fixed-point addition of two numbers A and B is performed as $[A+B]_F$=add(A,B,t), where add function represents both A and B in t-bits, adds them and then truncates the result after t-bits. Certain complex fixed-point operations are done by special functions mentioned below as and when required. The analysis is carried out for a mixture of two signals for the sake of simplicity which can be easily extended for more than two signals.

A. *The BFOICA Algorithm*

1. Centering the data matrix X:

$$[meanx_1]_F = ((((x_{11} + x_{12})_F + x_{13})_F + ... + x_{1N})/N)_F \quad (9a)$$

$$[meanx_2]_F = ((((x_{21} + x_{22})_F + x_{23})_F + ... + x_{2N})/N)_F \quad (9b)$$

where $meanx_1$, $meanx_2$ and N denote the mean of first row of X, mean of second row of X and number of columns of X. Mean is subtracted from the data X to get the centered data $(cx_{1i}; cx_{2i})^T$.

$$cx_{1i} = (x_{1i} - [meanx_1]_F)_F \quad (10a)$$

$$cx_{2i} = (x_{2i} - [meanx_2]_F)_F \quad (10b)$$

2. The centered data is whitened by finding out its covariance matrix and then by eigen value decomposition. Let covX is the covariance matrix of centered data.

$$[\mathrm{cov}X_{11}]_F = [((((x_{11}.x_{11})_F + (x_{12}.x_{12})_F)_F + (x_{13}.x_{13})_F)_F + \ldots + (x_{1N}.x_{1N})_F)_F / N]_F \quad (11a)$$

$$[\mathrm{cov}X_{11}]_F = [((((x_{11}.x_{11})_F + (x_{12}.x_{12})_F)_F + (x_{13}.x_{13})_F)_F + \ldots + (x_{1N}.x_{1N})_F)_F / N]_F \quad (11b)$$

$$[\mathrm{cov}X_{22}]_F = [((((x_{21}.x_{21})_F + (x_{22}.x_{22})_F)_F + (x_{23}.x_{23})_F)_F + \ldots + (x_{2N}.x_{2N})_F)_F / N]_F \quad (11c)$$

$$[\sigma]_F = [([\mathrm{cov}X_{22}]_F - [\mathrm{cov}X_{11}]_F)_F / (2 * [\mathrm{cov}X_{22}]_F)_F]_F \quad (12)$$

$$[T]_F = [sign(\sigma)_F / [abs(\sigma)_F + [\sqrt{[1 + [(\sigma)_F.(\sigma)_F]_F]_F}]_F]_F \quad (13)$$

where $[\sqrt{y}]_F$ =scaled value of √y.

$$[c]_F = [1/[\sqrt{1 + [(T)_F(T)_F]_F}]_F]_F \quad (14)$$

$$[s]_F = [[T]_F[c]_F]_F \quad (15)$$

$$[E]_F = \begin{bmatrix} [c]_F & [s]_F \\ -[s]_F & [c]_F \end{bmatrix} \quad (16)$$

$$[D]_F = [[E]_F^T * [[\mathrm{cov}X]_F * [E]_F]_F]_F \quad (17)$$

Here the matrix multiplications are implemented by a function which does fixed point multiplication and additions required for t-bits.

$$\text{If } [D]_F = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \text{ then } [D^{-1/2}]_F = \begin{bmatrix} [\lambda_1^{-1/2}]_F & 0 \\ 0 & [\lambda_2^{-1/2}]_F \end{bmatrix} \quad (18)$$

$$\text{With } [\lambda_1^{-1/2}]_F = [\frac{1}{[\sqrt{\lambda_1}]_F}]_F \text{ and } [\lambda_2^{-1/2}]_F = [\frac{1}{[\sqrt{\lambda_2}]_F}]_F \quad (19)$$

$$[whitened\_matrix]_F = [[D^{-1/2}]_F * [E^{-1/2}]_F]_F \quad (20)$$

$$[whitened\_X]_F = [[whitened\_matrix]_F * [X]_F]_F \quad (21)$$

$$[Z]_F = [whitened\_X]_F \quad (22)$$

3) Initial random vector $w = (w_1, w_2)^T$ is chosen and norm of w is calculated as

$$[norm\_w]_F = [\sqrt{[w_1 * w_1]_F + [w_2 * w_2]_F}]_F \quad (23)$$

$$[w]_F = ([w_1 / [norm\_w]_F]_F, [w_2 / [norm\_w]_F]_F) \quad (24)$$

The normalized fixed-point value of w is given in (24).

4) BFOICA Iteration process :

The direction for bacterial search algorithm is given by Delta. The fixed-point representation of Delta is

$$[Delta]_F = [[2 * [round(rand(p,1)) - 1]_F]_F .* rand(p,1)]_F \quad (25)$$

where rand and round are the random and rounding operators respectively.

$$[u]_F = [[w_1 / [norm\_w]_F]_F * y]_F \quad (26)$$

$[norm\_w]_F$ is norm of w and represented as in (23).

$$[P]_F = [mean[u^4]_F, 2]_F \quad (27)$$

The nutrient function in fixed point is represented as

$$[J]_F = [1/abs[P-3]_F]_F \quad (28)$$

where abs is the absolute operator and P is represented in (27).

B. The CGAICA Algorithm

The fixed-point models for preprocessing steps like centering and whitening remains same as described above for BFOICA algorithm. The decoding of parent and children population is carried out as follows

$$[original\ decoded\ value]_F = [range\ \min + [[[range\ \max - range\ \min]_F / [2^{[n_1/p]_F} - 1]_F]_F * decoded\ value]_F]_F \quad (29)$$

In the above equation range min and range max are the minimum and maximum range for the GA optimization process. n_1 and p represent the number of bits and number of parameters to be optimized in binary coded GA.

The fitness function in fixed point is represented as

$$[J]_F = [abs[P-3]_F]_F \quad (30)$$

where P is represented in (27).

Since binary coded GA is used for optimization in CGAICA, number of bits in GA is kept same as the finite register length for the fixed-point analysis.

IV. Experimental Setup

In the simulation experiment study fixed-point as well as floating point programs of BFOICA and CGAICA are written. The fixed-point programs are equivalent to fixed-point machines capable of simulating the operations of any word size. With a known value of t, the fixed-point iteration for BFOICA algorithm with *kurtosis* as the optimization function is run and the mean square error (MSE) is computed. The MSE is computed as

$$MSE = \frac{\sum_{t=1}^{n} (s(t) - y(t))^2}{n} \quad (31)$$

where n, $s(t)$ and $y(t)$ stand for the number of samples, actual signal and estimated signal respectively.

The value of t is varied from 7 to 32 and corresponding MSEs are computed. The same procedure was followed for CGAICA algorithm with *kurtosis* as optimization function. From the MSEs and output signal powers the noise to signal ratio (NSR) for each t is computed for both the algorithms.

V. Analysis of Results

The simulation studies, on fixed-point models as described in the previous section for BFOICA and CGAICA algorithm with *kurtosis* as contrast function are performed. The mean square error (MSE) decreases with increase in the register length for all cases as depicted in Table 1.

TABLE 1
COMPARISON OF MSE OF DIFFERENT ICAS AT DIFFERENT BIT LENGTHS

No of Bits	MSE	
	BFOICA	CGAICA
7	9.3×10^{-3}	1.56×10^{-2}
8	1.9×10^{-3}	5.2×10^{-3}
10	1.0997×10^{-4}	2.2334×10^{-4}
12	3.0983×10^{-7}	9.43×10^{-6}
14	1.0714×10^{-7}	1.5372×10^{-6}
16	8.4878×10^{-7}	8.4302×10^{-6}
20	1.2013×10^{-6}	8.703×10^{-7}
24	1.1135×10^{-6}	2.6459×10^{-6}
32	1.1130×10^{-6}	3.9018×10^{-7}

The noise to signal ratio (NSR) parameter in dB as a function of register length has been plotted. Fig.1 shows the NSR variation with t for BFOICA and CGAICA algorithm. This clearly indicates that NSR decreases with increase in t in both the cases. At lower register (bit) lengths (t<16) BFOICA consistently performs better than CGAICA. Also it is observed that at higher bit lengths CGAICA yields comparable or better performance. This is because increasing the number of bits of binary coded GA will tend towards real coded GA. NSR is observed to be minimum at register length 14 for both the algorithms. The NSR fluctuations at certain bit lengths for both the algorithms is due to the need of variations of the different tuning parameters that greatly influence the computational overhead.

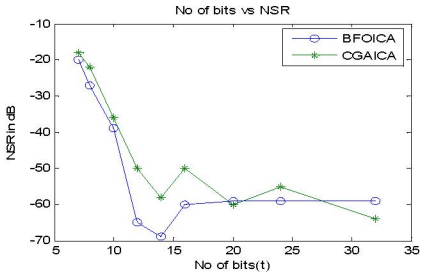

Fig.1: NSR variation with bit length for BFOICA and CGAICA with *kurtosis* as contrast function.

Figs.2 & 3 show the original signals (a simple case of rectangular wave and noise has been considered) and the mixed signals respectively.

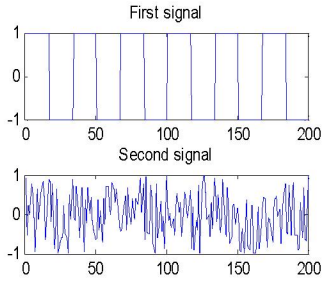

Fig.2 Original Signals

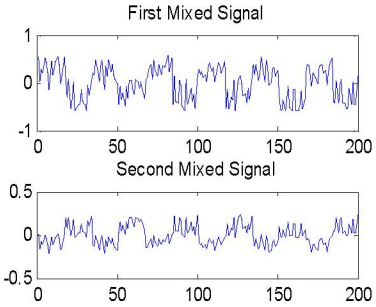

Fig.3 Mixed Signals

Figs.4&5 also display the separation with register lengths 8 and 16 respectively for BFOICA algorithm.

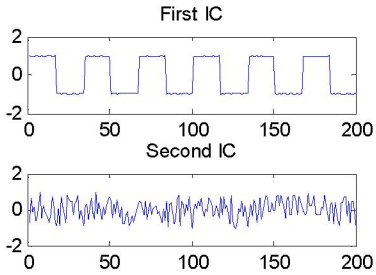

Fig.4 Separated Signals for BFOICA with 8 bits

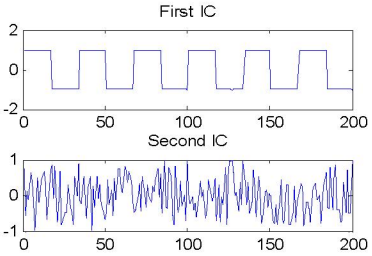

Fig.5 Separated Signals for BFOICA with 16 bits

The separation of signals with register lengths of 8 and 16 are shown in Figs.6 & 7 respectively for the CGAICA. It can be well observed that separation with BFOICA is better than that with CGAICA with 8 bits and also same is the case for 16 bits.

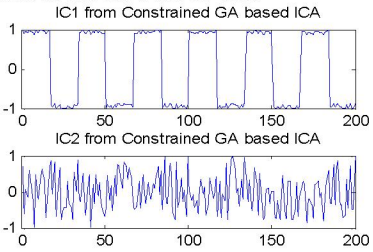

Fig.6 Separated Signals for CGAICA with 8 bits

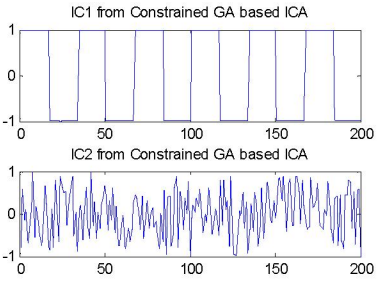

Fig.7 Separated Signals for CGAICA with 16 bits

VI. CONCLUSIONS

The present paper studies the effect of finite register length on the accuracy of two different evolutionary computation based ICA algorithms BFOICA and CGAICA. Extensive simulation studies reveal that *kurtosis* based CGAICA yields higher MSE compared to *kurtosis* based BFOICA. Further for the same bit length, the fixed-point BFOICA offers substantially low MSE compared to the CGAICA (*kurtosis* based). The separation ability of fixed-point ICA depends on the number of bits used. Fixed-point BFOICA performs superior to the fixed-point CGAICA.

REFERENCES

[1] P. Comon, "Independent Component Analysis-A new concept?", Signal Processing, vol-36,pp.287-314,1994.

[2] L.Potanutis et.al., "Independent Component Analysis applied to Feature Extraction for Robust Automatic Speech Recognition", Electronics Lett., vol-36, No.23,pp.1977-1978, Nov.2000.

[3] S.Z.Li, et.al.,"Learing Multiview Face Subspaces and Facial Pose Estimation using Independent Component Analysis", IEEE Trans. Image Processing, vol-14, No.6,pp.705-712,June 2005.

[4] K.Nojun, Chong-Ho Chai, "Feature Extraction Based on ICA or Binary Classification Problem",IEEE Trans. Knowledge and data Engineering, vol-15, No.6, pp.1374-1388,Nov-Dec 2003.

[5] I.Dagher,R.Nachar, "Face Recognition using IPCA-ICA Algorithm", IEEE Trans. Pattern Analysis and Machine Intelligence, vol-28,No.6,pp.996-1000, June 2006.

[6] M.D.Plumbky, "Algorithm for nonnegative Independent Component Analysis", IEEE Trans. Neural Networks, vol-14, No.3,pp.534-543, May 2003.

[7] Li Hong, Sun Y. "The Study and test of ICA algorithms", International Conference on Wireless Communications, Networking and Mobile Computing, vol-1,pp.602-605, 2005.

[8] Chang-Min Kim et al. "FPGA Implementation of ICA Algorithm for Blind Signal Separation and Adaptive Noise Cancellation", IEEE Trans. on Neural Networks, vol-14, No.15,pp1038-1046, 2003.

[9] Charoensak C. et. al." A Single-Chip FPGA Design for Real-time ICA-based Blind Source Separation Algorithm", IEEE International Symposium on Circuits and Systems, vol-6,pp.5822-5825, 2005.

[10] A.M.Reza, L.Zhu, "Analysis of Error in the Fixed-Point Implementation of Two-Dimensional Discrete Wavelet Transforms", IEEE Trans. Circuits and Systems –I, vol.-53, No-3, pp.641-655, Mar. 2005.

[11] G. Panda, R.N.Pal,.and B.Chatterjee, "Error Analysis of Good-Wino grad Algorithm Assuming Correlator Truncation Error", IEEE Trans. on Acoustic., Speech and Signal Processing, pp.502-512, Apr.1983.

[12] D.P.Acharya, G.Panda and Y.V.S.Lakshmi, "Fixed-point Error Evaluation of Fast ICA and Algebraic ICA Algorithms", IEEE International Conference on Industrial Technology, Mumbai, India, Dec. 2006.

[13] D.P.Acharya, G.Panda, S.Mishra and Y.V.S.Lakshmi, "Bacteria Foraging Based Independent Component Analysis", Accepted for presentation in IEEE International Conference on Computational Intelligence and Multimedia Applications, Sivakasi, India, Dec. 2007.

[14] D.P.Acharya, G.Panda and Y.V.S.Lakshmi, "Constrained Genetic Algorithm Based Independent Component Analysis", IEEE Congress on Evolutionary Computation, Singapore, Sept. 2007.

[15] A. Hyverinen., J.Kahrunen. and E. Oja, " Independent Component Analysis", John Wiley & Sons, 2001.

[16] K.M.Passino, "Biomimicry of Bacterial Foraging for distributed optimization and control," IEEE Control Syst. Mag., vol.22, no.3, pp52-67, Jun.2002.