Investigation on Raster CNN Simulation by Numerical Integration Algorithms

R. Ponalagusamy, S. Senthilkumar
Department of Mathematics,
National Institute of Technology,
Tiruchirappalli-620 015, Tamilnadu, INDIA
rpalagu@nitt.edu; ssenthilkumar1974@yahoo.co.in

October 1, 2007

Abstract

The aim of this article is focused on developing an efficient algorithm for simulating Cellular Neural Network arrays (CNNs) using numerical integration techniques. The role of the simulator is that it is capable of performing raster simulation for any kind as well as any size of input image. It is a powerful tool for researchers to investigate the potential applications of CNN. This article proposes an efficient pseudo code for exploiting the latency properties of Cellular Neural Networks along with well known numerical integration algorithms. Simulation results and comparison have also been presented to show the efficiency of the numerical integration algorithms. It is observed that the Runge-Kutta (RK)sixth order algorithm outperforms well in comparison with the Explicit Euler, RK-Gill and RK-fifth order algorithms.

Key-words: Cellular Neural Networks, Numerical Integration Techniques, Raster CNN Simulation, Edge Detection.

1 Introduction

It is understood that the characteristics of Cellular Neural Networks (CNNs) are analog, time-continuous, non-linear dynamical systems and formally belong to the class of recurrent neural networks. CNNs have been proposed

by Chua and Yang [6, 7], and they have found that CNN has many important applications in signal and real-time image processing [16]. Roska et al. [17] have presented the first widely used simulation system which allows the simulation of a large class of CNN and is especially suited for image processing applications, real-time image processing (feature extraction, motion detection etc.,) patttern recognition, analysis of $3D$ complex surfaces (minima, maxima detection etc.,) solving ordinary and partial differential equations etc.

RK techniques have become very popular for computational purpose [3, 4]. RK algorithms are used to solve differential equations efficiently that are equivalent to approximate the exact solutions by matching 'n' terms of the Taylor series expansion. Butcher [4] derived the best RK pair along with an error estimate and by all statistical measures it appeared as the RK-Butcher algorithms. This RK-Butcher algorithm is nominally considered sixth-order since it requires six functions evaluation, but in actual practice the "working order" is equivalent to five (fifth order).

The RK-Butcher algorithm has been introduced by Morris Bader [1, 2] for finding the truncation error estimates and intrinsic accuracies and the early detection of stiffness in coupled differential equations that arises in theoretical chemistry problems.

Ponalagusamy and Senthilkumar [14] have adapted the RK-sixth order algorithm for computing the Time-Multiplexing CNN simulation using limiting formulas of RK(7,8). Oliveria [12] introduced the popular RK-Gill algorithm for evaluation of effectiveness factor of immobilized enzymes.

Chi-Chien Lee and Jose Pineda de Gyvez [5] introduced Euler, Improved Euler, Predictor-Corrector and Fourth-Order (quadratic) Runge-Kutta algorithms in Raster CNN simulation. Leon et al [10] proposed an efficient algorithm for converting digital documents to multi-layer raster formats. Hadad and Piroozmand [11] have described the application of a multi-layer cellular neural network to model and solve the nuclear reactor dynamic equations. In this article, the raster CNN simulation problem is solved with different approach using the algorithms such as Explicit Euler, RK-Gill, RK-fifth order and RK-sixth order to yield higher accuracy, with less error.

2 Structure and Functions of Cellular Neural Network

CNN is a hybrid of Cellular Automata and Neural Networks and it shares the best features of both areas. Like Neural Networks, its continuous time feature allows real-time signal processing, and like Cellular Automata, its local interconnection feature makes VLSI realization feasible. Its grid-like structure is suitable for the solution of a high order system of first order non-linear differential equations on-line and in real-time. CNN is an analog nonlinear dynamic processor array shown Figure 1(a). The following are the features of CNN [8].

1. Each analog processor is capable of processing continuous signals, in either continuous-time or discrete-time modes.

2. The processors are placed on a 3D geometric cellular grid (several 2D layers) and are basically similar.

3. Interaction among processors is local and mainly translation invariant.

4. The mode of operation may be transient, equilibrium, periodic, chaotic, or combined with logic (without Analog to /Digital Conversion)

The general CNN architecture consists of $M * N$ cells placed in a rectangular array. The basic circuit unit of CNN is called a cell. It contains linear and nonlinear circuit elements. Any cell, $C(i,j)$ is connected only to its neighbor cells (adjacent cells interact directly with each other). This intuitive concept is called neighborhood and is denoted by $N(i,j)$. Cells not in the immediate neighborhood have indirect effect because of the propagation effects of the dynamics of the network. Each cell has a state x, input u, and output y. The state of each cell is bounded for all time $t > 0$ and after the transient has settled down, a cellular neural network always approaches one of its stable equilibrium points. This last fact is relevant because it implies that the circuit will not oscillate. The dynamics of a CNN has both output feedback (A) and input control (B) mechanisms.

The first order nonlinear differential equation defining the dynamics of a cellular neural network cell can be written as follows.

$$c\frac{dx_{ij}(t)}{dt} = \begin{cases} -\frac{1}{R}x_{ij}(t) + \sum_{c(k,l)\in N(i,j)} A(i,j;k,l)y_{kl}(t) \\ +\sum_{c(k,l)\in N(i,j)} B(i,j;k,l)u_{kl}(t) + I, \quad 1 \le i \le M, \quad 1 \le j \le N \end{cases} \tag{2.1}$$

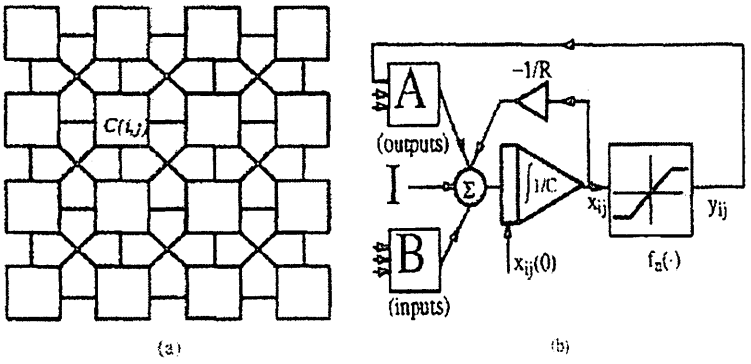

Figure 1: Cellular Neural Networks:(a) Array Structure and (b) Block Diagram

and the output equation is given by

$$y_{ij}(t) = \frac{1}{2}\{|\, x_{ij}(t)+1\,| - |\, x_{ij}(t)-1\,|\} \quad 1 \le i \le M, \quad 1 \le j \le N, \ (2.2)$$

where x_{ij} is the state of cell $C(i,j)$, $x_{ij}(0)$ is the initial condition of the cell, c is the linear capacitor, R is the linear resistor, I is an independent current source, $A(i,j;k,l)y_{kl}$ and $B(i,j;k,l)u_{kl}$ are voltage controlled current sources for all cells $C(k,l)$ in the neighborhood $N(i,j)$ of cell $C(i,j)$, and y_{ij} represents the output equation.

From the equation (2.1) it is observed that the summation operators of each cell is affected by its neighboring cells. $A(.)$ represents on the output of neighboring cells and is called as feedback operator, $B(.)$ in turn affects the input control and is known as the control operator. In particular, the entry values of matrices $A(.)$ and $B(.)$ are dependent on the application chosen by the user which are space invariant and are referred as cloning templates. A current bias I and cloning templates establishes the transient behavior of the cellular nonlinear network. A continuous-time cell implementation is shown in figure 1(b) as an equivalent block diagram. CNNs have as input a set of analog values and its programmability is done via cloning templates. Thus, programmability is one of the most attractive properties of CNNs.

3 Performance of Raster CNN Simulations

Raster CNN simulation is an image scanning-processing technique for solving the system of difference equations of CNN. The equation (2.1) is space invariant, which means that $A(i,j;k,l) = A(i-k,j-1)$ and $B(i,j;k,l) = B(i-k,j-1)$ for all i,j,k,l. Therefore, the solution of the system of difference equations can be seen as a convolution process between the image and the CNN processors. The fundamental approach is to imagine a square sub-image area centered at (x,y), with the sub-image being the same size of the templates involved in the simulation. The center of this sub-image is then moved from pixel to pixel starting, say, at the top left comer and applying the A and B templates at each location (x,y) to solve the differential equation. This procedure is repeated for each time step, for all the pixels in the image. An instance of this image scanning-processing is referred to as "iteration". The processing stops when it is found that the states of all CNN processors have converged to steady-state values, and the outputs of its neighbor cells are saturated, e.g. they have value [6, 7]. This whole simulating approach is referred to as raster simulation. A simplified pseudo code is presented below gives the exact notion of this approach.

3.1 Pseudo Code for Raster CNN Simulation

Step 1. Initially get the input image, initial conditions and templates from end user.

$/* M, N$ =Number of rows and columns of the $2D$ image $*/$

while (converged-cells < total number of cells)

{

for $(i = 1; i <= M; i++)$ for $(j = 1; j <= N; j++)$

{

if (convergence-flag[i] [j])continue;

/* current cell already converged */

Step 2. /*Calculate the next state*/.

$x_{ij}(t_{n+1}) = x_{ij}(t_n) + \int_{t_n}^{t_{n+1}} f'(x(t_n))dt$

Step 3. /* Check the convergence criteria*/

if $\left(\frac{dx_{ij}(t_n)}{dt}\right) = 0$ and $y_{kl} = \pm 1$ for all $c(k,l) \in N_r(i,j)$

{

convergence-flag[i][j] = 1;

converged-cells++ ;

}

}

/* end for */

Step 4 /* Update the state values of the entire image */

for ($i = 1; i <= M; i++$) for ($j = 1; j <= N; j++$)

{

if (convergence-flag[i][j]) continue;

$x_{ij}(t_n) = x_{ij}(t_{n+1})$;

}

Number of iteration++;

}

/* end while */

For simulation purposes, a discretized form of equation (2.1) is solved within each cell to simulate its state dynamics. One common way of processing a large complex image is using a raster approach [6, 7]. This approach implies that each pixel of the image is mapped onto a CNN processor. That is, it has an image processing function in the spatial domain that is expressed as:

$$g(x,y) = T(f(x,y)) \tag{3.1}$$

where $f(.)$ is the input image, $g(.)$ the processed image, and T is an operator on $f(.)$ defined over the neighborhood of (x,y). It is an exhaustive process

from the view of hardware implementation. For practical applications, in the order of 250,000 pixels, the hardware would require a large amount of processors which would make its implementation unfeasible. An alternative option to this scenario is multiplex the image processing operator.

4 Numerical Integration Techniques

The CNN is described by a system of nonlinear differential equations. Therefore, it is necessary to discretize the differential equation for performing behavioral simulation. For computational purposes, a normalized time differential equation describing CNN is used by Nossek et al.,[13]

$$f'(x(n\tau)) = \frac{dx_{ij}(n\tau)}{dt} \tag{4.1}$$

$$= \begin{cases} -x_{ij}(n\tau) + \sum_{c(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(n\tau) \\ +\sum_{c(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl} + I, \\ 1 \le i \le M,\ 1 \le j \le N \end{cases} \tag{4.2}$$

$$y_{ij}(n\tau) = \frac{1}{2}\{\mid x_{ij}(n\tau)+1 \mid - \mid x_{ij}(n\tau)-1 \mid\} \quad 1 \le i \le M, \quad 1 \le j \le N, \tag{4.3}$$

where τ is the normalized time. For the purpose of solving the initial-value problem, well established Single Step methods of numerical integration techniques are used [15]. These methods can be derived using the definition of the definite integral

$$x_{ij}((n+1)\tau) - x_{ij}(n\tau) = \int_{\tau_n}^{\tau_{n+1}} f'(x(n\tau))d(n\tau) \tag{4.4}$$

Explicit Euler's, the Improved Euler Predictor-Corrector and the RK - fourth order are the mostly widely used single step algorithm in the CNN behavioral raster simulation. These methods vary in the way they evaluate the integral presented in [4].

4.1 Explicit Euler's Algorithm

Euler's method is the simplest of all algorithms for solving ordinary differential equations. It is an explicit formula which uses the Taylor series expansion to calculate the approximation.

$$x_{ij}((n+1)\tau) = x_{ij}(n\tau) + \tau f'(x(n\tau)) \tag{4.5}$$

4.2 RK-Gill Algorithm

The RK-Gill algorithm was discussed by Oliveria [12] is an explicit method, which requires the computation of four derivatives per time step. The increase of the state variable x_{ij} is stored in the constant k_1^{ij}. This result is used in the next iteration for evaluating k_2^{ij} and repeat the same process to obtain the values of k_3^{ij} and k_4^{ij}.

$$
\begin{aligned}
k_1^{ij} &= f'(x_{ij}(n\tau)) \\
k_2^{ij} &= f'(x_{ij}(n\tau)) + \frac{1}{2}k_1^{ij} \\
k_3^{ij} &= f'(x_{ij}(n\tau)) + (\frac{1}{\sqrt{2}} - \frac{1}{2})k_1^{ij} + (1 - \frac{1}{\sqrt{2}})k_2^{ij} \\
k_4^{ij} &= f'(x_{ij}(n\tau)) - (\frac{1}{\sqrt{2}})k_2^{ij} + (1 + \frac{1}{\sqrt{2}})k_3^{ij}.
\end{aligned}
$$

Therefore, the final integration is a weighted sum of the four calculated derivatives is given below.

$$x_{ij}((n+1)\tau) = x_{ij}(n\tau) + \frac{1}{6}\{k_1^{ij} + (2 - \sqrt{2})k_2^{ij} + (2 + \sqrt{2})k_3^{ij} + k_4^{ij}\} \quad (4.6)$$

4.3 RK-Fifth Order Algorithm

The RK-fifth order algorithm is an explicit method discussed by Morris Badder [1, 2]. It starts with a simple Euler method. The increase of the state variable x_{ij} is stored in the constant k_1^{ij}. This result is used in the next iteration for evaluating k_2^{ij}. The same procedure must be repeated to compute the values of $k_3^{ij}, k_4^{ij}, k_5^{ij}$ and k_6^{ij}.

$$
\begin{aligned}
k_1^{ij} &= \tau f'(x_{ij}(n\tau)), \\
k_2^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{1}{4}k_1^{ij}, \\
k_3^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{1}{8}k_1^{ij} + \frac{1}{8}k_2^{ij}, \\
k_4^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{1}{2}k_2^{ij} + k_3^{ij}, \\
k_5^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{3}{16}k_1^{ij} + \frac{9}{16}k_4^{ij}, \\
k_6^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{3}{27}k_1^{ij} + \frac{2}{7}k_2^{ij} + \frac{12}{7}k_3^{ij} - \frac{12}{7}k_4^{ij} + \frac{8}{7}k_5^{ij}.
\end{aligned}
$$

Therefore, the final integration is a weighted sum of the five calculated derivatives which is given below.

$$x_{ij}((n+1)\tau) = x_{ij}(n\tau) + \frac{1}{90}\{7k_1^{ij} + 32k_3^{ij} + 12k_4^{ij} + 32k_5^{ij} + 7k_6^{ij}\} \quad (4.7)$$

where $f(.)$ is computed according to (2.1).

4.4 RK-Sixth Order Algorithm

The RK-sixth order algorithm is an explicit method discussed by Ponalagusamy and Senthilkumar [14]. It starts with a simple Euler method. The increase of the state variable x_{ij} is stored in the constant k_1^{ij}. This result is used in the next iteration for evaluating k_2^{ij} . The same procedure must be repeated to compute the values of k_3^{ij} k_4^{ij} k_5^{ij} k_6^{ij} and k_7^{ij}.

$$k_1^{ij} = \tau f'(x_{ij}(n\tau)),$$
$$k_2^{ij} = \tau f'(x_{ij}(n\tau)) + \frac{1}{2}k_1^{ij},$$
$$k_3^{ij} = \tau f'(x_{ij}(n\tau)) + \frac{2}{9}k_1^{ij} + \frac{4}{9}k_2^{ij},$$
$$k_4^{ij} = \tau f'(x_{ij}(n\tau)) + \frac{7}{36}k_1^{ij} + \frac{2}{9}k_2^{ij} - \frac{1}{12}k_3^{ij},$$
$$k_5^{ij} = \tau f'(x_{ij}(n\tau)) - \frac{35}{144}k_1^{ij} - \frac{55}{36}k_2^{ij} + \frac{35}{48}k_3^{ij} + \frac{15}{8}k_4^{ij},$$
$$k_6^{ij} = \tau f'(x_{ij}(n\tau)) - \frac{1}{360}k_1^{ij} + \frac{11}{36}k_2^{ij} - \frac{1}{8}k_3^{ij} + \frac{1}{2}k_4^{ij} + \frac{1}{10}k_5^{ij},$$
$$k_7^{ij} = \tau f'(x_{ij}(n\tau)) - \frac{41}{260}k_1^{ij} + \frac{22}{13}k_2^{ij} + \frac{43}{156}k_3^{ij} - \frac{118}{39}k_4^{ij} + \frac{32}{195}k_5^{ij} + +\frac{80}{39}k_6^{ij}$$

Therefore, the final integration is a weighted sum of the seven calculated derivatives which is given below.

$$x_{ij}((n+1)\tau) = x_{ij}(n\tau) + \frac{13}{200}\{k_1^{ij} + \frac{11}{40}k_3^{ij} + \frac{11}{40}k_4^{ij} + \frac{4}{25}k_5^{ij} + \frac{4}{25}k_6^{ij} + \frac{13}{200}k_7^{ij}\} \tag{4.8}$$

where $f(.)$ is computed according to (2.1).

5 Simulation Results and Comparisons

All the simulated outputs presented below here are performed using a high power workstation, and the simulation time used for comparisons is the actual CPU time used. The input image format is the X windows bitmap format (xbm), which is commonly available and easily convertible from popular image formats like GIF or JPEG. Figures (2) and (3) show the results of the raster simulator obtained from a complex image of 1,25,600 pixels. Using RK-fifth order and RK-sixth order algorithms the results of the raster simulator obtained from a complex image of 1, 25,600 pixels are depicted respectively in figures (2) and (3). For the present example an

averaging template followed by an edge detection template were applied to the original image to yield the images displayed in figures 2(a) and 2(b), respectively. The same procedure has been adapted for getting the results shown in figures 3 (a) and 3(b). It is observed from figures (2) and (3) that the edges obtained by the RK-sixth order algorithm is better than that obtained by the RK-fifth order algorithm. As speed is one of the major concern in the simulation, determining the maximum step size that still yields convergence for a template can be helpful in speeding up the system.

The speed-up can be achieved by selecting an appropriate (Δt) for that particular template. Even though the maximum step size may slightly vary from one image to another, the values in figure (4) tend to be served as good references. These results were obtained by trial and error over more than 100 simulations on a Lena image. Figure(5) shows that the importance of selecting an appropriate time step size (Δt). If the step size is chosen is too small, it might take many iterations, hence longer time, to achieve convergence. But, on the other hand, if the step size taken is too large, it might not converge at all or it would be converges to erroneous steady state values; the latter remark can be observed in the case of the Euler algorithm. The results of figure 5 were obtained by simulating a small image of size 256*256 pixels using Averaging template on a Lena image. For a larger step size (Δt), the RK-sixth order algorithm takes lesser simulation time in comparison with other numerical integration algorithms namely RK-fifth order, RK-Gill and Explicit Euler.

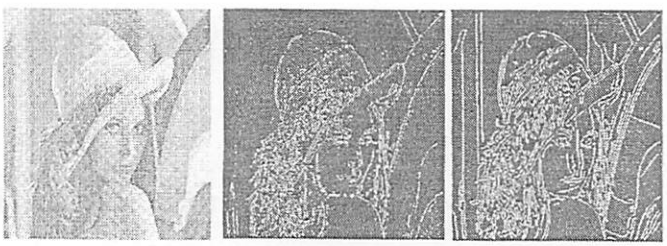

Figure 2: (a) Original Lena Image, (b) After Averaging Template, (c) After Averaging and Edge Detection Templates by employing RK-fifth order algorithm.

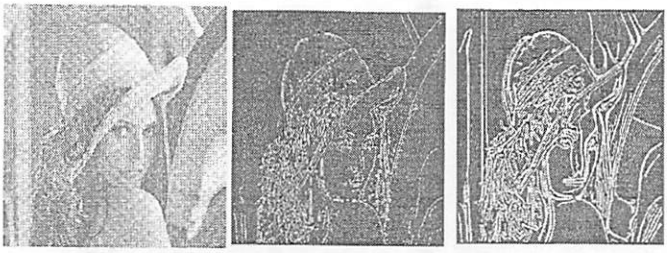

Figure 3: (a) Original Lena Image, (b) After Averaging Template, (c) After Averaging and Edge Detection Templates by employing RK- sixth order algorithm.

6 Conclusion

The present article sheds some light on different numerical integration algorithms involved in the raster CNN simulation. The importance of the simulator is capable of performing raster simulation for any kind as well as any size of input image. It is a powerful tool for researchers to investigate the potential applications of CNN. The simulator adapted in the present investigation meets the need in three ways: (1) Depending on the accuracy required for the simulation, the user can choose from five numerical integration methods (2) The input image format is the X Windows bitmap (xbm), which is commonly available and (3) The input image can be of any size, allowing simulation of images available in common practices. It is pertinent to pin-point out here that the RK-sixth order algorithm guarantees the accuracy of the detected edges and greatly reduces the impact of random noise on the detection results in comparison with the RK-fifth order algorithm. It is of interest to mention that using RK-sixth order algorithm; the edges of the output images are proved to be feasible and effective by theoretic analysis and simulation.

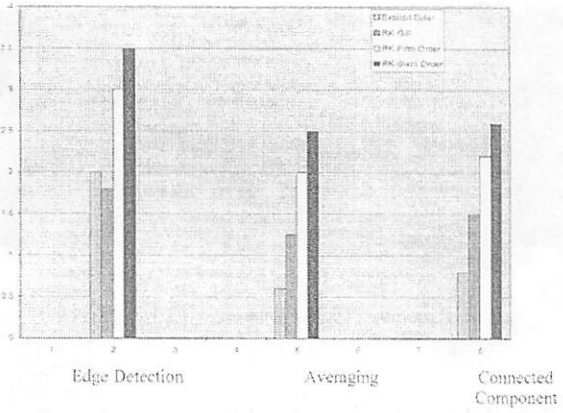

Figure 4: Maximum Step Size (Δt) yields the convergence for four different Templates .

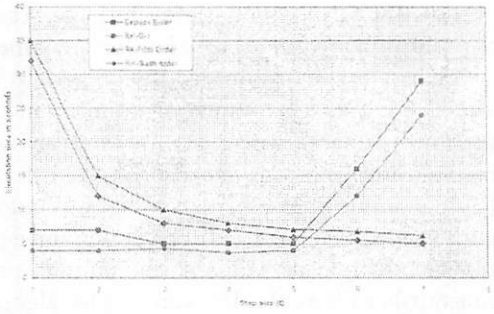

Figure 5: Comparison of Four Numerical Integration Techniques using the Averaging Template.

Acknowledgment

This research work was fully supported as a part of Technical Quality Improvement Programme [TEQIP], sponsored by Govt. of India, National Institute of Technology, Tiruchirappalli-620 015, Tamilnadu, India. Telephone No: +91-0431-2501801/10; Fax No. : +91-0431-2500133; URL:http://www.nitt.edu; The authors would like to thank Dr. V. Shanthi, Lecturer, Department of Mathematics,National Institute of Technology, Tiruchirappalli-620 015, Tamilnadu, India, for her timely help to submit this manuscript in time.

References

[1] M.Bader, A comparative study of new truncation error estimates and intrinsic accuracies of some higher order Runge-Kutta algorithms. *Computers and Chemistry*, 11: 121-124,1987.

[2] M.Bader, A new technique for the early detection of stiffness in coupled differential Equations and application to standard Runge-Kutta algorithms. *Theoretical Chemistry Accounts*, 99: 215-219, 1988.

[3] J.C.Butcher, *The Numerical Analysis of Ordinary Differential Equations.* John Wiley and Sons, U.K,2003.

[4] J.C.Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge- Kutta and General Linear Methods*, John Wiley and Sons, U.K, 1987.

[5] Chi-Chien Lee and Jose Pineda de Gyvez, Single-Layer CNN Simulator. *International Symposium on Circuits and Systems*, 6: 217-220, 1994.

[6] L.O.Chua and L. Yang, Cellular Neural Networks: Theory, *IEEE Transactions on Circuits and Systems*, 35: 1257-1272, 1988.

[7] L.O.Chua and L. Yang, Cellular Neural Networks: Applications, *IEEE Transactions on Circuits and Systems*, 35: 1273-1290, 1988.

[8] L.O. Chua, T. Roska, *Cellular Neural Networks and Visual Computing*, Cambridge University press, UK, 2002.

[9] L.O. Chua and T. Roska, The CNN Universal Machine Part 1: The Architecture, *in Int. Workshop on Cellular Neural Networks and their Applications (CNNA)*, 1-10, 1992.

[10] Leon Bottou, Patrick Haffner and Yann LeCun, Efficient convergence of digital documents to multilayer raster formats , *IEEE*, 444-448, 2001.

[11] K. Hadad and A. Piroozmand, Application of cellular neural network (CNN) method to the nuclear reactor dynamic equations , *Annals of Nuclear Energy*, 34:406-416,2007.

[12] S.C. Oliveira, Evaluation of effectiveness factor of immobilized enzymes using Runge-Kutta-Gill method, how to solve mathematical undetermination at particle center point?, *Bio Process Engineering*, 20: 185-187,1999.

[13] J.A. Nossek, G. Seiler, T. Roska, and L.O.Chua, Cellular Neural Networks: Theory and Circuit Design, *Int. J. of Circuit Theory and Applications*,20: 533-553, 1992.

[14] R.Ponalagusamy and S.Senthilkumar, Time-Multiplexing CNN Simulation Using Limiting Formulas of RK(7,8),*Research Journal of Information Technology*,Accepted for publication, 2007.

[15] W. H. Press, B. P. Flannery, S.A. Teukolsky, and W.T.Vetterling, *Numerical Recipes. The Art of Scientific Computing*, Cambridge University Press, New York, 1986.

[16] Rafael C. Gonzalez, Richard .E. Woods and Steven .L. Eddin, *Digital Image Processing using MATLAB*, Pearson Education Asia, 2005.

[17] Roska et al. *CNNM Users Guide*, Version 5.3x, Budapest, 1994.