Cellular Neural Networks: Applications

LEON O. CHUA, FELLOW, IEEE, AND LIN YANG, STUDENT MEMBER, IEEE

Abstract — **The *theory* of a novel class of information-processing system, called *cellular neural networks*, which is capable of *high-speed parallel* signal processing, has been presented in a companion paper [1]. Our "dynamic route" approach for analyzing the "local dynamics" of this class of *neural* circuits" will be exploited in this paper *to steer the system trajectories* into various stable equilibrium configurations which maps onto some binary patterns we seek to recognize. Although still in its embryonic stage, some impressive applications of cellular neural networks to such areas as *image processing* and pattern recognition will be demonstrated in this paper, albeit with only a crude circuit. In particular, examples of cellular neural networks which can be designed to recognize the key features of Chinese characters will be presented.**

I. Introduction

RECENTLY, a novel class of information-processing system called *cellular neural networks* has been proposed [1]. Like *neural network* [2]–[5], it is a large-scale nonlinear analog circuit which processes signals in *real* time. Like *cellular automata* [6]–[9] it is made of a massive aggregate of regularly spaced circuit clones, called *cells*, which communicate with each other directly only through its nearest neighbors. Each cell is made of a *linear* capacitor, a *nonlinear* voltage-controlled current source, and a few *resistive* linear circuit elements.

Cellular neural networks share the best features of both worlds; its continuous time feature allows *real-time* signal processing found wanting in the digital domain and its local interconnection feature makes it tailor made for VLSI implementation.

Some theoretical results concerning the dynamic range and the steady-state behavior of cellular neural networks have been presented in [1]. In this paper, some impressive and promising applications of cellular neural networks to image processing will be described.

In the following sections, we will use cellular neural networks to solve some image processing and pattern recognition problems. We have stressed only the steady-state behavior of cellular neural networks in [1]. However, for applications in image processing, the transient behavior is equally important. In fact, it is the transient behavior which makes it possible to extract a variety of features from a single picture, and to solve various image processing problems. The role played by the cellular neural network's transient behavior will be clearly demonstrated in the following examples.

In *Section II* we will review briefly the architecture of cellular neural networks proposed in [1], and summarize some related results. In *Section III*, we will explain how cellular neural networks can be applied to image processing. In *Section IV*, the dynamical behavior of cellular neural networks will be studied by analyzing a very simple example in image processing. In *Section V*, we will discuss the computer-aided design problem, that is, the simulation of cellular neural networks. In *Section VI*, a cellular neural network for *noise removal* will be presented. In *Section VII*, we present a set of cellular neural networks for *feature extraction*. As an application of these properties, a new approach for Chinese character recognition will be described in *Section VIII*.

Manuscript received July 20, 1987; revised April 19, 1988. This work was supported in part by the Office of Naval Research under Contract N00014-86K-0351 and by the National Science Foundation under Grant MIP-8614000. The cost of the color printing is paid for entirely by a University-Industry grant. This paper was recommended by Associate Editor H. Gharavi.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

IEEE Log Number 8822600.

II. Architecture of Cellular Neural Networks

The basic circuit unit of a cellular neural network is called a *cell*. It contains linear and nonlinear circuit elements, which typically are linear capacitors, linear resistors, linear and nonlinear controlled sources, and independent sources. The structure of cellular neural networks is similar to that found in cellular automata; namely, any cell in a cellular neural network is connected only to its neighbor cells. 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 the network. An example of a two-dimensional 4×4 cellular neural network is shown in [1, fig. 1]. The ith row and jth column cell is indicated as $C(i, j)$. The *r-neighborhood* N_r of radius r of a cell, $C(i, j)$, in a cellular neural network is defined by

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

where r is a positive integer number. Usually, we call the $r=1$ neighborhood a "3×3 neighborhood." It can be shown that the neighborhood system defined above exhibits a *symmetry* property in the sense that if $C(i, j) \in N_r(k, l)$, then $C(k, l) \in N_r(i, j)$, for all $C(i, j)$ and $C(k, l)$ in a cellular neural network.

A typical example of a cell $C(i, j)$ is shown in [1, fig. 3], where the suffixes u, x, and y denote the *input*, *state*, and *output*, respectively. The node voltage v_{xij} of $C(i, j)$ is defined as the state of the cell whose *initial condition* is assumed to have a magnitude less than or equal to 1. The node voltage v_{uij} is defined as the input of $C(i, j)$ and is

assumed to be a constant with magnitude less than or equal to 1. The node voltage v_{yij} is defined as the output.

$$v_{\max} = 1 + R_x|I| + R_x \max_{1 \leq i \leq M, 1 \leq j \leq N} \left[\sum_{C(k,l) \in N_r(i,j)} \left(|A(i,j;k,l)| + |B(i,j;k,l)| \right) \right]. \quad (3)$$

C is a linear capacitor; R_x and R_y are linear resistors; I is an independent current source; $I_{xy}(i, j; k, l)$ and $I_{xu}(i, j; k, l)$ are linear voltage-controlled current sources with the characteristics $I_{xy}(i, j; k, l) = A(i, j; k, l)v_{ykl}$ and $I_{xu}(i, j; k, l) = B(i, j; k, l)v_{ukl}$ for all $C(k, l) \in N_r(i, j)$; $I_{yx} = (1/R_y)f(v_{xij})$ is a piecewise-linear voltage-controlled current source with characteristic $f(\cdot)$ as shown in [1, fig. 4]; E_{ij} is a time-invariant independent voltage source.

Applying KCL and KVL, the circuit equations of a cell are easily derived as follows:

State equation:

$$C\frac{dv_{xij}(t)}{dt} = -\frac{1}{R_x}v_{xij}(t) + \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l)v_{ykl}(t) + \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l)v_{ukl} + I, \quad 1 \leq i \leq M; 1 \leq j \leq N. \quad (2a)$$

Output equation:

$$v_{yij}(t) = \frac{1}{2}\left(|v_{xij}(t)+1| - |v_{xij}(t)-1|\right), \quad 1 \leq i \leq M; 1 \leq j \leq N. \quad (2b)$$

Input equation:

$$v_{uij} = E_{ij}, \quad 1 \leq i \leq M; 1 \leq j \leq N. \quad (2c)$$

Constraint conditions:

$$|v_{xij}(0)| \leq 1, \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (2d)$$

$$|v_{uij}| \leq 1, \quad 1 \leq i \leq M; 1 \leq j \leq N. \quad (2e)$$

Parameter assumptions:

$$A(i,j;k,l) = A(k,l;i,j), \quad 1 \leq i,k \leq M; 1 \leq j,l \leq N. \quad (2f)$$

$$C > 0, \; R_x > 0. \quad (2g)$$

In order to guarantee that the circuit equations (2) are valid for all cellular neural networks, we have proved the following theorem, which can be used to determine the dynamic range of all node voltages in the network [1].

Theorem 1

The state v_{xij} of each cell in a cellular neural network is bounded for all time $t > 0$ and the bound $v_{\max}$ can be computed by the following formula for *any* cellular neural network:

The basic function of a cellular neural network for image processing is to map or transform an input image into a corresponding output image. Here, we restrict our output images to binary images with -1 and 1 as the pixel values. However, the input images can have multiple gray levels, provided that their corresponding voltages satisfy (2e). This means that our image processing cellular neural network must always converge to a constant steady-state following any transient regime which has been initialized and/or driven by a given input image. We have also proved that our cellular neural networks [1] are completely stable.[1] The results are summarized as follow for convenience:

Theorem 2

After the transient has settled down, a cellular neural network always approaches one of its stable equilibrium points. In other words, we have

$$\lim_{t \to \infty} v_{xij}(t) = \text{constant}, \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (4)$$

or

$$\lim_{t \to \infty} \frac{dv_{xij}(t)}{dt} = 0, \quad 1 \leq i \leq M; 1 \leq j \leq N. \quad (5)$$

Moreover, we have the following.

Theorem 3

If the circuit parameters satisfy

$$A(i,j;i,j) > \frac{1}{R_x} \quad (6)$$

then

$$\lim_{t \to \infty} |v_{xij}(t)| \geq 1, \quad 1 \leq i \leq M; 1 \leq j \leq N \quad (7)$$

or equivalently,

$$\lim_{t \to \infty} v_{yij}(t) = \pm 1, \quad 1 \leq i \leq M; 1 \leq j \leq N. \quad (8)$$

Remarks:

(a) The above theorem is significant for cellular neural networks, because it implies that the circuit will not oscillate or become chaotic [10], [11].

(b) Theorem 3 guarantees that our cellular neural networks have binary-value outputs. This property is crucial for solving *classification* [12] problems in image processing applications.

III. Application of Cellular Neural Networks to Image Processing

In order to see why cellular neural networks can be used for image processing, let us first approximate the differential equation (2a) by a *difference* equation. In particular,

[1] A circuit is said to be *completely stable* iff every trajectory tends to an equilibrium state. Consequently, such a circuit *cannot* oscillate or become chaotic.

let $t = nh$, where h is a constant time step, and approximate the derivative of $v_{xij}(t)$ by its corresponding difference form

$$\frac{C}{h}\left[v_{xij}((n+1)h) - v_{xij}(nh)\right] = -\frac{1}{R_x}v_{xij}(nh) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{ykl}(nh) + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl} + I, \quad 1\leqslant i\leqslant M;\ 1\leqslant j\leqslant N \quad (9a)$$

and

$$v_{yij}(nh) = 0.5R_y\left(|v_{xij}(nh)+1| - |v_{xij}(nh)-1|\right) \equiv f\left(v_{xij}(nh)\right), \quad 1\leqslant i\leqslant M;\ 1\leqslant j\leqslant N. \quad (9b)$$

Let

$$I_{ij} = \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl} + I, \quad 1\leqslant i\leqslant M;\ 1\leqslant j\leqslant N. \quad (10)$$

We can recast (9a) into the form

$$v_{xij}(n+1) = v_{xij}(n) + \frac{h}{C}\left[\frac{-1}{R_x}v_{xij}(n) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{ykl}(n) + I_{ij}\right], \quad 1\leqslant i\leqslant M;\ 1\leqslant j\leqslant N \quad (11)$$

where we have suppressed the time step h from "nh" for simplicity; i.e., $v_{xij}(n) \equiv v_{xij}(nh)$ and $v_{yij}(n) \equiv v_{yij}(nh)$.

If we substitute (9b) for $v_{ykl}(n)$ in (11), we would obtain

$$v_{xij}(n+1) = v_{xij}(n) + \frac{h}{C}\left[\frac{-1}{R_x}v_{xij}(n) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)f(v_{xkl}(n)) + I_{ij}\right], \quad 1\leqslant i\leqslant M;\ 1\leqslant j\leqslant N. \quad (12)$$

Equation (12) can be interpreted as a *two-dimensional filter* for transforming an image, represented by $v_x(n)$, into another one, represented by $v_x(n+1)$. The filter is nonlinear because $f(v_{xkl}(n))$ in (12) is a nonlinear function. Usually, the filter is *space invariant* [12] for image processing, which means that $A(i,j;k,l) = A(i-k, j-l)$ for all i, j, k, and l. The property of the filter is determined by the parameters in (12). How to choose the filter parameters to achieve a desired image transformation is currently still an active research problem. However, some basic properties of image processing filters can be found in [12], [13].

For the one-step filter in (12), the pixel values, $v_{xij}(n+1)$, of an image is determined directly from the pixel values, $v_{xij}(n)$, in the corresponding neighborhood $N_r(i,j)$. From the practical point of view, this neighborhood is always chosen to be as small as possible. A typical choice is the 3×3 neighborhood, $N_1(i,j)$. Therefore, a one-step filter can only make use of the *local* properties of images.

When the *global* properties of an image is important, the above one-step filter can be iterated n times to extract additional global information from the image. One well-known property of an iterative filter is the so-called *propagation property*, which asserts that the pixel values of the output image after n iterations can be *indirectly* affected by a larger neighbor region of the input image. This property can be observed by substituting $v_{xij}(n)$ in (12) iteratively down to $v_{xij}(0)$, which coincides with the input image. Since (12) contains some nonlinear functions, it is convenient to represent $v_{xij}(n)$ by

$$v_{xij}(n) = \sum_{C(k,l)\in N_{nr}(i,j)} g^n_{ijkl}\left(v_{xkl}(0)\right), \quad 1\leqslant i\leqslant M;\ 1\leqslant j\leqslant N \quad (13)$$

where g^n_{ijkl} is a nonlinear function, which depends on (i,j), (k,l), and n. Note that the neighborhood $N_{nr}(i,j)$ is n times larger than $N_r(i,j)$ and depends on the iteration number n. It is easy to see that when the iteration number n is sufficiently large, the neighborhood $N_{nr}(i,j)$ will eventually cover the entire image, independent of (i,j) and r.

Therefore, the propagation property of iterative filters makes it possible to extract some global features in images. Of course, the local properties are still preserved in iterative filters with the closer neighbors having more effects than those farther away.

K. Preston, Jr. and M. J. B. Duff considered a special class of iterative filters in [9], where they considered the images as binary images and the operations as Boolean functions. They call these kind of filters *cellular logic transforms*. Several machines have been built to implement cellular logic transforms. One of them, called a cellular logic image processor (CLIP) [9], has been used successfully in practice.

Now, it is not difficult to understand how cellular neural networks can be used for image processing. Indeed, if we let $h \to 0$ in (9a), we would recover the system equation (2) defining a cellular neural network. To understand the image transform mechanism in our cellular neural networks, let us rewrite (2) in its equivalent integral form as follows:

$$v_{xij}(t) = v_{xij}(0) + \frac{1}{C}\int_0^t \left[\frac{-1}{R_x}v_{xij}(\tau) + f_{ij}(\tau) + g_{ij}(u) + I\right]d\tau, \quad 1\leqslant i\leqslant M;\ 1\leqslant j\leqslant N, \quad (14a)$$

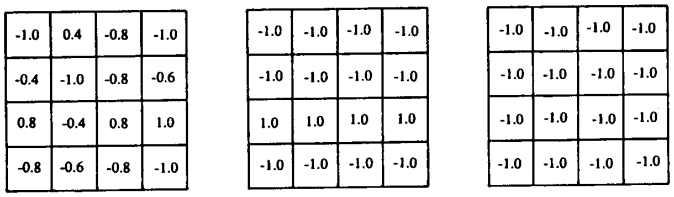

Fig. 1. Input and output images for the simple example. (a) The input image to be processed. (b) The output image of the horizontal line detector. (c) The output image of the vertical line detector.

where

$$f_{ij}(t) = \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l) v_{ykl}(t) \quad (14b)$$

and

$$g_{ij}(u) = \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l) v_{ukl}. \quad (14c)$$

Equation (14) represents the image at time t, which depends on the initial image $v_{xij}(0)$ and the *dynamic rules* of the cellular neural network. Therefore, we can use a cellular neural network to obtain a *dynamic transform* of an initial image at *any time t*. In the special case where $t \to \infty$, the state variable v_{xij} tends to a constant and the output v_{yij} tends to either $+1$ or -1 as stated in Theorem 2 and Theorem 3. In the following sections, we will present some examples of this kind of image transforms.

IV. A Simple Example

Before we consider real image processing problems, it is instructive to look at a very simple example. Although it is a much simplified image processing problem, this example will help us to understand some of the dynamic behavior of cellular neural networks and to derive some intuitive ideas on how to design cellular neural networks for solving a specific practical image processing problem.

One important problem in image processing is *pixel classification* [12]. To illustrate this concept, consider a small image, as shown in Fig. 1(a). This image is a 4×4 pixel array with each pixel value $P_{ij} \in [-1,1]$, for $1 \leqslant i \leqslant 4$ and $1 \leqslant j \leqslant 4$. Assume that the pixel value -1 corresponds to a white background, the pixel value 1 corresponds to a black object point value, and the pixel values between -1 and 1 correspond to the gray values. The *pixel classification* problem is to classify each pixel of the image into two or more classes.

From the mathematical point of view, *pixel classification* can be considered as a map, F, which maps a continuous vector space into a discrete vector space as defined below:

$$F: [a,b]^{M \times N} \to \{A, B, C, \cdots\}^{M \times N} \quad (15)$$

where $M \times N$ is the number of pixels in an image and $A, B, C, \cdots$ stand for different classes. For this example, we wish to assign to each pixel in the array one of the two values, -1 and 1, based on some classification rules and the original pixel values. So, F is defined by

$$F: [-1.0,1.0]^{M \times N} \to \{-1,1\}^{M \times N}. \quad (16)$$

0.0	0.0	0.0
1.0	2.0	1.0
0.0	0.0	0.0

Fig. 2. A typical cloning template of an interactive cell operator. The unit used here is $10^{-3}\ \Omega^{-1}$.

Suppose we wish to design a horizontal line detector to filter out the horizontal lines in the input image in Fig. 1(a) by using a cellular neural network. In order to simplify our analysis, we have chosen a very simple dynamic rule for this "horizontal line detector" circuit. The circuit element parameters of the cell $C(i,j)$ are chosen as follows:

$$C = 10^{-9}\text{F}; \quad R_x = 10^3\ \Omega; \quad I = 0$$
$$A(i,j;i-1,j-1) = A(i,j;i-1,j) = A(i,j;i-1,j+1) = 0$$
$$A(i,j;i,j) = 2 \times 10^{-3}\ \Omega^{-1};$$
$$A(i,j;i,j-1) = A(i,j;i,j+1) = 10^{-3}\Omega^{-1}$$
$$A(i,j;i+1,j-1) = A(i,j;i+1,j) = A(i,j;i+1,j+1) = 0$$

for a 3×3 neighborhood system. Since the feedback operators $A(i,j;k,l)$, as shown above, are independent of the absolute position of the cell, we can simplify the expressions of $A(i,j;k,l)$ by coding them as follows:

$$A(-1,-1) = A(-1,0) = A(-1,1) = 0$$
$$A(0,0) = 2 \times 10^{-3}\Omega;$$
$$A(0,-1) = A(0,1) = 10^{-3}\Omega$$
$$A(1,-1) = A(1,0) = A(1,1) = 0.$$

The indexes in the above interactive parameters indicate the relative positions with respect to $C(i,j)$. The cellular neural network has the space invariance property, that is, $A(i,j;k,l) = A(i-k,j-l)$ for all i, j, k, and l. Therefore, as in image processing filters, we can use a cloning template, as shown in Fig. 2, to describe the feedback operator of the cell, henceforth called a feedback operator cloning template. The cloning template is constructed as follows: the center entry of the cloning template corre-

sponds to $A(0,0)$; the upper left corner entry of the cloning template corresponds to $A(-1,-1)$; the lower right-hand corner entry of the cloning template corresponds to $A(1,1)$; and so forth. Since it is extremely convenient and clear to characterize the interactions of a cell with its neighbors by means of a cloning template, we will use the cloning template expression in the following examples.

The dynamical equations of the cellular neural network corresponding to the above parameters are given by

$$\frac{dv_{xij}(t)}{dt} = 10^6\left[-v_{xij}(t)+v_{yij-1}(t)+2v_{yij}(t)+v_{yij+1}(t)\right] \tag{17a}$$

and

$$v_{yij}(t) = 0.5\left(|v_{xij}(t)+1|-|v_{xij}(t)-1|\right), \quad \text{for } 1\leqslant i\leqslant 4;\ 1\leqslant j\leqslant 4. \tag{17b}$$

Note that we have chosen the control operator $B(i, j; k, l) = 0$ for all i, j, k, and l in this circuit, and $A(i, j; k, l) > 1/R_x$ as stipulated in the condition of Theorem 3.

For this example, the initial state of the cellular neural network is the pixel array in Fig. 1(a). From the circuit equation (17), we note that the derivative of the pixel values depends on their left and right neighbors and themselves, but not on the upper and lower neighbors. This particular dynamic rule will therefore enhance the detection of horizontal lines in the original image.

The circuit equations in (17) are first-order nonlinear ordinary differential equations. In system theory, they are also called a piecewise-linear autonomous system. In general, it is difficult if not impossible to predict the behaviors of complex nonlinear dynamical systems. But our analysis in [1] shows that the circuit in this example will tend to one of its equilibrium points after its transient has settled down. Let us now analyze the steady-state behavior of this system.

The equilibrium points of the system can be found by solving the equivalent dc circuit equations (replace all capacitors by open circuits):

$$v_{xij}(t) = v_{yij-1}(t)+2v_{yij}(t)+v_{yij+1}(t) \tag{18a}$$

$$v_{yij}(t) = 0.5\left(|v_{xij}(t)+1|-|v_{xij}(t)-1|\right), \quad 1\leqslant i\leqslant 4;\ 1\leqslant j\leqslant 4. \tag{18b}$$

In general, for piecewise-linear circuits, we can find all solutions of the dc circuit equations either by the brute force algorithm [14] or by using some more efficient algorithms [15], [16]. However, even for this very simple example, there are 32 unknown variables and 32 equations (16 linear equations and 16 piecewise-linear equations). It is time consuming to find all of the equilibrium points of a cellular neural network by using the algorithms mentioned above because of the large size of its circuit equations. (Note that if the nonlinearity of the circuit is not piecewise-linear, there is no general method to find all of its equilibrium points.)

To simplify our problem, we will take advantage of the various features of cellular neural networks in our analysis. As mentioned before, every cell in a cellular neural network has the same connections as its neighbors. Therefore, each cell's circuit equation is the same as that of the other cells in the same circuit. (Without loss of generality, we ignore boundary effects.) Hence, *we can understand the global properties of a cellular neural network by studying the local properties of a single cell.* This approach is extremely useful for the analysis and design of cellular neural networks.

Before analyzing this example, it is helpful to recall the following definitions [1]:

Definition 1: Cell equilibrium state:

A *cell equilibrium state* v_{xij}^* of a cell circuit $C(i, j)$ in a cellular neural network with dc input voltages v_{ukl} is any value of the state variable v_{xij} which satisfies

$$\text{(a)}\quad \left.\frac{dv_{xij}(t)}{dt}\right|_{v_{xij}=v_{xij}^*} = 0 \tag{19a}$$

$$\text{(b)}\quad v_{ykl} = \pm 1 \quad \text{for } \textit{all neighbor cells } C(k,l)\in N_r(i,j). \tag{19b}$$

It follows from *Definition 1* that the set of all cell equilibrium states of a cell circuit $C(i, j)$ can be found by equating the right-hand side of (2a) to zero and by setting for v_{ykl} to be *any* combination of ± 1, and then solving for v_{xij}. Note that since the resulting equation is decoupled from the rest, it can be trivially solved. On the other hand, many of these solutions could be *extraneous* since when the entire coupled system (2) is solved for its equilibrium points, some combinations of $v_{yij} = \pm 1$ may not satisfy these equations.

Observe also that it is possible for a cell equilibrium state to have an absolute value less than one, in which case even if it is a valid solution, it would *not* be observable (i.e., *unstable*) in view of *Theorem 3*, assuming that $A(i, j; k, l) > 1/R_x$. We will usually be interested only in those cell equilibrium states having a magnitude greater than one.

Definition 2: Stable cell equilibrium states:

A cell equilibrium state v_{xij}^* of a cell circuit $C(i, j)$ is said to be *stable* iff

$$|v_{xij}^*| > 1. \tag{20}$$

Observe that since the *observable* (i.e., *stable*) *output* variable v_{yij} of each cell $C(i, j)$ can assume only the values $+1$ or -1, it follows that any observable output solution vector $\boldsymbol{v}_y$ of any cellular neural network with $A(i, j; k, l) > 1/R_x$ must necessarily be located at a vertex of an n-dimensional hypercube $\boldsymbol{S}$, where $n = M \times N$ denotes the total number of cells. If we let $\boldsymbol{S}_o$ denote the set of all n-dimensional vectors $\boldsymbol{v}_y$ whose components v_{yij} correspond to all combinations of stable cell equilibrium points, then $\boldsymbol{S}_o$ can have at most 2^n members, henceforth called the set of *virtual equilibrium output vectors*. We use the adjective "virtual" to emphasize that some members of

S_o may not be valid equilibrium points of the *overall* circuit. This is because some of the *arbitrary* combinations of $v_{ykl} = \pm 1$ used in calculating the circuit equilibrium states may *not* correspond to actual solutions obtained by solving the *complete* coupled systems of algebraic equations (2) with the left-hand side of (2a) set equal to zero.

Definition 3: Stable system equilibrium point:

A *stable system equilibrium point* of a cellular neural network with $A(i, j; k, l) > 1/R_x$ is any equilibrium state vector v_x whose state variable components v_{xij} (i.e., all capacitor voltages) consist of *stable* cell equilibrium states.

It is important in the following analysis to remember that whereas every stable system equilibrium point is made up of stable cell equilibrium states, the converse is not true as some combinations of stable cell equilibrium points may not correspond to an actual equilibrium point of the overall circuit. If we denote the corresponding set S_0^* of observable equilibrium output vectors, then we have $S_0^* \subset S_o \subset S$.

Now, let us compute the stable cell equilibrium states of an inner cell of the preceding circuit example. Considering the condition in the above definitions, the stable cell equilibrium states can be obtained by solving (18a) for v_{xij} with v_{yij} taking on either the value $+1$ or -1:

$$v_{xij} = \operatorname{sgn}[v_{yij-1}] + 2\operatorname{sgn}[v_{yij}] + \operatorname{sgn}[v_{yij+1}] \quad (21a)$$

$$|v_{xij}| \geq 1, \qquad 1 \leq i \leq M;\ 1 \leq j \leq N. \quad (21b)$$

Substituting v_{yij-1} and v_{yij+1} in the above equations by ± 1, and considering that $\operatorname{sgn}[v_{yij}] = \operatorname{sgn}[v_{xij}]$ from (18b), we obtain the following four cases:

(a) For $v_{yij-1} = -1$ and $v_{yij+1} = -1$, we have $v_{xij} = -2 + 2\operatorname{sgn}[v_{xij}]$, and hence $v_{xij} = -4$.
(b) For $v_{yij-1} = +1$ and $v_{yij+1} = -1$, we have $v_{xij} = 2\operatorname{sgn}[v_{xij}]$, and hence $v_{xij} = -2$, *or* 2.
(c) For $v_{yij-1} = -1$ and $v_{yij+1} = +1$, we have $v_{xij} = 2\operatorname{sgn}[v_{xij}]$, and hence $v_{xij} = -2$, *or* 2.
(d) For $v_{yij-1} = 1$ and $v_{yij+1} = 1$, we have $v_{xij} = 2 + 2\operatorname{sgn}[v_{xij}]$, and hence $v_{xij} = 4$.

It follows from the above analysis that the stable cell equilibrium states of any inner cell circuit for our present example are -4, -2, 2, and 4.

It is obvious that the stable cell equilibrium state of each cell depends on the stable cell equilibrium states of its neighbor cells. Of course, if the input of the cellular neural network is not zero, then the stable cell equilibrium states of a cell circuit will also depend on the input, v_u. Therefore the stable system equilibrium points of a cellular neural network by Definition 3 depend on the initial conditions, the inputs, and the dynamic rule of the circuit. *Any stable system equilibrium point can be considered as a two-dimensional pattern*. For a given cellular neural network, like the one in this example, there are only a finite number of discrete patterns. The continuous image space, $[-1, 1]^{M \times N}$, therefore, is transformed into a finite number of discrete patterns by a cellular neural network. The properties of the patterns can be determined by those of the stable cell equilibrium states. Consequently, the output, v_y, must possess some structural features for a specific cellular neural network. For example, if $v_{yij-1} = 1$ and $v_{yij+1} = 1$, as prescribed by the above case (d), then $v_{yij} = 1$ is the only choice.

As noted in *Section III*, the image transform by a cellular neural network is a dynamical transform. So, it is important to consider the *transient response* of a cellular neural network. The transient response of a cellular neural network is simply the *trajectory* from the initial state to the steady state of the system. It is well known from the theory of differential equations that any stable system equilibrium point, as defined in Definition 3, of a cellular neural network is a *limit point* of a set of trajectories of the corresponding differential equations (2). Such an attracting limit point has a *basin of attraction* which consists of the union of all trajectories tending to this limit point. The state space of the system is partitioned into a set of basins centered at the stable system equilibrium points. Then the map F, as defined in (16), can be characterized by the basins and, hence, by the transient response. Since it is extremely difficult if not impossible to analyze the transient behavior of such complex nonlinear dynamical system, we will use computer simulation method to study the transient behaviors of cellular neural networks in the following examples.

Let us summarize the above observations from this example,

(a) Any input image represented in analog form can be mapped into a specific output image with binary values by a cellular neural network.
(b) For a given cellular neural network, the output images are inbued with some spatial structures resulting from the dynamic rule of the circuit. For instance, it is impossible to have a row like $[1, -1, 1, -1]$, which is a rejected pattern, in the output image of the cellular neural network in the above example. Hence, an appropriately chosen dynamic rule could imbue a cellular neural network with the ability to recognize and extract some special patterns from input images.
(c) Different input images may be mapped into the same output image if they have the same patterns, and the same input image may be mapped into different output images by different cellular neural networks for different image processing or pattern recognition purposes.

The dynamic behavior of the cellular neural network with zero control operators and nonzero feedback operators in this example is reminiscent of two-dimensional *cellular automata* [6]–[8]. K. Preston, Jr. and M. J. B. Duff have provided some applications of cellular automata in their image processing systems [9]. The main difference between a cellular neural network and a cellular automata machine is in their dynamic behavior. The former is a *continuous*-time dynamical system while the latter is a

```
*************************************************************************

name_of_circuit  "simple horizontal lines detector"
comments  "An example of the application of cellular neural networks."
circuit_size  4 4
vector_dimension  1
c1  1.0E-9
r1  1000
y1  "pwl  term = 3 nseg = 3"
    "ap= 0,0 bp= 0,0,0,0 cp= 0,0,-0.5,0.5 alphap= 1,0,1,0  betap=-1,1"
all size 3
    templet 0.0    0.0    0.0
            0.001  0.002  0.001
            0.0    0.0    0.0
data_type x1
transient "0.1us    5us    UIC"
output    y1

*************************************************************************
```

(a)

```
*************************************************************************

      This is the initial state of the horizontal line detector.

      Where      "." : -1.0;
                 "1" : -0.8;
                 "2" : -0.6;
                 "3" : -0.4;
                 "4" : -0.2;
                 "5" :  0.0;
                 "6" :  0.2;
                 "7" :  0.4;
                 "8" :  0.6;
                 "9" :  0.8;
                 "*" :  1.0.

$
                              .71.
                              3.12
                              939*
                              121.

*************************************************************************
```

(b)

Fig. 3. (a) The cellular neural network description file of the simple horizontal line detector for the circuit simulation preprocessor CELL. (b) The data file of CELL for the image in Fig. 1(a).

discrete-time dynamical system. Because the two systems have many similarities, we can use cellular automata theory to study the steady-state behavior of cellular neural networks. Another remarkable distinction between them is that while the states of a cellular neural network will always tend to equilibrium points, a cellular automaton will usually have a much richer dynamical behavior, such as periodic, chaotic and even more complex phenomena. This is because we have chosen a sigmoid nonlinearity for the nonlinear circuit elements in our cellular neural networks. If we choose some other form of nonlinearity for the nonlinear elements, many more complex phenomena will also occur in cellular neural networks.

V. Simulation of Cellular Neural Networks

Since cellular neural networks are nonlinear dynamical systems, there are presently few analytical methods for studying their transient behaviors. Consequently, it is necessary to use computer simulation. The circuit simulator we used is PWLSPICE [17], which is a modified version of SPICE.3 [18] for piecewise-linear circuit analysis. In this section, we will introduce a *preprocessor* of PWLSPICE, called CELL, which automatically generates the input circuit files for PWLSPICE according to an easily understandable circuit description file and a data file. We will also describe two *postprocessors*, called BINF and PLOT, which transfer the standard outputs from PWLSPICE into a binary data file and then feed them to a color graphics terminal for display.

Our computer simulation procedure consists of the following steps:

Step 1: Write a cellular neural network description file for CELL. As an example, the circuit description file for the simple example in the preceding section is shown in Fig. 3(a). The first word of each line in the description file is a *key* word: it tells CELL the meaning of the parameters which follow.

Step 2: Write a data file for CELL, which consists of the initial condition and/or the input of the cellular neural network. The data file for the horizontal line detector is shown in Fig. 3(b). Note that we have quantized the pixel values in this example into 11 discrete levels to simplify the display.

Step 3: After running CELL with the circuit description and data files, we obtain the input file for the circuit simulator PWLSPICE. The PWLSPICE input file of the horizontal line detector is shown in Table I in the Appendix.

(To save space, only part of the input file is presented.)

Step 4: Obtain the transient simulation output of PWLSPICE. The transient simulation of the horizontal line detector is shown in Table II in the Appendix. (Again only part of the output file is presented.)

Step 5: Transfer the PWLSPICE output file to a binary data file by using BINF.

Step 6: Display the output image on a color graphics terminal (we use a MASSCOMP terminal) by using PLOT with the binary output data file.

Our simulation result of the simple example in *Section IV* is shown in Fig. 1(b). Note that row 3 stands out as a black horizontal line with value 1.0 and flanked by a white background with value -1.0. Hence, even such a crude horizontal line detector circuit is capable of extracting the horizontal line structures in the given image in Fig. 1(a). (Of course, this simple dynamic rule cannot handle more sophisticated line detection problems.)

If we change the dynamic rule of our circuit, say, instead of (17a), we use

$$\frac{dv_{xij}(t)}{dt} = 10^6\left[-v_{xij}(t) + v_{yi-1j}(t) + 2v_{yij}(t) + v_{yi+1j}(t)\right] \tag{17a'}$$

then this cellular neural network becomes a simple vertical line detector. Its simulation result for the pixel array in Fig. 1(a) is shown in Fig. 1(c). Here, all pixels are of uniformly white with value -1.0. From the simulation results of the above two detector circuits, we know that the pixel array in Fig. 1(a) has horizontal lines but no vertical lines.

VI. Cellular Neural Networks for Noise Removal

Now, let us consider one of the most important problems in image processing. Since the input pictures usually come from the real world through a camera or some other optical equipment, there will always be some noise superimposed on the images of the objects. In this paper, we will concentrate on text processing problems, specifically, on the recognition of Chinese characters. Suppose that the characters in the input images are smeared in some way such as the picture in the upper left-hand corner of Fig. 5. The colors in Figs. 5–14 and 19–35 are chosen to distinguish the gray levels of the pixels in the pictures, where

background = light blue;

$\left[-1.0, -\frac{7}{8}\right]$ = greenish blue; $\left(-\frac{7}{8}, -\frac{6}{8}\right)$ = blue green;

$\left(-\frac{6}{8}, -\frac{5}{8}\right)\Big]$ = bluish green; $\left(-\frac{5}{8}, -\frac{4}{8}\right]$ = green;

$\left(-\frac{4}{8}, -\frac{3}{8}\right]$ = yellowish green; $\left(-\frac{3}{8}, -\frac{2}{8}\right]$ = green yellow;

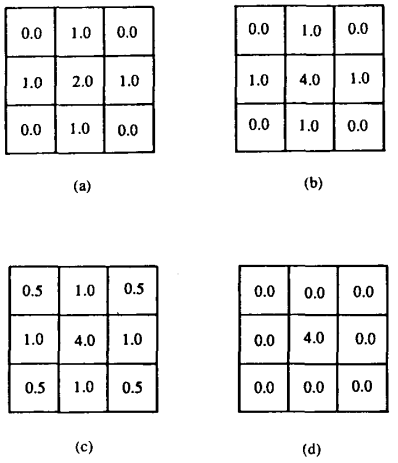

Fig. 4. Cloning templates for various interactive cell operators for noise removing cellular neural networks. The unit used here is $10^{-3}\ \Omega^{-1}$.

$\left(-\frac{2}{8}, -\frac{1}{8}\right]$ = greenish yellow; $\left(-\frac{1}{8}, 0.0\right]$ = yellow;

$\left(0.0, \frac{1}{8}\right]$ = orangish yellow; $\left(\frac{1}{8}, \frac{2}{8}\right]$ = yellow orange;

$\left(\frac{2}{8}, \frac{3}{8}\right]$ = yellowish orange; $\left(\frac{3}{8}, \frac{4}{8}\right]$ = orange;

$\left(\frac{4}{8}, \frac{5}{8}\right]$ = reddish orange; $\left(\frac{5}{8}, \frac{6}{8}\right]$ = orange red;

$\left(\frac{6}{8}, \frac{7}{8}\right]$ = orangish red; $\left(\frac{7}{8}, 1.0\right]$ = red.[2]

For this case, the smeared character in Fig. 5 is generated from a perfect binary image by adding a Gaussian white noise with $\sigma = 0.2$, $m = 0$. The size of the image in Fig. 5 is 16×16 pixels, so the noise removing cellular neural network should have 16×16 cells. In image processing, the simplest way to delete noise from the image is to use an averaging operator [12]. We choose therefore the averaging operator as the dynamic rule for our "noise removing" cellular neural network. This averaging operator can be expressed by a cloning template as shown in Fig. 4(a), and we use it as the feedback operator. Assuming that the other circuit parameters are the same as those in the simple example in Section IV, the resulting cell circuit equations are given by

$$\frac{dv_{xij}(t)}{dt} = 10^6\left[-v_{xij}(t) + v_{yi-1j}(t) + v_{yij-1}(t) + 2v_{yij}(t) + v_{yij+1}(t) + v_{yi+1j}(t)\right] \tag{22a}$$

and

$$v_{yij}(t) = 0.5\left(\left|v_{xij}(t)+1\right| - \left|v_{xij}(t)-1\right|\right). \tag{22b}$$

[2] The above color classification terminology follows the MASSCOMP graphics applications programming manual. Due to printing inperfections, the actual printed color may differ from this legend.

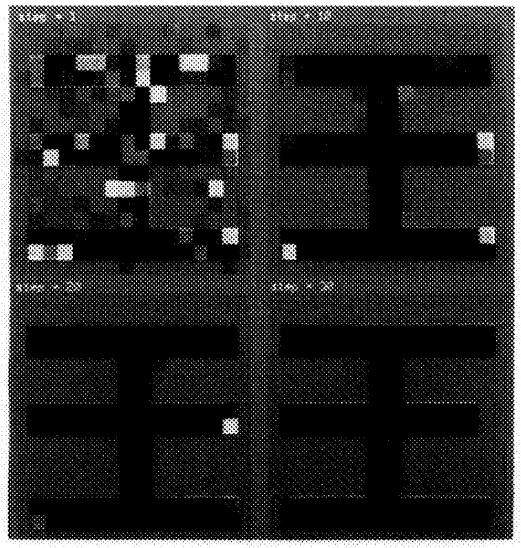

Fig. 5. The simulation result of a noise removing cellular neural network. Here $\sigma = 0.2$ and the interactive cell operator is defined by the cloning template in Fig. 4(a). (a) The upper left picture, is the input image. (b) The upper right picture, is the image at time step 10. (c) The lower left picture is the image at time step 20. (d) The lower right picture, is the image at time step 30 (output image).

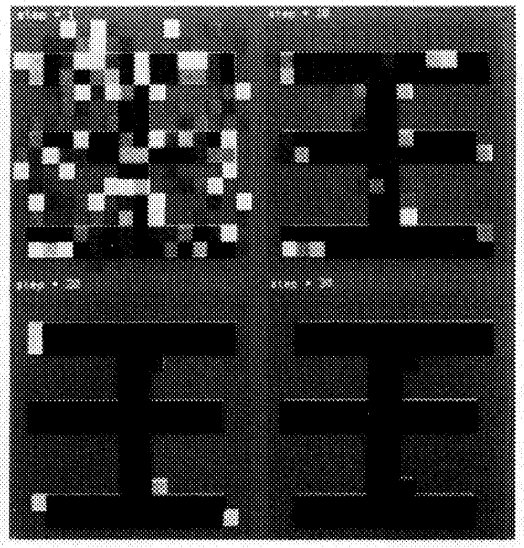

Fig. 6. The same as Fig. 5 except that $\sigma = 0.4$.

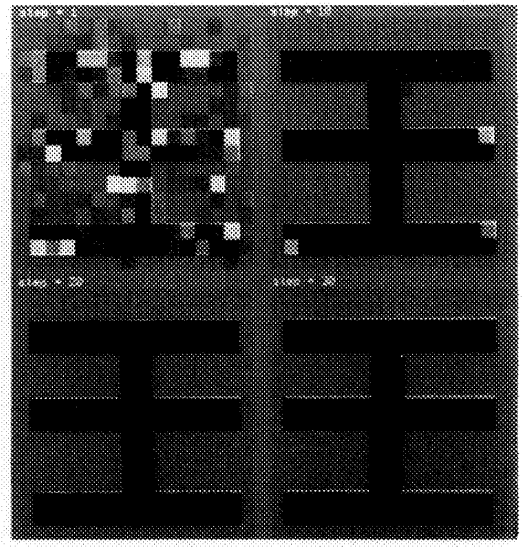

Fig. 7. The same as Fig. 5 except that the interactive cell operator is defined by the cloning template in Fig. 4(b).

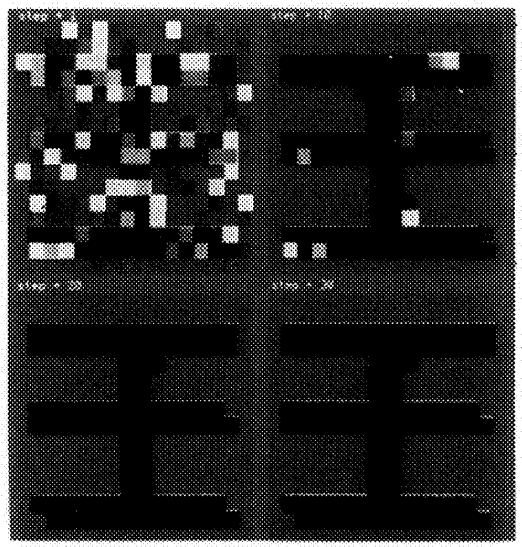

Fig. 8. The same as Fig. 6 except that the interactive cell operator is defined by the cloning template in Fig. 4(b).

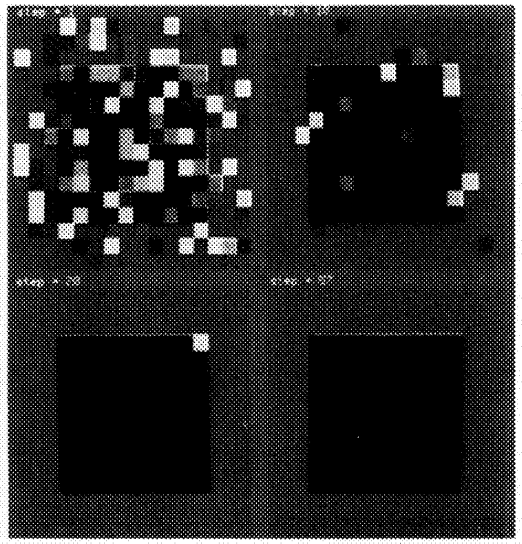

Fig. 9. Simulation results of a noise removing cellular neural network. Here $\sigma = 0.6$ and the interactive cell operator is defined by the cloning template in Fig. 4(b).

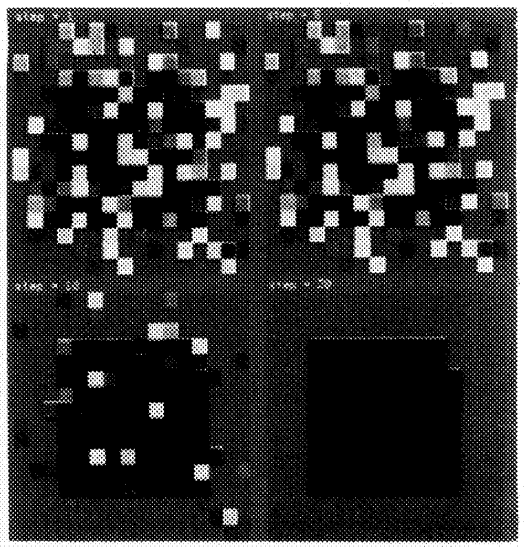

Fig. 10. The same as Fig. 9 except that $\sigma = 1.0$.

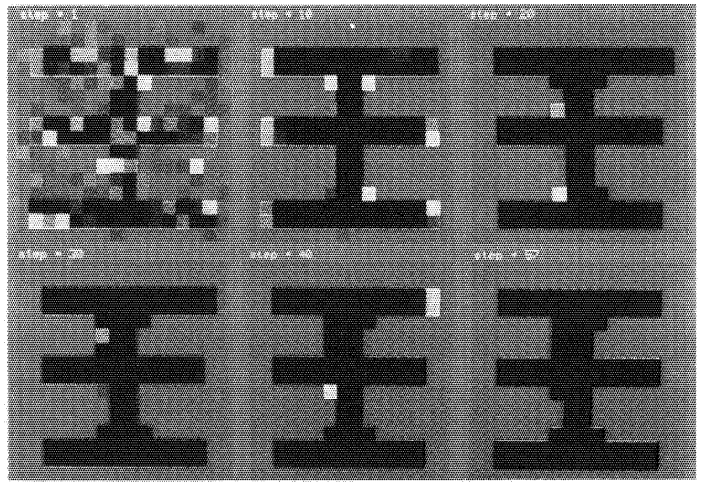

Fig. 11. Simulation results of a noise removing cellular neural network. Here $\sigma = 0.2$ and the interactive cell operator is defined by the cloning template in Fig. 4(c). (a) The upper left picture, is the input image. (b) The upper middle picture, is the image at time step 10. (c) The upper right picture, is the image at time step 20. (d) The lower left picture, is the image at time step 30. (e) The lower middle picture, is the image at time step 40. (f) The lower right picture, is the image at time step 57 (output image).

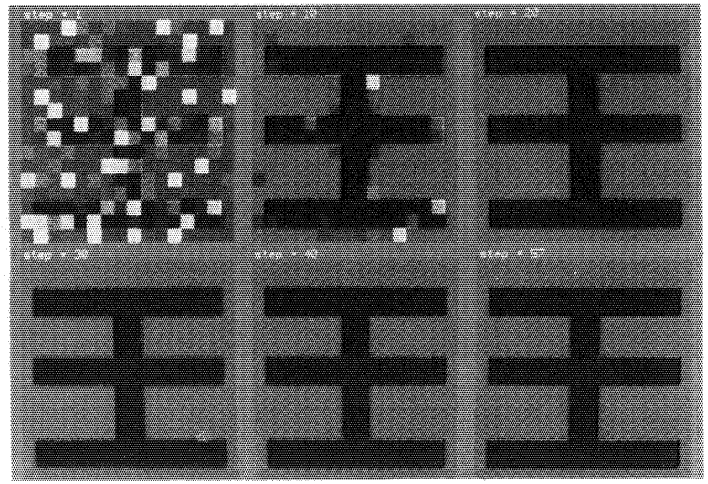

Fig. 13. Simulation results of a noise removing cellular neural network for a non-Gaussian noise image. Here the interactive cell operator is defined by the cloning template in Fig. 4(a).

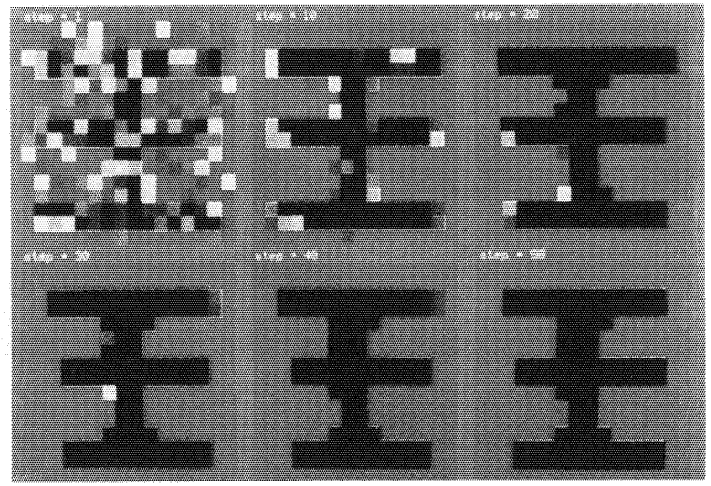

Fig. 12. The same as Fig. 11 except that $\sigma = 0.4$.

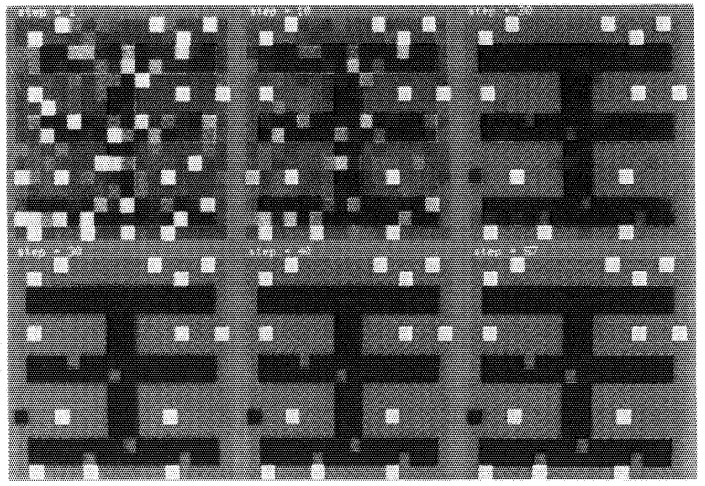

Fig. 14. Simulation results of a noise removing cellular neural network. Here the interactive cell operator is defined by the cloning template in Fig. 4(d).

Note that the rate of change of the state of cell $C(i, j)$ is approximately proportional to the average of the outputs of the neighborhood $N_1(i, j)$. Hence, the steady state of $C(i, j)$ depends on the average of those of its neighbor cells.

The rest of the pictures in Fig. 5 are the outputs of the cellular neural network at time steps 10, 20, and 30, respectively. This cellular neural network has the same properties as a two-dimensional low-pass filter. It retains the low-frequency components while eliminating the high-frequency components. Consequently, the corners of the objects in the images suffer from the same problem we experienced in two-dimensional low-pass filters. In the spectrum of an image, the high-frequency components contain information about the corners of objects. These high-frequency components are removed along with the high-frequency noise because of the low-pass filter effect. Therefore, the *pixel classification* will not be correct at the corners of objects. To see this, let us consider the upper left picture of Fig. 6. The only difference between the smeared pictures in Figs. 6 and 5 is that the standard deviation of the Gaussian white noise in Fig. 6 is 0.4 ($\sigma = 0.4$). For comparison, the circuit simulation results are shown in Fig. 6, for time steps 10, 20, and 30.

If we increase the weight of the cell itself in the feedback operator, as shown in Fig. 4(b), we would obtain the simulation results in Figs. 7 and 8 for $\sigma = 0.2$ and $\sigma = 0.4$, respectively. Observe that the results in Figs. 7 and 8 are better than those in Figs. 5 and 6 in the sense that they are closer to the undisturbed character. Particularly, the character in the lower right picture of Fig. 7 is exactly the original undisturbed character.

The above cell feedback operator involves only four nearest neighbors. To see the effects of the other neighbors of a cell, let us use a feedback operator as shown in Fig. 4(c). The corresponding simulation results are as shown in Fig. 11 for $\sigma = 0.2$, and in Fig. 12 for $\sigma = 0.4$.

Usually the noise in images is not Gaussian noise. Fig. 13 shows a simulation result for a non-Gaussian noise image. The feedback operator used here is the same as that for Fig. 4(a).

To see the effects of interactions between cells in cellular neural networks, Fig. 14 provides a simulation result of a non-Gaussian noise image, which is processed by a cellular neural network with its feedback operator as shown in Fig. 4(d). (Note that, the pixel values of the yellow points in the output image (the lower right picture) of Fig. 14 are zeros. They are the *unstable local equilibrium states*. Unlike those in a physical electronic circuit, the *unstable local equilibrium states* can be observed from computer simulations.)

From the above results, it can be seen that cellular neural networks are effective for removing noise in image processing, especially for images with large objects and few corners. Fig. 9 is the simulation result for a large object with a Gaussian white noise ($\sigma = 0.6$). The feedback operator used in this simulation is that in Fig. 4(a). Fig. 10 is the result for $\sigma = 1.0$ case.

VII. Cellular Neural Networks for Feature Extraction

Feature extraction is another important problem in image processing. As we have seen in the previous simple example in *Section IV*, the cellular neural network can extract horizontal lines in the input image in a very simple case. In this section, we will give some other examples of cellular neural networks for feature extraction in image processing. These examples can help the reader to understand the application of cellular neural networks to feature extraction.

7.1. Extract the Edges of a Diamond

Consider the upper left image shown in Fig. 19; namely, the picture of a diamond. What we want to do here is to extract the edges of this diamond, since they contain most of the information regarding the shape of the diamond. This time we will use another two-dimensional filter, called the *Laplacian operator*, shown in Fig. 15, as the feedback operator for our cellular neural network. The Laplacian is a well-known operator which is good for edge detection [12], [13]. We still use the same parameters C, R_x and $B(i, j; k, l)$ as those in the circuit in *Section II*. However, we choose $I = -1.75 \times 10^{-3}$ A for our diamond edge-extraction cellular neural network.

The cell circuit equation of this cellular neural network is given by

$$\frac{dv_{xij}(t)}{dt} = 10^6\big[-v_{xij}(t) - 0.5v_{yi-1j}(t) - 0.5v_{yij-1}(t) + 2v_{yij}(t) - 0.5v_{yij+1}(t) - 0.5v_{yi+1j}(t) - 1.75\big] \tag{23a}$$

and

$$v_{yij}(t) = 0.5\big(|v_{xij}(t)+1| - |v_{xij}(t)-1|\big), \quad 1 \leqslant i \leqslant 16,\ 1 \leqslant j \leqslant 16. \tag{23b}$$

The result of our circuit simulation is shown in Fig. 19, which is just what we expect. The parameter I in this example can control the derivatives of the state variables, and thus affects the dynamics of cellular neural networks. For example, if we choose $I = -1.5 \times 10^{-3}$ A or $I = -2.0 \times 10^{-3}$ A and keep the other circuit parameters the same as those above, then the results of simulations will be those as shown in Fig. 20 or Fig. 21, respectively.

7.2. Extract the Edges of a Square

If we use the cellular neural network designed in the preceding section with (23) to process the upper left image shown in Fig. 22, we would obtain the results as shown in Fig. 22. Observe that the output image, or the output of the cellular neural network at steady state, fails to extract the edges of the square: only the corners have been extracted. Why is this? Let us take a closer look at the dynamic rule of this cellular neural network. At the initial time $t = 0$ of the circuit transient, one can enumerate a maximum of ten possible structures, as shown in Fig. 16, around a cell. (Here we have ignored the cells on the boundary of the image and listed only those structures

0.0	-0.5	0.0
-0.5	2.0	-0.5
0.0	-0.5	0.0

Fig. 15. The cloning template defining the Laplacian operator.

which *cannot* be obtained from the others by a symmetrical transformation; e.g., a rotation.) The derivative of the state voltage with respect time t at the initial time $t=0$ can be computed from (23); namely,

(a) -0.75×10^6; (b) -0.75×10^6;
(c) -1.75×10^6; (d) -1.75×10^6;
(e) -2.75×10^6; (f) -2.75×10^6;
(g) -2.75×10^6; (h) -1.75×10^6;
(i) -1.75×10^6; (j) -0.75×10^6.

Observe that all cells have a *negative* state voltage derivative at the initial time of the circuit transient. It follows that the state voltages of all cells must decrease at $t=0$. For the cells with the structures of Fig. 16(a)–(e), the state voltages $v_{xij}(t)$ all begin by decreasing from -1.0 (the center of each template in (a)–(e) is equal to -1.0), and hence the corresponding output voltages will remain unchanged at $v_{yij}(t)=-1.0$. Consequently, these cells do not change their effects on their neighbor cells. However, for those cells with the structures of (f)–(j) in Fig. 16, both the state and the output voltages must decrease from 1.0 (the center of each template of (f)–(j) is equal to 1.0) simultaneously. Hence, these cells will affect the rate of change of the state voltages of their neighbor cells. Since those cells with the structure of (j) in Fig. 16 (the corner cells) have the slowest rate (-0.75×10^6), the voltage values at the corner cells will become relatively, as time increases, larger than those of their neighbors during the transient evolution. This may cause their state voltage rates to eventually become positive. (In fact, this is exactly what happens in this example.) Since the voltages of the corner cells will increase after their rates become positive, their corresponding output voltages will eventually tend to 1.0 at the circuit steady state. In light of the above circuit dynamics, the result in Fig. 22 is not surprising.

Let us replace the Laplacian operator by the one in Fig. 17, and use the parameter $I=-2.0\times10^{-3}$ A. From the simulation results shown in Fig. 23, we can see that, in addition to extracting the edges of the square, we have also extracted four other points. This occurs because the derivatives of the state variables of the cells keep changing during the circuit transient. For instance, the derivative of the state voltage for the cell $C(6,7)$ in Fig. 23 is negative at the initial time, but it becomes positive after 15 time steps in the transient response. This in turn causes the state voltage of $C(6,7)$ to increase to its stable equilibrium state.

The cause of the above problem can therefore be traced to a lost of the original information of the image during the circuit transient. This observation suggests that if we choose a *nonzero* control operator in a cellular neural network and use the input image both as the input and as the initial condition of the circuit, then the above problem should not occur since the input (input image) will maintain the same value during circuit transients.

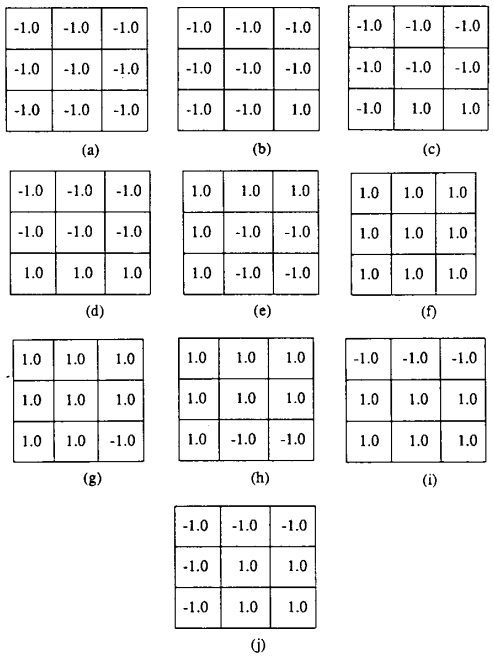

Fig. 16. Ten relationships for the cells with their neighbors.

0.0	-1.0	0.0
-1.0	4.0	-1.0
0.0	-1.0	0.0

Fig. 17. Cloning template defining the Laplacian operator. The unit used here is $10^{-3}\ \Omega^{-1}$

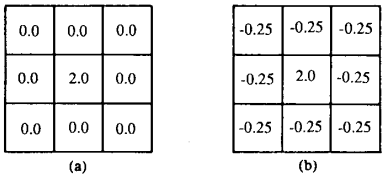

Fig. 18. (a) Closing template of the feedback operator for the edge detector. (b) Cloning template of the controlling (feedforward) operator for the edge detector.

7.3. *An Edge Detecting Cellular Neural Network*

Using our above experience, let us now design a more practical edge detector. In this edge detecting cellular neural network, both the feedback and control operators are nonzero; their cloning templates are shown in Figs. 18(a) and (b), respectively. The other circuit parameters are chosen as follows: $C=10^{-9}$ F; $R_x=10^3\ \Omega$; and $I=-1.5\times10^{-3}$ A.

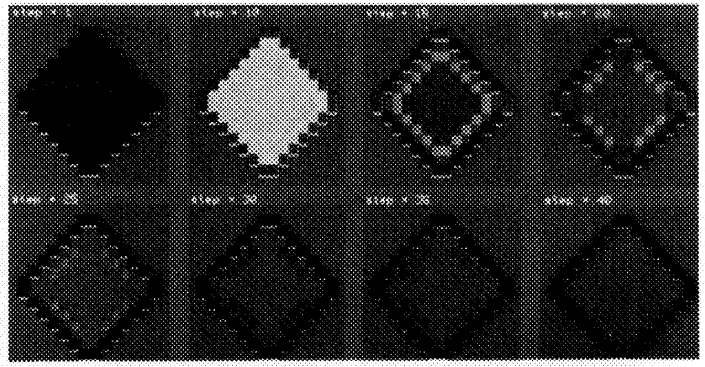

Fig. 19. Simulation results of the cellular neural network for extracting the edges of a diamond ($I = -1.75 \times 10^{-3}$A).

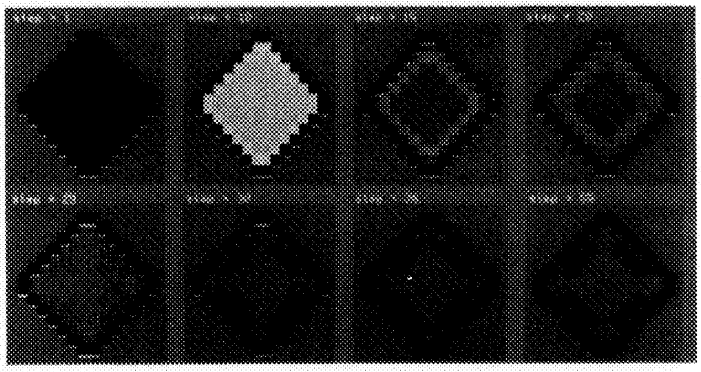

Fig. 20. Simulation results of the cellular neural network for extracting the edges of a diamond ($I = -1.5 \times 10^{-3}$A).

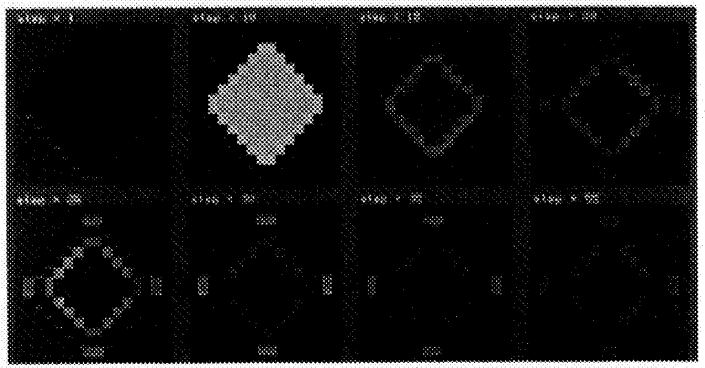

Fig. 21. Simulation results of the cellular neural network for extracting the edges of a diamond ($I = -2.0 \times 10^{-3}$A).

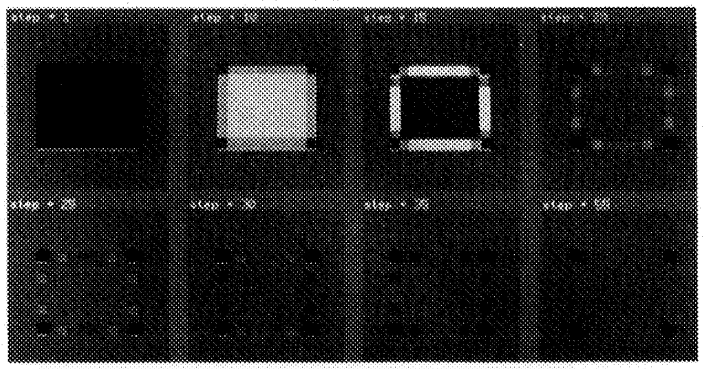

Fig. 22. Simulation results of the cellular neural network for extracting the corners of a square.

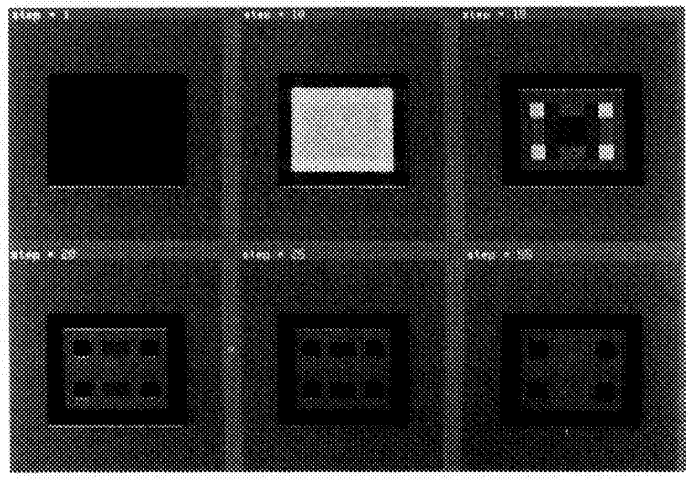

Fig. 23. Simulation results of the cellular neural network for extracting the edges of a square.

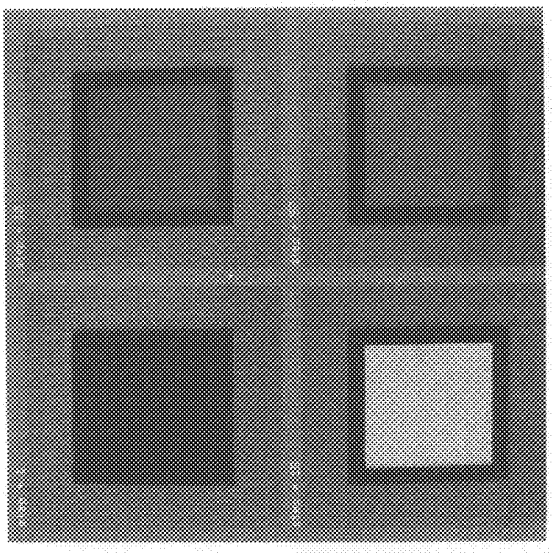

Fig. 25. Simulation results of the cellular neural network for extracting the edges of a square.

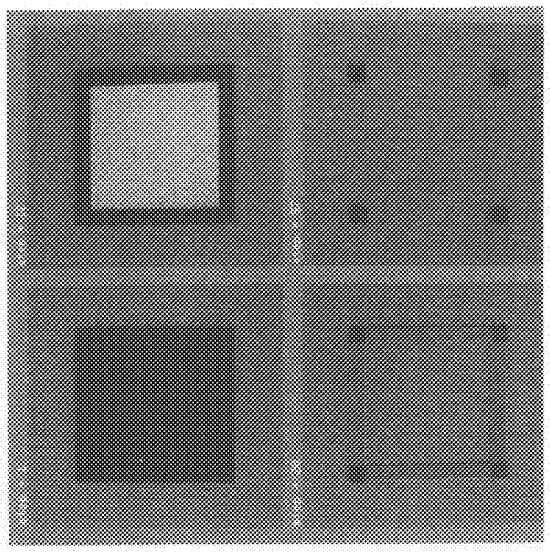

Fig. 27. Simulation results of the cellular neural network for extracting the corners of a square.

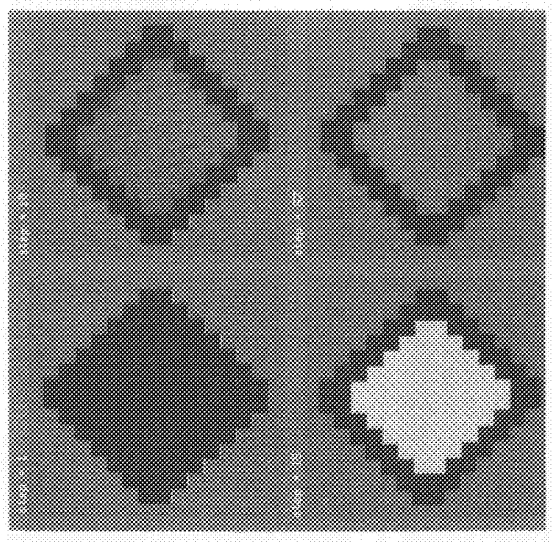

Fig. 24. Simulation results of the cellular neural network for extracting the edges of a diamond.

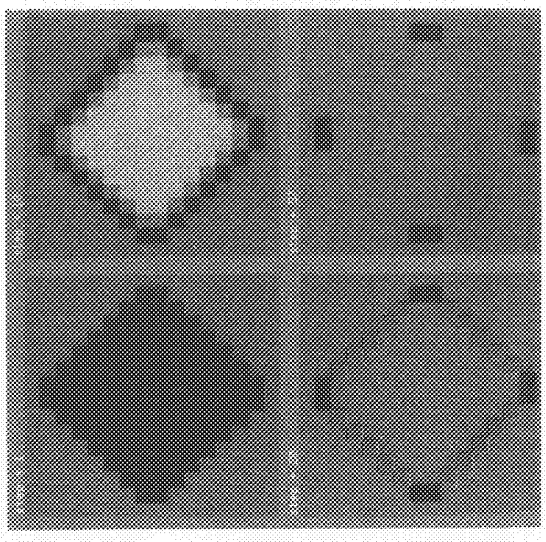

Fig. 26. Simulation results of the cellular neural network for extracting the corners of a diamond.

The circuit simulation results of this cellular neural network for detecting the edges of a diamond and a square are shown in Figs. 24 and 25, respectively, where the input images are chosen for both the inputs and the initial conditions of the edge detector. From the pictures in Figs. 24 and 25, we can see the perfect performance of this cellular neural network.

Note that, although we may use a digital computer to do the same job as the above edge detector, here the processing speed of the cellular neural network is much faster than that of the digital computers. For the circuit parameters chosen in this example, the processing speed is about 10^{-6} seconds. Furthermore, the processing speed of cellular neural networks is independent of the circuit size. This means it takes no longer to process a 512×512 image than one of 16×16 pixels.

7.4. A Corner Detecting Cellular Neural Network

To obtain a corner detecting cellular neural network, we change the circuit parameter I to -3×10^{-3} A and keep the other parameters the same as those in the edge detector designed in the preceding section. The circuit simulation results of this corner detector for detecting the corners of a diamond and a square are shown in Figs. 26 and 27, respectively.

VIII. Cellular Neural Networks for Chinese Character Recognition

There are about 60 000 Chinese characters and approximately 6000 of them are used in daily life. The prohibitively large number of Chinese characters makes the Chinese character recognition problem much more difficult than other character recognition problems. For the past twenty years, research on the Chinese character recognition problem has focused exclusively on algorithms using digital computers.

Theoretically, the Chinese character recognition problem has been solved for the 6000 most commonly used characters [19]. But the slow recognition speed continues to be the main problem in this area. To the best of our knowledge, the current average speed for Chinese character recognition using ordinary digital computers is about 2 characters per second, from a vocabulary consisting of 6000 basic characters. This recognition speed is much too slow for practical needs. The reason for this low recognition speed is that the nature of the algorithm for Chinese character recognition involves mainly parallel processing, because of the two-dimensional structure of the characters, but conventional digital computers are sequential processing machines.

From the above feature extraction examples, we have seen that cellular neural networks can extract certain features of images using appropriate dynamic rules. Consequently, the cellular neural network could be an efficient tool for solving the Chinese character recognition problem. First, we can design various cellular neural networks for extracting different features from Chinese characters. Then we can pass the character image simultaneously to all distinct feature extraction circuits in parallel. After the transient has settled down, (the time constants of cellular neural networks are generally less than 10^{-6} seconds) we would have extracted the different features of the original input character, which can then be used for higher level character recognition using a computer or any other kind of processing machine. As an example, we will present a simplified feature extraction circuit and use it to process a few simple Chinese characters.

The feature we want to extract here are the convex corners[3] of the strokes of Chinese characters. For this reason, we will use the corner detector designed in *Section 7.4* as our feature extracting cellular neural network. Figs. 28–35 are the circuit simulation results of 8 Chinese characters obtained by using our corner extraction circuit. From the simulation results, it can be seen that the convex corners of the strokes of the characters have been extracted. These corners contain most of the structural information of the characters, and can be used for coding the characters.

The purpose of these examples is to demonstrate the feature-extraction capability of cellular neural networks in image processing. What kind of structural features are useful for Chinese character recognition and how to extract them using cellular neural networks represent two important future research problems.

The resolution of the character images then is poor because of the small size of our pixel array. In Chinese character recognition, the typical pixel array for the character images is 48×48 or 64×64. Our experience shows that the larger the size of the cellular neural network, the better is the feature extraction capability for the characters due to increased spatial resolution. VLSI techniques will make it possible to implement large-sized cellular neural networks.

IX. Concluding Remarks

We have presented some applications of cellular neural networks in image processing and pattern recognition. For such applications, the cellular neural network functions as a two-dimensional filter. However, unlike the conventional two-dimensional *digital* filters, our cellular neural network uses *parallel* processing of the input image space and delivers its output in *continuous* time. This remarkable feature makes it possible to process a large-size image in real time. Moreover, the *nearest neighbor* interactive property of cellular neural networks makes them much more amenable to VLSI implementation.

All applications described in this paper assume an $N_r(i, j)$ neighborhood with $r = 1$; namely, the nearest neighbors. Other applications may call for a larger neighborhood with $r > 1$. In the extreme case where $r = M = N$, i.e., $N_r(i, j)$ is chosen to be the entire circuit, a cellular neural network may be interpreted as a *generalization* of a Hopfield neural network upon choosing $B(i, j; k, l) = 0$ and $I = I_i$ in (2a). Even in this case, the resemblance of (2a) to the well-known Hopfield model is only a superficial

[3]A corner in a pattern is said to be convex (resp., concave) if the area in the vicinity of the corner appears like a convex (resp., concave) object. For example, for the character in the upper-left hand corner in Fig. 28, only the "outer" corners in the character pattern are convex, as highlighted in the edges in the character pattern shown in the upper-right hand corner.

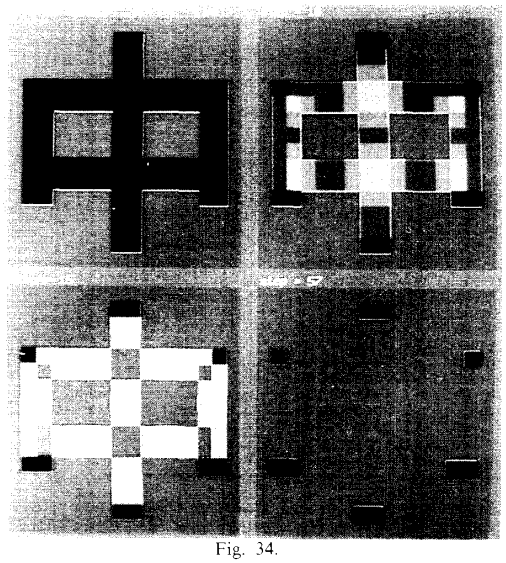

Fig. 34.

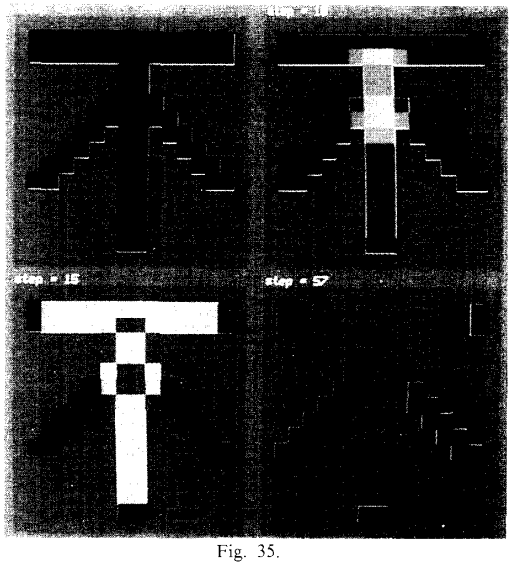

Fig. 35.

Simulation results of the cellular neural network for feature extraction of the Chinese characters.

one. To uncover the *fundamental* difference between these two models, observe that all current applications of the Hopfield model assumes *zero* "diagonal" coupling coefficients: i.e., $A(i, j; i, j) = 0$, whereas all current applications of the cellular neural network requires the condition

$$A(i, j; i, j) > \frac{1}{R_x} > 0$$

self-feedback in each neuron-like unit in the Hopfield as stipulated in (6). In other words, whereas there is no model, *every cell* in a cellular neural network requires a minimum amount of self-feedback. This self-feedback condition is in fact responsible for the extremely robust performance we have observed from cellular neural networks, where the output of each cell is guaranteed to be either $+1$ or -1, even though the slope of the *linear* region of the feedback nonlinearity is always unity. In sharp contrast, in order to achieve the same result in the Hopfield model, it is necessary that this slope approach *infinity* [4]. This "infinite-gain" condition is difficult to control in practice and is in fact responsible for various anomalous results (e.g., hysteresis, erratic outputs) that have been observed when a Hopfield model is operating under this condition.

APPENDIX

TABLE I

SIMPLE HORIZONTAL LINE DETECTOR: TYPICAL INPUT FILE FOR THE CIRCUIT SIMULATOR PWLSPICE

```
C110101          110101              0              1e-09
R110101          110101              0              1000
G1101010101  0  110101  210101  0          0.002
G1101010102  0  110101  210102  0          0.001
P210101          110101  210101     0              modpwl1
R210101          210101              0              1
C110102          110102              0              1e-09
R110102          110102              0              1000
G1101020101  0  110102  210101  0          0.001
G1101020102  0  110102  210102  0          0.002
G1101020103  0  110102  210103  0          0.001
P210102          110102  210102     0              modpwl1
R210102          210102              0              1
C110103          110103              0              1e-09
R110103          110103              0              1000
G1101030102  0  110103  210102  0          0.001
G1101030103  0  110103  210103  0          0.002
G1101030104  0  110103  210104  0          0.001
P210103          110103  210103     0              modpwl1
R210103          210103              0              1
C110104          110104              0              1e-09
R110104          110104              0              1000
G1101040103  0  110104  210103  0          0.001
G1101040104  0  110104  210104  0          0.002
P210104          110104  210104     0              modpwl1
R210104          210104              0              1
* This is a            PWL V            CCS model with a common node.
.model  modpwl1 pwl  term = 3 nseg = 3
+ ap= 0,0 bp= 0,0,0,0 cp= 0,0,-0.5,0.5 alphap= 1,0,1,0  betap=-1,1
.ic  v(110101)=-1  v(110102)=0.4  v(110103)=-0.8  v(110104)=-1
.tran   0.1us    5us    UIC
.print tran   v(210101) v(210102) v(210103) v(210104)
.end
```

The indexes of nodes are coded as follows: The first number from the left specifies the type of the nodes in a cell, 1 for the state voltage node, 2 for the output voltage node; the second number from the left specifies the layer number; the third and fourth numbers specify the rows; and the fifth and sixth numbers specify the columns. For example, **v**(110102) means v_{x12}, and **V**(210104) means v_{y14}, and so on.

TABLE II

SIMPLE HORIZONTAL LINES DETECTOR: TRANSIENT ANALYSIS. (THE INDEXES OF THE NODES ARE THE SAME AS THOSE IN TABLE I.)

Index	TIME	v(210101)	v(210102)	v(210103)	v(210104)
0	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+0
1	1.000000E-09	-1.000000E+00	3.985972E-01	-8.014028E-01	-1.000000E+
2	2.000000E-09	-1.000000E+00	3.971916E-01	-8.028084E-01	-1.000000E+
3	4.000000E-09	-1.000000E+00	3.943747E-01	-8.056253E-01	-1.000000E+
4	8.000000E-09	-1.000000E+00	3.887070E-01	-8.112930E-01	-1.000000E+
5	1.600000E-08	-1.000000E+00	3.772346E-01	-8.227654E-01	-1.000000E+
6	3.200000E-08	-1.000000E+00	3.537300E-01	-8.462700E-01	-1.000000E+
7	6.400000E-08	-1.000000E+00	3.043898E-01	-8.956102E-01	-1.000000E+
8	1.280000E-07	-1.000000E+00	1.957343E-01	-1.000000E+00	-1.000000E+
9	2.280000E-07	-1.000000E+00	5.811578E-03	-1.000000E+00	-1.000000E+
10	3.280000E-07	-1.000000E+00	-2.041030E-01	-1.000000E+00	-1.000000E+
11	4.280000E-07	-1.000000E+00	-4.361138E-01	-1.000000E+00	-1.000000E+
12	5.280000E-07	-1.000000E+00	-6.925469E-01	-1.000000E+00	-1.000000E+
13	6.280000E-07	-1.000000E+00	-9.759729E-01	-1.000000E+00	-1.000000E+
14	7.280000E-07	-1.000000E+00	-1.000000E+00	-1.000000E+00	-1.000000E+
15	8.280000E-07	-1.000000E+00	-1.000000E+00	-1.000000E+00	-1.000000E+
16	9.280000E-07	-1.000000E+00	-1.000000E+00	-1.000000E+00	-1.000000E+
17	1.028000E-06	-1.000000E+00	-1.000000E+00	-1.000000E+00	-1.000000E+
18	1.128000E-06	-1.000000E+00	-1.000000E+00	-1.000000E+00	-1.000000E+
19	1.228000E-06	-1.000000E+00	-1.000000E+00	-1.000000E+00	-1.000000E+
20	1.328000E-06	-1.000000E+00	-1.000000E+00	-1.000000E+00	-1.000000E+

REFERENCES

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, this issue.

[2] J. J. Hopfield, "Neural networks and physical systems with emergent computational abilities," *Proc. Natl. Acad. Sci. USA.*, vol. 79, pp. 2554–2558, 1982.

[3] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Bio. Cybern.*, vol. 52, pp. 141–152, 1985.

[4] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: a model," *Science (USA)*, vol. 233, no. 4764, pp. 625–633, 1986.

[5] D. W. Tank and J. J. Hopfield, "Simple 'neuron' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, CAS-33, pp. 533–541, 1986.

[6] S. Wolfram (Eds), *Theory and Applications of Cellular Automata*. New York: World Scientific, 1986.

[7] N. Packard and S. Wolfram, "Two-dimensional cellular automata," *J. Stat. Phys.*, vol. 38, nos. 5/6, pp. 901–946, 1985.

[8] T. Toffoli and N. Margolus, *Cellular Automata Machines — A New Environment for Modeling*. Cambridge, MA: M.I.T. Press, 1986.

[9] K. Preston, Jr. and M. J. B. Duff, *Modern Cellular Automata: Theory and Applications*. New York, Plenum, 1984.

[10] Special Issue on Chaotic Systems, *Proc. IEEE*, Aug. 1987.

[11] L. O. Chua and R. N. Madan, "The sights and sounds of chaos," *IEEE Circuits Devices Mag.*, pp. 3–13, Jan. 1988.

[12] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York: Academic, 1982.

[13] B. K. P. Horn, *Robot Vision*. Cambridge, MA: M.I.T. Press, 1986.

[14] L. O. Chua and P. M. Lin, *Computer Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[15] L. O. Chua and R. L. P. Ying, "Finding all solutions of piecewise-linear circuits," *J. Circuit Theory Appl.*, vol. 10, pp. 201–229, 1982.

[16] L. O. Chua and A. C. Deng, "Canonical piecewise-linear analysis: Generalized breakpoint hopping algorithm," *Int. J. Circuit Theory Appl.*, vol. 14, pp. 35–52, Jan. 1986.

[17] L. O. Chua and Y. Liao, "PWLSPICE: SPICE for piecewise-linear circuits," in preparation.

[18] T. Quarles, A. R. Newton, D. O. Pederson, and A. Sangiovanni-Vincentelli, "SPICE.3A7 User's Guide," Dept. Elect. Eng. and Comp. Sci., Univ. of California, Berkeley, June 1986.

[19] X. Zhu, Y. Wu, and X. Ding, "The recognition of 6763 printed Chinese characters," *J. Tsinghua Uni.*, vol. 27, no. 1, pp. 39–49, 1987.

✠

L. O. Chua (S'60–M'62–SM'70–F'74), for a photograph and biography please see page 880 of the July 1988 issue of this TRANSACTIONS.

✠

Lin Yang (S'87), for a photograph and biography please see page 1272 of this issue.