Optimizing Time-multiplexing Raster Cellular Neural Network simulator using genetic algorithms with RK4(2),RK4(3) and RK 6(4)

O.H. Abdelwahed, M. El-Sayed Wahed and O. Mohamed Eldaken
Department of Computer Science
Faculty of Computers and Information
Suez Canal University
EGYPT
E-mail: mewahed@yahoo.com, osama.abdelwahed@gmail.com

Abstract - This paper presents proper CNN templates for edge detection in image processing. Training of CNN is done by using genetic algorithm. In this research, a versatile algorithm for simulating CNN arrays and time multiplexing is implemented using numerical integration algorithms; RK4(2), RK4(3) and RK6(4). The approach, time-multiplexing simulation, plays a pivotal role in the area of simulating hardware models and testing hardware implementations of CNN. Owing to hardware limitations in practical sense, it is not possible to have a one-one mapping between the CNN hardware processors and all the pixels of the image. 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.

Key-words:- time multiplexing Cellular neural networks, genetic algorithms, RK4(2), RK4(3),and RK6(4)

1 Introduction

Most of the widely applied genetic cellular neural networks fall into two main classes: (1) memoryless cellular neural networks and (2) dynamical cellular neural networks. As in Hopfield networks (HN) and CNN, dynamical neural networks have usually been designed as dynamical systems where the inputs are set to some constant values and each trajectory approaches one of the stable equilibrium points depending upon the initial state. Cellular Neural Network is a large-scale non-linear analog circuits which processes signals in real time[2]. The network behavior of CNN depends on the initial state of the cells activation, namely bias I and on weights values of A and B matrices which are associated with the connections inside the well-defined neighborhood of each cell. There are many approaches in estimation of A,B, I matrices. Here we prefer genetic algorithm. Genetic algorithm is a learning algorithm based on the mechanism of natural selection and genetics, which have proved to be effective in a number of applications. It works with a binary coding of the parameter set, searches from a number of points of the parameter space. It uses only the cost function during the optimization, it need not derivatives of the cost function or other information. [16-22].

In this paper, we focus on edge detection which is much significant in various application as virtual reality, intelligent human-computer interface and TV-conference, security system.[18,19,20]. First step in edge detection is locating face and facial features [18],[19]. Then, the detected edges have to

be normalized and recognized by specially designed classifier. In this study, we find the facial features and face region using CNN templates estimated by using genetic algorithms.

2 Cellular Neural Networks

Like cellular automata, the CNN is made of a massive aggregate of regularly spaced circuits clones, called cells, which communicate with each other directly only through nearest neighbors. In Figure 2, each cell is modeled as squares. The adjacent cells can interact directly with each other. Cells not directly connected together may affect each other indirectly because of the propagation effects of the continuous-time dynamics of cellular neural networks. An example of a two-dimensional CNN is shown in Figure 3. Now let us define the neighborhood of C(i,j).

Definition : r-neighbourhood

The r-neighbourhood of a cell C(i,j) , in a cellular neural network is defined by,

$$N_r(i,j) = \{C(k,l) \mid \max\{|k-i|, |l-j|\} \le r\}$$
$$1 \le k \le M; 1 \le l \le N \qquad (1)$$

where r is a positive integer number.

Figure 3 shows neighborhoods of the C(i,j) cell (located at the center and shaded) with r=1 and 2,respectively. To show neighbourhood relations more clearly, the center pixel is coloured as black and related pixels in brown in Figure 3 and 4. Cells are multiple input-single output nonlinear processors all described by one, or one among several different, parametric functionals. A cell is characterized by a state variable, that is generally not observable as such outside the cell itself. It contains linear and non-linear circuit elements such as linear resistors, capacitors and non-linear controlled sources (Figures 4 and 5).

Every cell is connected to other cells within a neighborhood of itself. In this scheme, information is only exchanged between neighbouring neurons and this local information characteristic does not prevent the capability of obtaining global processing. The CNN is a dynamical system operating in continuous or discrete time. A general form of the cell dynamical equations may be stated as follows:

$$C\frac{d x_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{kl \in N_r(ij)} A_{(i-k)(j-l)}(t) y_{kl} + \sum_{kl \in N_r(ij)} B_{(i-k)(j-l)}(t) u_{kl} + I$$
$$y_{ij}(t) = \frac{1}{2}\left(|x_{ij}(t)+1| - |x_{ij}(t)-1|\right) \qquad (2)$$

where x,y,u,I denote respectively cell state, output, input, bias and j and k are cell indices. CNN parameter values are assumed to be spaced-invariant and the nonlinear function is chosen as piece-wise linear. Since we use discrete 2-D images, Equation (2) is rewritten as,

$$x_{ij}(n+1) = -x_{ij}(n) + \sum_{kl \in N_r(ij)} A_{(i-k)(j-l)}(t)(n) y_{kl} + \sum_{kl \in N_r(ij)} B_{(i-k)(j-l)}(t) u_{kl} + I$$
$$y_{ij}(n) = \frac{1}{2}\left(|x_{ij}(n)+1| - |x_{ij}(n)-1|\right) \qquad (3)$$

with A, B and I being cloning template matrices that are identically repeated in the neighbourhood of every neuron as,

$$A = \begin{pmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ a_{0,-1} & a_{0,0} & a_{0,-1} \\ a_{1,-1} & a_{1,0} & a_{-1,1} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,-1} \\ b_{1,-1} & a_{1,0} & b_{-1,1} \end{pmatrix}, I \quad (4)$$

The network behaviour of CNN depends on the initial state of the cells activation, namely bias I and on weights values of A and B matrices which are associated with the connections inside the well-defined neighbourhood of each cell. CNN's are arrays of locally and regularly interconnected neurons, or, cells, whose global functionality are defined by a small number of parameters (A,B, I) that specify the operation of the component cells as well as the connection weights between them. CNN can also be considered as a nonlinear convolution with the template. Cells can be characterized by a functional block diagram that is typical of neural network theory: Figure 4 depicts a two-stage functional block diagram of a cell, composed of a generalized weighted sum (in general nonlinear with memory) integration, output nonlinear function/functional. Data can be fed to the CNN through two different ports: initial conditions of the state and proper input u. Bias values I may be used as a third port.

The network behaviour of CNN depends on the initial state of the cells activation, namely bias I and on weights values of A and B matrices which are associated with the connections inside the well-defined neighbourhood of each cell. CNN's are arrays of locally and regularly interconnected neurons, or, cells, whose global functionality are defined by a small number of parameters (A,B, I) that specify the operation of the component cells as well as the connection weights between them. CNN can also be considered as a nonlinear convolution with the template. Since the introduction of Chua [2], CNN has attracted a lot of attention. Not only from a theoretical point of view these systems have a number of attractive properties, but furthermore there are many well-known applications like image processing, motion detection, pattern recognition, simulation. The reduced number of connections within a local neighbourhood, the principle of cloning template etc., turn out to be advantage of CNN's.

3 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[2] 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.

In the following two subsections we will discuss genetic algorithms and edge detection by CNN and genetic algorithms.

3.1 Genetic Algorithms

In the estimation of A.B. and I matrices of CNN, we use genetic algorithms. Genetic algorithm is a learning algorithm based on the mechanism of natural selection and genetics, which have proved to be effective in a number of applications. It works with a binary coding of the parameter set, searches from a number of points of the parameter space. It uses only the cost function during the optimization, it need not derivatives of the cost function or other information. [11,12]. Processes of natural selection cause chromosomes that encode successful structures to reproduce more often than those that do not. In addition to reproduction, mutations may cause the chromosomes of children to be different from those of their biological parents, and crossing over processes create different chromosomes in children by changing the some parts of the parent chromosomes between each other. Like nature, genetic algorithms solve the problem of finding good chromosomes by manipulating in the chromosomes blindly without any knowledge about the problem they are solving.[12]. The underlying principles of GA were first published by Holland in1962, [13]. The mathematical framework was developed in the 1960s and is presented in his pioneering book in 1975 [14]. In optimization applications, they have been used in many diverse fields such as function optimization, image processing, the traveling salesperson problem, system identification and control. A high-level description of GA has been done by Davis in 1991 as follows. [15] Given a way or a method of encoding solutions of problem into the form of chromosomes and given an evaluation function that returns a measurement of the cost value of the following steps:

Step1: Initialize a population of chromosomes

Step2: Evaluate each chromosomes in the population.

Step3: Create new chromosomes by mating current chromosomes; apply mutation and recombination as the parent chromosomes mate.

Step4: Delete members of the population to make room for new chromosomes.

Step5: Evaluate the new chromosomes and insert them into the population.

Step6: If the stopping criterion is satisfied, then stop and return the best chromosome; otherwise, go to step3

3.2 Edge detection by CNN and genetic algorithms

In this work, proper CNN templates are described to locate edge detection in image by using genetic algorithms. For this aim, CNN templates are designed so that they satisfy the stability. So, A and B templates are selected as symmetric. Because of selecting size of templates as 3*3, totally 11 template parameters are searched. One of these parameters is offset, five of them belongs the A matrix, and the other five parameters belongs the B matrix. Each parameter are encoded by 16 bits in chromosomes. So, the length of chromosomes has been selected as 176 bits. In training process, 72 chromosomes are constructed as initial population randomly. The number of population is kept constant as 72 during the algorithm. Mutation probability m p has been set % 1. Training process includes these steps as follows.

(i).Construct initial population; A matrix is constructed called as population matrix. Each row of the population matrix represents chromosomes. Because of selecting number of chromosomes is 72, there are 72 rows in population matrix. Number of columns of this matrix is 176, because there are 176 bits in each chromosome. At the beginning this matrix is constructed randomly.

(ii). Extract the CNN template: Chromosomes represents the binary codes of the elements of the CNN template A,B, I . In this step, each chromosomes are decoded the elements of the CNN

are computed in [-8,8] interval. Since each element is coded as 16 bits, each parameter can take 216 different value in [-8,8] interval. In each chromosomes first 11 bits represents first bits of the template elements. And second 11 bits of chromosomes represents the second bits of the template elements and so on. These elements are

$$S = \lfloor A_{11}, A_{12}, A_{13}, A_{21}, A_{22}, B_{11}, B_{12}, B_{13} B_{21} B_{22,} I \rfloor \quad (5)$$

(iii). Evaluate cost function value for each chromosomes; In this step, an image which was selected as training image is given as input to CNN. Normally in this gray-level image, brightness varies in 0 (black) through 1(white) interval. To fit this image to CNN operation, brightness of the image is converted from [0,1] to [-1,1]. According the same rule, brightness of the CNN output image is converted from [-1,1] to [0,1].Then CNN works with templates belonging with first chromosome. After the CNN output appears as stable, cost function is computed between this output image and target image which we want to obtain. This process is repeated with template sets belongs each chromosomes in the population. Cost function has been selected in this study as follow

$$Cost(A, B, I) = \sum_{i}^{M} \sum_{j}^{N} P_{i,j} \oplus T_{i,j} \quad (6).$$

where A,B, I represents CNN templates, m, n represents number of pixels of the image, P and T represent input and target image, respectively, notation $\oplus$ represents XOR operation between each elements of the P and T .

After the finding the cost function, fitness function is evaluated for each chromosome according this rule;

fitness(A, B, I) = m* n - cost(A, B, I) (7).

Another definition has been defined for stopping criterion as follows;

stcriterion = 0.99 *m* n (8).

where m represents the number of rows of the image matrix and n represents the number of the columns of the image matrix. If the maximum fitness value of the chromosomes is greater than stop criterion, algorithms is stopped and the chromosome whose fitness value is the maximum fitness in the population is selected. The templates which has been extracted from this selected chromosomes are the most proper the templates which satisfy the task we wanted to realize.

(iv). Creating new generation; Before creating next generation, fitness values of the population are sorted by descendent order. And all of the fitness values are normalized related to the sum of the fitness values of the population. A random number r between 0 and 1 is generated. Then the first population member is selected whose normalized fitness, added to normalized fitnesses of the proceedings population members, is greater than or equal to r . This operation is repeated 72 times. So, the chromosomes whose fitness are bad are deleted from the population. This procedure mentioned above is called reproduction process in genetic algorithms. Reproduction process does not generate new chromosomes. It elect the best chromosomes in the population and increases the number of the chromosomes whose fitness values are relatively greater than the others.

After the reproduction 36 pairs of chromosomes are selected as parents randomly. Two numbers s1, s2 between 1 and length of chromosomes,176 are generated. The bit strings between s1 and s2 are called crossover site. During the crossing over process, bit strings in crossover site in each pair of chromosomes are interchanged. Then two new chromosomes are created from a pair of old chromosomes. At the last, 72 new chromosomes which are called children are generated to build new population. Over these chromosomes, mutation operation is processed. Since mutation probability has been set to %1, 126 bits are selected randomly in the population and they are inverted. And the chromosome whose fitness value was the best before the reproduction process is added instead of

deleted a chromosome which randomly selected in the population that was obtained after the mutation process population to save the best chromosome. This new population is the next generation population. After the obtaining the new generation searching procedure goes to second step and goes on until the stopping criterion was happened in the second step.

3.3 Time multiplexing simulation approach

In this procedure it is possible to define a block of CNN processors which will process a subimage whose number of pixels is equal to the number of CNN processors in the block. The processing within this subimage follows the raster approach adapted in Chua and Yang (1988b). Once convergence is achieved, a new subimage is processed. The same approach is being carried out until the whole image has been scanned. It is clear that with this approach the hardware implementation becomes feasible since the number of CNN processors is finite. Also, the entire image is scanned only once since each block is allowed to fully converge. An important point is to be noticed that the processed border pixels in each subimage may have incorrect values since they are processed without neighboring information only local interactions are important for the latency of CNNs. To overcome the aforementioned problem two sufficient conditions must be considered while performing time-multiplexing simulation. Alternatively, to ensure that each border cell properly interacts with its neighbors it is necessary to have the following. (1) To have a belt of pixels from the original image around the subimage and (2) to have pixel overlaps between adjacent subimages.

It is possible to quantize the processing error of any border cell Cij with neighborhood radius of 1. By computing independently the error owing to the feed forward operator and interaction among cells for the two horizontally adjacent processing blocks, the absoulte processing error owed only to the effect of the B template is obtained by subtracting the erroneous state value from the error free states using Eq. 1. This gives, $\varepsilon_{ij}^{\ B} = \sum_{i=1}^{i=3} b_{i,j+1} sign(u_{i,j+!})$ (9)

Where

$b_{i,j+1}$ = The missing entries from the B template due to the absence of input signals $u_{i,j+1}$

$sign(.) = The\ sign\ \ function$

The latter function is used to represent the status of a pixel, e.g. black = 1 and white = -1. It is seen that the error is both image and template dependent. Alternatively, the steady state of a border cell may converge to an incorrect value due to the absence of its neighbors weighted input. Given the local interconnectivity properties of CNN, one can conclude that the minimum width of the input belt of pixels is equal to the neighborhood radius of the CNN. Interaction between Cells in view of interaction between cells, it is possible to compute the absolute error in a similar way. In absence of the B template for a moment, the error is expressed as:

$$\varepsilon_{ij}^{\ A} = \sum_{i=1}^{i=3} b_{i,j+1} y_{i,j+1}(t) \qquad (10)$$

Where

$a_{i,j+1}$ = The missing entries from the A template due to the absence of input signals $y_{i,j+1}(t)$

In this case output signals depend on the state of their corresponding cells. The technique of an overlap of pixels between two adjacent blocks is proposed in order to minimize the error. The minimum overlap width must be proportional to two times of the neighborhood`s radius of the CNN. The time-multiplexing procedure deals with iterating

each block (subimage) until all CNN cells within the block converge. The block with converged cells will have state variables x which are the values used for the final output image shown in Fig. 1, the converged values from Block i are taken by the left side of the overlapped cells and the right side from Blocki+1. Further, the initial conditions for the border cells of Block i+1 are the state values obtained while processing Block i-1. In the simulator used here, the number of overlapping columns or rows between the adjacent blocks is defined by the user. Suppose if higher number of overlapping is obtained in columns or rows then it indicates the more accurate simulation of neighboring effects on the border cells.

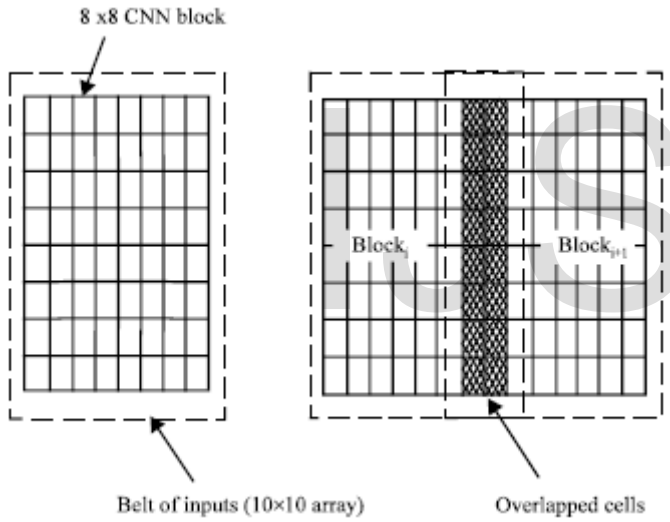


Fig. 1: CNN multiplexing with overlapped cells

In case of practical applications the correct final state is of more importance than the transient states, an overlap of two is usually sufficient. An even number of overlapping cells is recommended, because the converged cells in the overlapped region can be evenly divided by the two adjacent blocks. Having the added overlapping feature, better neighboring interactions are achieved, but at the same time, an increase in computation time is unavoidable. On the other hand, by taking advantage of the fact that the original input image is been divided into small CNN subimages, the chance of a subimage having all its pixels black or white is high. This is another feature that can be added to the time multiplexing simulation to improve computation times. The savings in simulation time come from avoiding repetitive simulations of all-black and all-white subimages. The notion behind this timesaving scheme is that when the very first all-black/all-white block is encountered, after processing that block, the final states of the block are stored separately from the whole image. When subsequent all-black/all-white blocks are found, there is no need to simulate these blocks since the converged states are readily available in memory, which in turn leads to avoiding the most time consuming part of the simulation which is the numerical integration. The overall idea of this simulation approach is given below in the form of program fragment.

Program Fragment for Time-Multiplexing CNN Simulation To understand the overall concept of overlap and belt approaches and raster simulation, the simplified version of algorithm with simulated annealing is given:

Algorithm: (Time-Multiplexing CNN simulation with genetic algorithm)
Obtain the input image, initial conditions and templates from user;
/* M,N = # of rows/columns of the image */
/* apply genetic algorithm in section 3.1*/
/* Use the optimized parameters from the algorithm */
B= {C_{ij} = 1,..., block_x and j = 1,..., block_y}P C S = set of border cells (lower left corner) overlap = number of cell overlaps;
belt = width of input belt M = number of rows of the image N = number of columns of the image
for (i=0; i < M; i += block_x - overlap)

for (j=0; j < N; j += block_y - overlap)
/* load initial conditions for the cells in the block except for those in the borders */
for (p=-belt; p < block._x + belt; p++)
for (q=-belt; q < block_y + belt; q++) {

$x_{i+p,\ j+q}(t)=$

$$\begin{cases} u_{ij} \\ 1 \quad \forall C_{i+pj+q} \in B \\ -1 \end{cases}$$

}/* end for */
/* if the block is all white or black don't process it */

$$if\ (x_{i+pj+q} = -1 \vee x_{i+pj+q} = 1 \forall C_{i+pj+q})$$

{
obtain the final states from memory;
continue;
}
do { /* normal raster simulation */
for (p=O; p < block_x; p++) (
for (q=O; q < block_y; q++)
/* calculation of the next state excluding the belt of inputs */

$x_{i+pj+q}(t_{n+1}) = x_{i+pj+q}(t_{n+1}) +$

$$\int_{t_n}^{t_{n+1}} f'(\mathrm{xi} + \mathrm{p}\ \mathrm{j} + \mathrm{q}\ (t_n))dt \forall C_{i+pj+q} \subset B$$

/* convergence criteria */

$$If\left(\frac{dx_{i+p,j+q}(t_n)}{dt} = 0\ and\ y_{k/} = \pm 1\ c(k,l) \in N_r(i+p, j+q)\right)$$

converged_cells++;
}/* end for */
/* update state values */

$x_{i+pj+q}(t_{n+1}) = x_{i+pj+q}(t_n)$

while(converged_cels < (block_x *block_y));
/* store new state values excluding the ones corresponding to the border cells */

$$A \leftarrow x_{ij}\ \forall C_{ij} \in B \setminus P$$

}/* end for */

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:

$$g(x,y) = T(f(x,y)) \tag{9}$$

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

4 Numerical Integration Methods

four of the single-step numerical integration algorithms used in the CNN behavioral simulator described here. They are Euler, RK4(2), RK4(3) and RK6(4).

4.1 The Proposed Methods

R.Ponalagusami and S.Senthilkumar introduced Time-Multiplexing CNN using Limiting Formula RK (7,8) (R.Ponalagusami and S.Senthilkumar 2008). In this paper, we consider the same problem but by using different approaches RK4(2) ,RK4(3) and RK6(4).

4.1.1 Stepsize selection algorithm.

There are currently two widely used methods that have appeared in the literature for changing the stepsize of p (q)-order RK codes. The first is to apply the formula (see [9])

$$h_{n+1} = f_1\left(\frac{TOL}{EST_n}\right)^{\frac{1}{q+1}}, \tag{10}$$

Where f1 is a safety factor and the new sought-after stepsize $h_{n+1} = x_{n+1} - x_n$ is predicted in terms of an

estimate of the local error ESTn which is based on the approximation

$$EST_n \approx y_n - \hat{y}_n, \qquad (11)$$

Assuming $y_n, \hat{y}_n$ to be the pth-, qth-order approximate solutions, respectively, at the previous grid point xn and TOL the requested tolerance. If

$$EST_n \le TOL, \qquad (12)$$

Then the computed solution yn+1 is accepted and the integration is carried out, otherwise(5) is reevaluated by substituting

$$EST_n \rightarrow EST_{n+1} \qquad (13)$$

This methodology is termed the error per step (EPS) mode (see Shampine [10]).

An alternative is to consider the same algorithm (5), but to use, instead of (6), the approximation

$$EST_n \cong \frac{y_n - \hat{y}_n}{h_n}, \qquad (14)$$

This is called error per unit step (EPUS) [10].

4.1.2 RK4(2) and RK4(3) at n = 4

According to [8], The equations of RK4(2) and RK4(3) are:

$$k^1_{ij} = \tau f'(x_{ij}(n\tau))$$

$$k^2_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{5}{14} k^1_{ij}$$

$$k^3_{ij} = \tau f'(x_{ij}(n\tau)) - \frac{52}{605} k^1_{ij} + \frac{819}{1210} k^2_{ij}$$

$$k^4_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{2576}{4745} k^1_{ij} - \frac{252}{365} k^2_{ij} + \frac{1089}{949} k^3_{ij}$$

$$k^5_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{19}{130} k^1_{ij} + \frac{343}{1215} k^2_{ij} + \frac{1331}{3159} k^3_{ij} + \frac{73}{486} k^4_{ij}$$

Therefore, the final integration is a weighted sum of the five calculated derivates is given:

$$x_{ij}((n+1)\tau) = x_{ij}(n\tau) + \frac{19}{130} k^1_{ij} + \frac{203}{990} k^2_{ij} + \frac{1331}{3159} k^3_{ij} + \frac{73}{486} k^4_{ij} \qquad (15)$$

The difference between Rk4(2) and RK4(3) is the local truncation error in the case of RK4(2) is given by using the RK(2)i.e.

$$y_{ij}((n+1)\tau) = y_{ij}(n\tau) + \frac{4}{55} k^1_{ij} + \frac{343}{1215} k^2_{ij} + \frac{13}{18} k^3_{ij} \qquad (16)$$

But local truncation error in the case of RK4(3) is given by using the RK(3)i.e.

$$y_{ij}((n+1)\tau) = y_{ij}(n\tau) + \frac{11}{130} k^1_{ij} + \frac{637}{1215} k^2_{ij} + \frac{605}{3159} k^3_{ij} - \frac{73}{1215} k^4_{j} \qquad (17)$$

4.1.3 RK6(4) at n= 6

According to [8], The equations of RK6(4) are:

$$k^1_{ij} = \tau f'(x_{ij}(n\tau))$$

$$k^2_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{4}{27} k^1_{ij}$$

$$k^3_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{1}{18} k^1_{ij} + \frac{1}{6} k^2_{ij}$$

$$k^4_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{66}{343} k^1_{ij} - \frac{727}{1372} k^2_{ij} + \frac{1053}{1372} k^3_{ij}$$

$$K^5_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{13339}{49152} k^1_{ij} - \frac{4617}{16384} k^2_{ij} + \frac{5427}{53248} k^3_{ij} + \frac{95207}{159744} k^4_{ij}$$

$$K^6_{ij} = \tau f'(x_{ij}(n\tau)) - \frac{6935}{57122} K^1_{ij} + \frac{23085}{48334} k K^2_{ij} + \frac{333633360}{273642941} K^3_{ij} + \frac{972160}{118442467} K^4_{ij} + \frac{172687360333633360}{610434253} K^5_{ij}$$

$$K^6_{ij} = \tau f'(x_{ij}(n\tau)) + \frac{611}{1891} K^1_{ij} - \frac{4617}{7564} K^2_{ij} + \frac{6041007}{13176488} K^3_{ij} + \frac{12708836}{22100117} K^4_{ij} - \frac{3584000}{62461621} K^5_{ij} + \frac{6597591}{7972456} K^6_{ij}$$

Therefore, the final integration is a weighted sum of the seven calculated derivates is given:

$$x_{ij}((n+1)\tau) = x_{ij}(n\tau) + \frac{131}{1800} K^1_{ij} + \frac{2031121931}{392080} K^3_{ij} + \frac{319333}{1682928} K^4_{ij} + \frac{262144}{2477325} K^5_{ij} + \frac{4084223}{15177600} K^6_{ij} + \frac{1891}{25200} K^7_{ij} \quad (18)$$

Where f(l.) 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 Rk6(4) to make a very efficient computer solving the problem the way it evaluates the integral presented.

5 Simulation Results and Comparisons

Fig.3. Image processing (a) After Averaging Template (b) After Averaging and Edge Detection

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

Fig. 3 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. 3a and 3b, respectively.

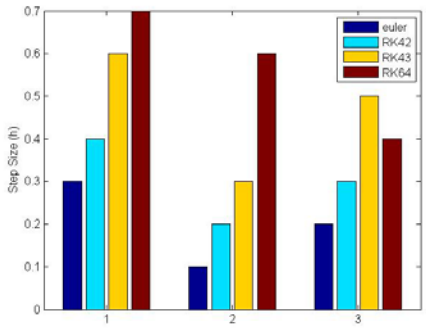
Also in figure 3, it has been shown the quality measures of the two pictures in 2a and 2b by using the numerical techniques RK4(2),RK4(3) and RK6(4) using simulated annealing. We notice that these results are better than those in the literature.

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

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

method	Mean Square Error	Peak Signal to Noise Ratio	MNormalized Cross-Correlation	Average Difference	Structural Content	Maximum Difference	Normalized Absolute Error
RK4(2)	1.6000e+003	14.4032	0.9745	5.4453	1.2237	223	0.0260
RK4(3)	1.6350e+003	13.4030	0.9605	5.2236	1.3108	227	0.2200
RK6(4)	1.6100e+003	11.4033	1.1100	5.0000	0.92200	228	0.2300

template. Even though the maximum step size may slightly vary from one image to another, the values in Fig.4 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. 5. 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. 4 were obtained by simulating Lena image of size 43x64(2752 pixels) using an Edge detection template. We notice that the CPU time for our method is better than those in the literature.



Edge Detection Averaging Connected Component

Fig.4. Maximum step size that still yield convergence for 4 different templates

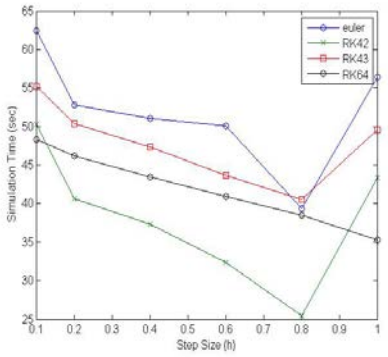


Fig.5. Simulation time comparisons for 4 different numerical techniques for four different templates

As researchers are coming up with more and more CNN applications, an efficient and powerful simulator is needed. So we use simulated annealing in optimizing CNN using the numerical integrations, especially using RK4(2),RK4(3) and RK6(4) for more efficiency with genetic algorithms. The simulator hereby presented meets the need in six 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 JPEG, which is commonly available, 3) The input image can be of any size, allowing simulation of images available in common practices, 4) CPU time of our methods is better than those in the literature, 5), the quality measures of the pictures and the edge detection for our method is better than those in the literature

References

[1] R.Ponalagusami and S.Senthilkumar(2008). "Time-Multiplexing CNN Simulation using Limiting Formula RK(7,8)".Research Journal of Information Technology 1(1):1-16.

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

[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.

[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] 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] 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.

[7] P.J.M. van Laarhoven and E.H.L. Aarts, (1987) Simulated Annealing: Theory and Application.
ISBN 90-277-2513-6

[8] Ch. Tsitouras and S. N. Papakostas(1991) "Cheap Error methods for Runge-Kutta methods ", SIAM J. SCI. COMPUT, Society for Industrial and Applied Mathematics, Vol. 20, No. 6, pp. 2067-2088.

[9] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick(1972), Comparing numerical methods for ordinary di_erential equations, SIAM J. Numer. Anal., 9 , pp. 603{637.

[10] L. F. Shampine(1986), Some practical Runge-Kutta formulas, Math. Comp., 46 , pp. 135{150.

[11] T. Sziranyi, M. Csapodi, '' Texture Classification and Segmentation by Cellular Neural Network Using Genetic Learning'', Research Report ,Budapest,Hungary, November,1994

[12] C.T. Lin, C.S. George Lee '' Neural Fuzzy Systems '' , Prentice-Hall Inc., New Jersey, 1995.

[13] J.H. Holland, ''Outline for a logical theory of adaptive systems.'', J. Assoc. Computing. Mach. 3:297-314,

[14] J.H. Holland, ''Adaptation in neural and artificial systems'', Ann Arbor, MI: University of the Michigan Press, 1975

[15] L. Davis,''Handbook of Genetic Algorithms'' New York:Van Nostrand, Reinhold,1991.
1988

[16] T. Kozek, T. Roska, L.O. Chua, :''Genetic Algorithms for CNN template Learning'', IEEE Trans. On Circuit and Systems, Vol.40, No.6 pp. 392-402, 1988

[17] T. Sziranyi, M. Csapodi, '' Texture Classification and Segmentation by Cellular Neural Network Using Genetic Learning'', Research Report ,Budapest,Hungary, November,1994

[18] J.Hu, H.Yan, M. Sakalli, ''Locating head and face boundaries for head-shoulder images'', Pattern Recognition 32 (1999) 1317-
1333

[19] A. Samal, P.A. Iyengar, ''Automatic recognition and analysis of human faces and facial expressions : a survey'', Pattern Recognition 25 (1) (1992) 65-77

[20] R. Chellappa, C.L. Wilson, S. Sirohey, ''Human and machine recognition faces: a survey'', Proc. IEEE 83 (5) (1995) 705-740.

[21] S.H. Jeng, H.Y.M. Liao, C.C.Han, M.Y. Chern, Y.T.Liu ‘‘Facial feature dedection using geometrical face model: an efficient approach’’

[22] C.T. Lin, C.S. George Lee ‘‘ Neural Fuzzy Systems ’’ , Prentice-Hall Inc., New Jersey, 1995.