ISSN 2347-1921

Single-Layer Raster CNN simulator using RK-Gill

M. El-Sayed Wahed,
Department of Computer Science, Faculty of Computers and information
Suez Canal University (Egypt) mewahed@yahoo.com
Wael M. Kader and Eyman Yosef
Department of Mathematics, Faculty of Sience, Zagazig University

ABSTRACT

An efficient numerical integration algorithm for single layer Raster Cellular Neural Networks (CNN) simulator is presented in this paper. The simulator is capable of performing CNN simulations for any size of input image, thus a powerful tool for researchers investigating potential applications of CNN. This paper reports an efficient algorithm exploiting the latency properties of Cellular Neural Networks along with numerical integration techniques; simulation results and comparisons are also presented.

Keywords

Single-Layer Cellular neural networks; numerical integration algorithms; Euler; Modified Euler; RK4 and RK-Gill.

Council for Innovative Research
Peer Review Research Publishing System
Journal: Journal of Advances in Mathematics

Vol 3, No 3
editor@cirworld.com
www.cirworld.com, member.cirworld.com

ISSN 2347-1921

CNN is a hybrid of *Cellular Automata* and *Neural Networks* (hence the name Cellular Neural Networks), and it shares the best features of both worlds. 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; see Fig, la, characterized by the following features [3,8].

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 identical.

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

identical.

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 *A/D* conversion).

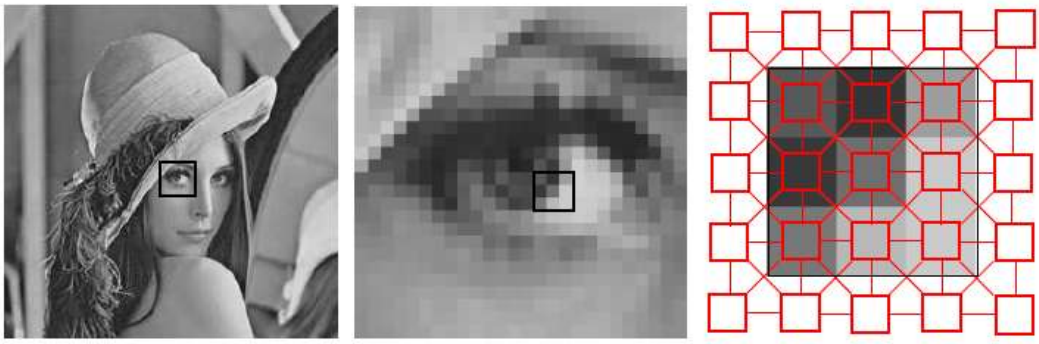

Fig. 1: CNNs are very suitable for image processing thanks to the one-to-one relationship between pixels and cells.

The basic circuit unit of CNN is called a *cell* [2,9]. It contains linear and nonlinear circuit elements. Any cell, *C(i,j),* is connected only to its neighbor cells, i.e. adjacent cells interact directly with each other. This intuitive concept is called *neighborhood* and is denoted as *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* > *U* 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:

$$C\frac{d\,x_{ij}(t)}{dt} = -\frac{1}{R}x_{ij}(t) + \sum_{C(k,l)\in N(i,j)} A(i,j;k,l)\,y_{kl}(t) + \sum_{C(k,l)\in N(i,j)} B(i,j;k,l)\,u_{kl}$$

$$Y_{ij}(t) = \frac{1}{2}\left(\left|x_{ij}(t)+1\right| - \left|x_{ij}(t)-1\right|\right) \qquad (1)$$

where x_{ij} is the state of cell *C(i,j),* $x_{ij}(0)$ is the initial condition of the cell, **C** is a linear capacitor, R is a linear resistor, ***I*** is an independent current source, $A(I,J;k,1)y_{kl}$ and $B\{i,j;k,l)u_{kl}$ are voltage controlled current Sources for all cells *C(k,l)* in the neighborhood ***N(ij)*** of cell ***C(ij),*** and y_{ij} represents the output equation.

Notice from the summation operators that each cell is affected by its neighbor cells. ***A(.)*** acts on the output of neighboring cells and is referred to as *the feedback operator.* ***B(.)*** in turn affects the input control and is referred to as the *control operator.* Specific entry values of matrices ***A(.)*** and ***B(.)*** are application dependent, are space invariant and are called *cloning templates.* A current bias **Z** and the cloning templates determine the *transient behavior* of the cellular nonlinear network. The equivalent block diagram of a continuous-time cell implementation is shown in Fig. lb.

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, but how to choose the optimal network and how to program it to perform a given task are still topics under investigation. This is the reason why there is a need for behavioral CNN simulator capable of helping investigators design and manipulate cloning templates (“programming”). Existent tools are not meant to deal

with a significant number of pixels typical in common image processing applications [5,8,9]. The simulator presented here not only satisfies this need, but it also can be used for *testing* CNN hardware implementations. Lee and Pineda de Gyvez introduced Euler, Improved Euler Predictor –corrector and Fourth-Order Runge –Kutta algorithms in Single- Layer Raster CNN simulation[1]. In this paper, we consider the same problem but by using a different approach such as Euler, RK4 and RK-Gill

2 BEHAVIORAL SIMULATION

Recall that equation (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,l)*** for all ***i,j,kl.***

Therefore, the solution of the system of difference equations can be seen as a convolution process between the image and the CNN processors. The basic approach is to imagine a square subimage area centered at ***(x,y),*** with the subimage being the same size of the templates involved in the simulation. The center of this subimage 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. An instance of this image scanning-processing is referred to as an "iteration". The processing stops when it is found that the states of all CNN processors have converged to steady-state values [3] and the outputs of its neighbor cells are saturated, e.g. they have a +1 value. This whole simulating approach is referred to as raster simulation. A simplified algorithm is presented below for this approach. The part where the integration is involved (i.e. calculation of the next state) is explained in the Numerical Integration Methods section.

Algorithm: (Single-Layer or Raster CNN simulation)

Obtain the input image, initial conditions and templates from user;

/* M,N = # of rows/columns of the image */

while (converged-cells < total # of cells) (

for (i=l; i<=M; i++)

for (j=l; j<=N; j++) (

if (convergence-flag[i] [i l)

/* calculation of the next state*/

continue; /* current cell already converged */

$$Xij(t_{n+1}) = Xij(t_n) + \int_{\tau_n}^{\tau_{n+1}} f`(x(n\tau))dn\tau$$

/* convergence criteria */

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

{

convergence-flag[i]fi] = 1;

converged-cells++ ;

}

} I* end for *I

/* update the state values of the whole image*/

for (i=l; i<=M; i++)

for (j=l; j<=N; j++) (

if (convergence-flag[i]h I) continue;

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

}

ISSN 2347-1921

#-ofjteration++;

) I* end while *I

The raster approach implies that each pixel is mapped onto a CNN processor. That is, we have an image processing function in the spatial domain that can be expressed as:

$$g(x,y) = T(f(x,y)) \quad (2)$$

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

3 Numerical Integration Methods

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 [4].

$$f'(x(n\tau)) = \frac{d\,x_{ij}(n\tau)}{d(n\tau)} = -x_{ij}(n\tau) + \sum_{C(k,l)\in N(i,j)} A(i,j;k,l)\,y_{kl}(n\tau) + \sum_{C(k,l)\in N(i,j)} B(i,j;k,l)\,u_{kl} + I \quad (3)$$

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 [7].

These methods can be derived using the definition of the definite integral:

$$X_{ij}((n+1)\tau) - X_{ij}(n\tau) = \int_{\tau_n}^{\tau_{n+1}} f'(x(n\tau))\,dn\tau \quad (4)$$

Three of the single-step numerical integration algorithms used in the CNN behavioral simulator described here. They are the Euler's algorithm, RK4, and RK-Gill algorithms.

1.Euler Method

Simplest of all algorithms for solving ODEs. It is an explicit formula which uses the Taylor-series expansion to calculate the approximation:

$$X_{ij}((n+1)\tau) = X_{ij}(n\tau) + \tau f'(x(n\tau)) \quad (5)$$

2. The Improved Euler Predictor-Corrector method

The Improved Euler Predictor-Corrector method uses both explicit (predictor) and implicit(corrector) formulae. The integral is calculated by multiplying the step size r with the averaged sum of both the derivative of x(nr) and the derivative of the predicted xp((n+l)'c) at the next time step:

$$X_{ij}((n+1)\tau) = X_{ij}(n\tau) + \tau/2[f'(x(n\tau) + f'(x_p((n+1)\tau))] \quad (6)$$

3. The Fourth-Order Runge-Kutta method

The Fourth-Order Runge-Kutta method is the most costly among the three methods in terms of computation time, as it requires four derivative evaluations per time step. However, its high cost is compensated by its accuracy in transient behavior analysis.

$$X_{ij}((n+1)\tau) = X_{ij}(n\tau) + \frac{k_1^{ij} + k_2^{ij} + k_3^{ij} + k_4^{ij}}{6} \quad (7)$$

Where $\forall c(l,m) \in N_r(i,j)$:

$$k_1^{ij} = \tau f'(x_{ij}(n\tau))$$

$$k_2^{ij} = \tau f'(x_{ij}(n\tau) + \frac{1}{2} k_1^{ij})$$

$$k_3^{ij} = \tau f'(x_{ij}(n\tau) + \frac{1}{2} k_2^{ij})$$

ISSN 2347-1921

$$k_4^{ij} = \tau f'(x_{ij}(n\tau) + k_3^{ij})$$

4. RK-Gill

RK-Gill [6] is formulated as

$$X_{ij}((n+1)\tau) = X_{ij}(n\tau) + \frac{1}{6}\left[k_1 + \left(2-\sqrt{2}\right)k_2 + \left(2+\sqrt{2}\right)k_3 + k_4\right] \quad (8)$$

Where

$$k_1 = h\, f'(x_{ij}(n\tau))$$

$$k_2 = h\, f'(x_{ij}(n\tau) + \frac{1}{2}k_1)$$

$$k_3 = f'(x_{ij}(n\tau) + \left(\frac{1}{\sqrt{2}} - \frac{1}{2}\right)k_1 + \left(1 - \frac{1}{\sqrt{2}}\right)k_2)$$

$$k_4 = f'(x_{ij}(n\tau) - \frac{1}{\sqrt{2}}k_2 + \left(1 + \frac{1}{\sqrt{2}}\right)k_3)$$

And f(.) is computed according to (1). There are many single-step methods available to us for this purpose. But, one option worth considering is the combination of two methods in solving for the solution. So we use RKEHM to make a very efficient computer simulation method for solving the problem.

4 Simulation Results and Comparisons

The simulation time used for comparisons is the actual CPU time used. The input image format for this simulator is a JPEG. format.

(a)

(b)

Fig.2. Image processing (a) After Averaging Template (b) After Averaging and Edge Detection

Fig. 2 shows results of the raster simulator obtained from a complex image of 65,536(256x256) pixels. For this example an Averaging template followed by an Edge Detection template were applied to the original image to yield the images displayed in Figs. 2a and 2b, respectively.

Also in figure 3, it has been shown the quality measures of the two pictures in 2a and 2b by using the numerical techniques Euler, Modified Euler, RK4 and RK-Gill.

ISSN 2347-1921

Method	Mean Square Error	Peak Signal to Noise Ratio	MNormalized Cross-Correlation	Average Difference	Structural Content	Maximum Difference	Normalized Absolute Error
Euler	1.7770e+003	11.4031	0.9245	9.1441	1.0057	245	0.0800
Modified Euler)	1.7200e+003	10.4031	0.9105	8.1001	1.0010	240	0.0700
RK4	1.7100e+003	9.4031	0.9000	7.1000	1.0000	238	0.0500
RK-Gill	1.7000e+003	9.4030	0.8900	7.0000	0.9000	237	0.0400

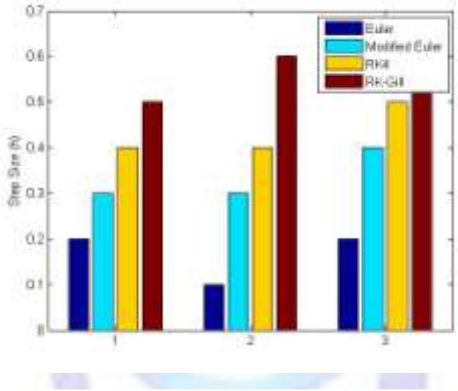

Fig.3. Quality measures of fig.2a and fig.2b

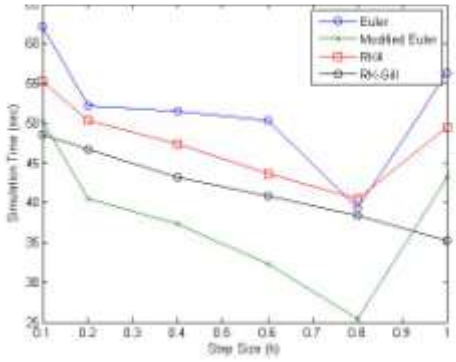

Fig.4. Simulation time comparisons for 4 different numerical techniques

Since speed is one of the main concerns in the simulation, finding 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 Fig.5 still serve as good references. These results were obtained by trial and error over more than 100 simulations On Lena image with small size 43x64(2752 pixels).The importance of selecting an appropriate Δt can be easily visualized in Fig. 4. If the step size chosen is too small, it might take many iterations, hence longer time, to achieve convergence. On the other hand, if the step size taken is too large, it might not converge at all or it would converge to erroneous steady state values. The results of Fig. 5 were obtained by simulating Lena image of size 43x64(2752 pixels) using an Edge detection template.

5 CONCLUSION

As researchers are coming up with more and more CNN

applications, an efficient and powerful simulator is needed.

The simulator hereby presented meets the need in three ways:

1 Depending on the accuracy required for the simulation, the user can choose from three numerical methods to perform the numerical integration,

2 The input image format is a JPEG image which is commonly available.

3 The input image can be of any size, allowing simulation of images available in common practices.

4 In Comparing to [1], we find that we used smaller step size range from 0 to 1 instead of 1 to 5 in [1].

5 Also, we added the quality features which is not existed in [1]. And the simulation time for our methods is better than those in [1].

REFERENCES:

[1] C.C.Lee and Jose de Gyvez (1994) Single-Layer CNN Simulator.

[2] L. 0. Chua and L. Yang(1988) Cellular Neural Networks: Theory & Applications, IEEE *Trans. Circuits and Systems,* Vol. CAS-35, pp. 1257-1290.(2)

[3] L.O. Chua and T. Roska(1992) The CNN Universal Machine Part 1: The Architecture, in Int. Workshop on Cellular Neural Networks and their Applications (CNNA), pp. 1-10.(3)

[4] J. A. Nossek, G. Seiler, T. Roska and L. 0. Chua (1992.) Cellular Neural Networks: Theory and Circuit Design, International Journal of Circuit Theory and Applications, Vol. 20, pp. 533-553.(5)

[5] J. Varrientos and E. Sanchez-Sinencio(1992) CELLSIM: A cellular neural network simulator for the personal computer, in Proc. 35th Midwest Symp. Circuits Systs, pp. 1384-1387.

[6] R.Ponalagusamy and S.Senthilkumar(2008), A comparison of RK- fourth orders of variety of means on multilayer raster CNN simulation Trends in Applied Science and Research,3(3), 242-252.

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

[8] O.H. ABDELWAHED, M. EL-SAYED WAHED and Eyman Yosef " Optimizing Time- multiplexing Raster Cellular Neural Network simulator using genetic algorithms with The RK-Embedded Centroidal Mean(RKECM) International Journal of Scientific & Engineering Research, Volume 4, Issue 6, June-2013.

[9] M. EL-SAYED WAHED and O.H. ABDELWAHED " An efficient numerical integration algorithm for single-layer Raster cellular neural networks simulator **"** International Journal of Physical Sciences Vol. 7(47), pp. 6144-6148, 16 December, 2012.