Journal of Circuits, Systems, and Computers
Vol. 12, No. 4 (2003) 473–490
© World Scientific Publishing Company

CHARACTERISTIC OF MUTUALLY COUPLED TWO-LAYER CNN AND ITS STABILITY

ZONGHUANG YANG, YOSHIFUMI NISHIO, and AKIO USHIDA

Department of Electrical and Electronic Engineering, Tokushima University, Minami-Josanjima 2-1, Tokushima, 770-8506, Japan

This paper presents some interesting image processing applications with the mutually coupled two-layer Cellular Neural Networks (CNNs). We found that the two-layer CNNs are very useful compared to single layer CNNs in some applications such as center point detection, skeletonization, and so on. We also focus our discussions on both their transients and operations. In addition, the stability of the two-layer CNNs with mutually coupled symmetric templates is also discussed based on those of decoupling CNN technique.

Keywords: Two-layer CNN; template; center point detection; skeletonization; stability.

1. Introduction

Since the Cellular Neural Networks (CNNs) model was invented,[1] due to the desirable feature such that their cells are analog process units computing in real time and arranged in two or more dimensional grid points with the same local coupling manner, the CNNs have been rapidly developed as various high-speed parallel signal-processing platforms. They have been successfully used for image processing[2–10] and as a model of physical phenomena.[11–17]

In this paper, we present a new technique, which is based on the mutually coupled characteristic between the two layers, for several important image processing tasks such as skeletonization, center point detection, etc. Observe that their solutions can be easily realized on the current CNN Universal Machine (CNNUM).[1,2]

Many image processing and pattern recognition applications are based on center point detection, skeletonizing, etc. Of course, several methods for center point detection and skeletonization have already been described in several papers. In the literature,[5–7] their algorithms were carried out with complex multi-layer, and sometimes even nonuniform CNN architectures. All of them are impractical, because it is difficult to build the CNNUM chips by using current VLSI technologies. There are also other methods based on the single layer CNNs,[9,10] but the final solutions can be obtained only by the iterative use of different templates, where each single-layer CNN is iteratively used to perform a part of the task. Namely, after the operation of CNNs has attained the steady state or reached some state, the next single-layer CNN begins to perform the next part of the task. In this way, the process iteratively continues until the whole task is completed. It is obvious that this procedure is tedious and belongs to the serial processing manner. Therefore, how to get a simple and efficient method that can be realized on the CNNUM chip using current VLSI technologies is important. Our method in this paper may give the solution of the implementation.

This paper is organized as follows: In Sec. 2, a mutually coupled two-layer cellular neural network model is described as the extension of the single-layer CNN,[1] in which two mutually coupled templates between two layers are introduced. In Sec. 3, an algorithm of the two-layer CNN for center point detection of a line is discussed in detail, which is the fundamental idea throughout the paper. Moreover, we propose the feasibility of the two-layer CNNUM chip. The applications of the proposed algorithm to center point detection, skeletonization, etc. are described in Sec. 4. In Sec. 5, the stability of the two-layer CNNs with mutually coupled symmetric templates is also discussed based on the decoupling technique.

2. Structure of Two-Layer CNNs

The concept of the two and more layer CNN structures have already been proposed. The earliest multi-layer CNN model[1] was proposed by Chua and Yang, it was a feedforward multi-layer CNN structure. In the literature,[11–17] the CNNs' cells are composed of Chua's oscillators, reduced Chua's circuits or second nonlinear oscillators. These CNNs are mainly used in generating nonlinear phenomena such as autowaves, pattern formation and so on. A unified notation for representing the CNN topology was proposed in the literature,[18] and other two-layer CNN versions were introduced for individual different applications.[19–21]

The mutually coupled two-layer CNNs in this paper are described as the extension of the single layer continuous time CNN,[1] their system equations are formulated by introducing two mutually coupled templates C_1 and C_2. We assume that the two-layer CNNs are also composed of two-dimensional M by N array structures as shown on the left hand side of Fig. 1. Each cell in the array is denoted by $c(i, j)$, its local coupling is shown on the right hand side of Fig. 1. The cell has two state variables, $x_1(i, j)$ and $x_2(i, j)$, where (i, j) stands for the position of a cell in the array, for $1 \le i \le M$ and $1 \le j \le N$. The state equations of each cell are given by two first-order differential equations given by Eq. (1), and the output equations are given by Eq. (2), where $f(\cdot)$ is a piecewise-linear nonlinear function defined by Eq. (3). We define state variables of the first layer by $x_1(i, j)$, and those of the second layer by $x_2(i, j)$. u and y refer to the input and output variables of the cell. $A(i, j; k, l)$, $B(i, j; k, l)$, and I are the feedback template, filter template and bias current, respectively. The index 1 and 2 stand for first layer and second layer of the two-layer CNN array. $C_1(i, j; k, l)$ is the coupling template to transfer the second layer output to the first layer input, and vice versa for $C_2(i, j; k, l)$. They also show

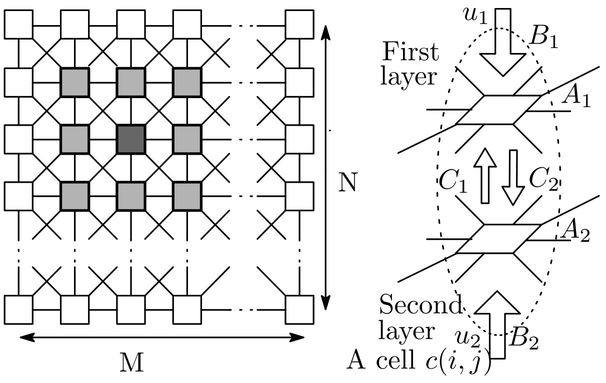

Fig. 1. A two-dimensional cellular neural network array.

the weight of the local couplings among the cells in the neighborhood $N_r(i,j)$ as the templates $A(i,j;k,l)$ and $B(i,j;k,l)$. r is the coupling radius. In general, r takes the value of 1 or 2. An example of $r = 1$ for a particular cell is shown by the gray cells in Fig. 1.

$$\begin{aligned}
\dot{x}_{1,ij} &= -x_{1,ij} + I_1 + \sum_{C(k,l)\in N_r(i,j)} A_1(i,j;k,l)y_{1,kl} \\
&\quad + \sum_{C(k,l)\in N_r(i,j)} B_1(i,j;k,l)u_{1,kl} \\
&\quad + \sum_{C(k,l)\in N_r(i,j)} C_1(i,j;k,l)y_{2,kl}\,, \\
\dot{x}_{2,ij} &= -x_{2,ij} + I_2 + \sum_{C(k,l)\in N_r(i,j)} A_2(i,j;k,l)y_{2,kl} \\
&\quad + \sum_{C(k,l)\in N_r(i,j)} B_2(i,j;k,l)u_{2,kl} \\
&\quad + \sum_{C(k,l)\in N_r(i,j)} C_2(i,j;k,l)y_{1,kl}\,,
\end{aligned} \tag{1}$$

$$\begin{aligned}
y_{1,ij} &= f(x_{1,ij})\,, \\
y_{2,ij} &= f(x_{2,ij})\,,
\end{aligned} \tag{2}$$

$$f(x) = 0.5(|x+1| - |x-1|)\,. \tag{3}$$

Figure 2 shows the block diagram of a two-layer CNN. Obviously, when both templates C_1 and C_2 are zero, the CNN becomes two independent single-layer CNNs, and when one of the templates is zero, the CNN becomes an open-loop system behaving as the cascade connections of two single-layer CNNs, i.e., feedforward

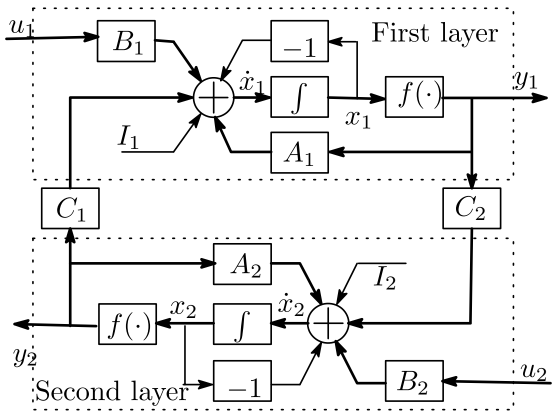

Fig. 2. Block diagram of the two-layer CNN.

two-layer CNN. Thus, we can conclude that the stability conditions and templates developed in single layer CNN are also valid for the above two cases of two-layer CNNs. Moreover, some solutions of image processing, which we need two or more steps for solving with single layer CNNs, can be obtained in one step with the two-layer CNNs. Their differences are that the former is sequential and the latter is a parallel signal processing. It's worth our attention that, if both templates C_1 and C_2 are not zero, the two layers of the CNN constitute a closed-loop system; namely, one layer output is fed into the other layer as time-varying input through templates C_1 and C_2. The mutually coupled characteristic between two layers provides some new methods for image processing. The two layers share out some tasks of image processing and cooperate in each other, so that the two-layer CNNs can execute these tasks more efficiently than single-layer CNN. Therefore, the two-layer CNNs show a real potential with this structure.

3. A Method Based on the Mutually Coupled Characteristic

In many applications of image processing and pattern recognition, center point detection and skeletonization are two very important central tasks. They have already been attacked in several papers,[5–9] but all of the solutions presented so far use complex multi-layer, time-variant templates, and sometimes even nonuniform architecture. All of them are tedious for the selection of the templates and impractical for use of the current CNNUM chip. Here, based on the mutually coupled characteristic between two layers in the two-layer CNNs, we propose a new method from an entirely different viewpoint, which overcomes the above defects.

The guideline of our method for obtaining the final solutions of the skeletonization or center point detection of objects can be simply expressed as: continuously and automatically peeling-off pixels from the objects or strokes in all directions until the expected results can be obtained in the stable states. Now, let us see how

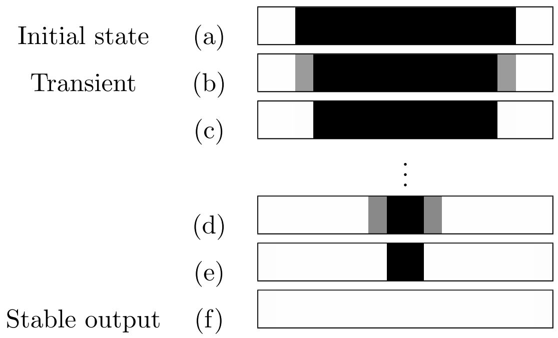

Fig. 3. An example of continuously peeling-off pixels from two ends in a single layer CNN.

to realize the continuous and automatic process and obtain the expected results. For simplification, we consider a simple example of the center point detection of a horizontal line. We define "the center point as located halfway from the furthermost ends of a given line".

3.1. *Single layer CNNs to peeling-off problem*

Before discussing our new method, let us recall the process of peeling-off pixels in all directions in single layer CNN. As is well known, the peeling-off of two-end pixels for a given line can be executed by the template (4).[9] The given image is set to the input of a CNN. After the CNN is attained to the stable state, two-end pixels of the line are peeled-off. This method of using template B multiplied by the input image can only execute two-end pixels peeling-off.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 2 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}, \; I = -4\,. \tag{4}$$

In order to automatically and continuously peel off two pixels from the two ends of the line, we need to have the aid of the self-feedback characteristic of the template A. Thus, the template is described as Eq. (5). Figure 3 shows the simulation results, where Fig. 3(a) is the given binary image that is set as the initial state of the CNN, Figs. 3(b)–3(e) are the transient results, Fig. 3(f) is the stable output. From this simulation, we observed that the single layer CNN could realize the automatic and continuous process. Unfortunately, the expected solution cannot be obtained from the CNN as the stable output. The result can also be confirmed theoretically.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 2 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = 0,\; I = -1.5\,. \tag{5}$$

3.2. *Two-layer CNNs to peeling-off problem*

Now, let us discuss the same problem using the two-layer CNNs. Consider two mutually coupled templates to realize the automatic process; namely, C_2 template transfers the first layer output to the second layer input, and vice versa for the C_1 template. We set the given image Fig. 3(a) to the initial states of both CNN layers as shown by Fig. 4(a), and adopt the template as follows:

$$A_1 = 2,\ A_2 = 2,\ B_1 = 0,\ B_2 = 0,\ I_1 = -2.5,\ I_2 = -0.5,$$
$$C_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0 & 0 \end{bmatrix},\quad C_2 = \begin{bmatrix} 0 & 0 & 0 \\ -0.5 & 1 & -0.5 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6}$$

Thus, the cell's state equations in the two-layer CNN can be rewritten in the following forms:

$$\begin{aligned} \dot{x}_{1;i,j} &= -f_1(x_{1;i,j}) + g_{1;i,j}\,, \\ \dot{x}_{2;i,j} &= -f_2(x_{2;i,j}) + g_{2;i,j}\,, \end{aligned} \tag{7}$$

where

$$\begin{aligned} f_1(x_{1;i,j}) &= x_{1;i,j} - (|x_{1;i,j}+1| - |x_{1;i,j}-1|)\,, \\ f_2(x_{2;i,j}) &= x_{2;i,j} - (|x_{2;i,j}+1| - |x_{2;i,j}-1|)\,, \\ g_{1;i,j} &= 0.5y_{2;i,j-1} + y_{2;i,j} + 0.5y_{2;i,j+1} - 2.5\,, \\ g_{2;i,j} &= -0.5y_{1;i,j-1} + y_{1;i,j} - 0.5y_{1;i,j+1} - 0.5\,. \end{aligned}$$

For the self-feedback coefficients $a_{1,00} = 2$ and $a_{2,00} = 2$, the trajectory of the state variables without $g_{1;i,j}$ and $g_{2;i,j}$ behaves as the solid line in Fig. 4. The trajectories of Eq. (7) can be obtained by shifting the $g_{1;i,j}$ and $g_{2;i,j}$ values up and down as shown by the dotted lines in the figure. Observe that if we choose

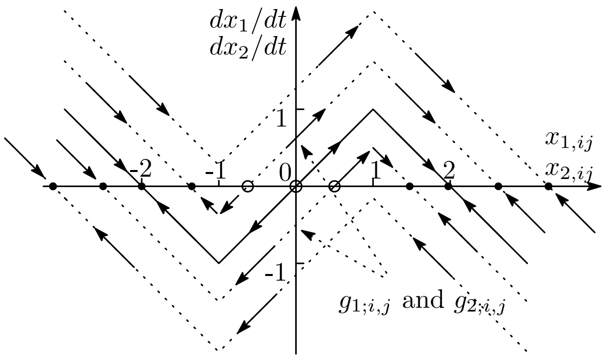

Fig. 4. The trajectories of state variables for various $g_{1;i,j}$ and $g_{2;i,j}$, where the stable and unstable equilibrium points are denoted by the solid dots and circles, respectively.

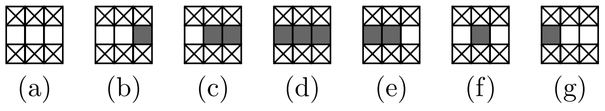
Fig. 5. All possible landscapes of a cell at the beginning.

the initial conditions $x_{1;i,j}(0)$ and $x_{2;i,j}(0)$ larger than the unstable equilibrium points shown by circles, the steady-state output will be given by 1, and otherwise the output by -1. Thus, we have the following output relations depending on the initial conditions.

$$\begin{aligned} y_{1;i,j} &= \operatorname{sgn}[x_{1;i,j}(0) + g_{1;i,j}]\,, \\ y_{2;i,j} &= \operatorname{sgn}[x_{2;i,j}(0) + g_{2;i,j}]\,. \end{aligned} \tag{8}$$

To analyze the stability of the equilibrium point of a cell in the two-layer CNN, we restrict our discussion to a landscape of a cell by listing all those neighboring cell states. Since both layer initial states are the same at the beginning of this simulation, the landscape corresponding to Fig. 7(a) can be divided into seven possible cases as shown in Fig. 5, where the black squares denote cells having value 1 and the white squares value -1. On the other hand, the crossed square cells stand for "don't care" for the center cell, because the template coefficient in Eq. (6) corresponding to those positions are zero.

For the case of Fig. 5(a), we calculate

$$\begin{aligned} g_{1;i,j} &= 0.5y_{2;i,j-1} + y_{2;i,j} + 0.5y_{2;i,j+1} - 2.5 \\ &= 0.5 \times (-1) + (-1) + 0.5 \times (-1) - 2.5 \\ &= -4.5\,, \\ g_{2;i,j} &= -0.5\,. \end{aligned}$$

Thus, we have $y_{1,ij} = -1$ (white) and $y_{2,ij} = -1$ (white) for this case from Eq. (8). Similarly, for other cases from Figs. 5(b)–5(g), the variation of the cell outputs can be obtained as listed in Tables 1 and 2.

It is observed from Tables 1 and 2 that the outputs of the first layer center cell, only in cases (c) and (e), will be changed from 1 (black) to -1 (white), and

Table 1. The landscape and the output variation of a cell in the first layer.

Landscape	$g_{1;i,j}$	Initial state	Output
case (a)	-4.5	-1	-1
case (b)	-3.5	-1	-1
case (c)	-1.5	1	-1
case (d)	-0.5	1	1
case (e)	-1.5	1	-1
case (f)	-3.5	-1	-1
case (g)	-2.5	1	-1

Table 2. The landscape and the output variation of a cell in the second layer.

Landscape	$g_{2;i,j}$	Initial state	Output
case (a)	−0.5	−1	−1
case (b)	−1.5	−1	−1
case (c)	0.5	1	1
case (d)	−0.5	1	1
case (e)	0.5	1	1
case (f)	−1.5	−1	−1
case (g)	1.5	1	1

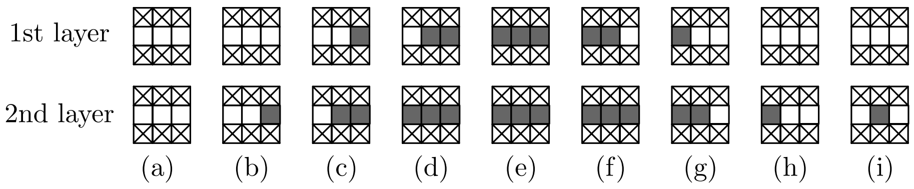

Fig. 6. All possible landscapes of a cell at a transient.

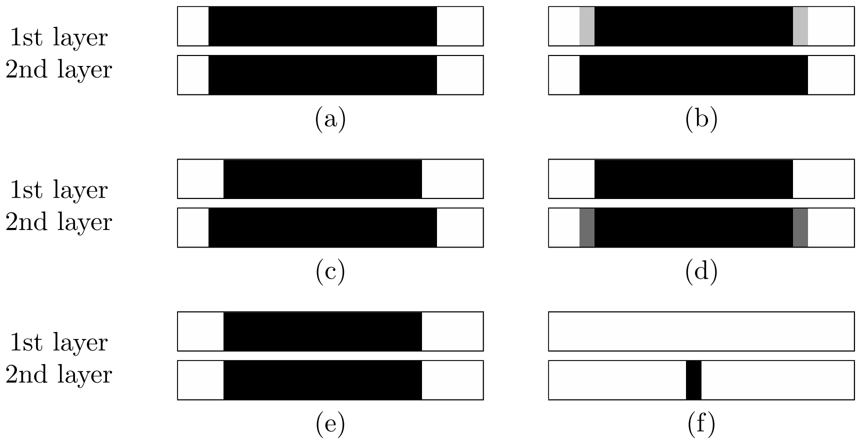

Fig. 7. An example of detecting the center point of a given line.

unchanged for the other cases. In other words, at the beginning, the first layer of the CNN performs a peeling-off of the leftmost and rightmost pixels of the line corresponding to the process of Figs. 7(a)–7(c). In this period, the output of the second layer remains the same. After that, the landscape corresponding to Fig. 7(c) can be divided into nine possible cases as shown in Fig. 6. By the similar analysis, we can conclude that, only in the cases Figs. 6(c) and 6(g), are the outputs of the second layer center cells changed from 1 (black) to −1 (white). Namely, the second layer performs a peeling-off of the leftmost and rightmost pixels of the line

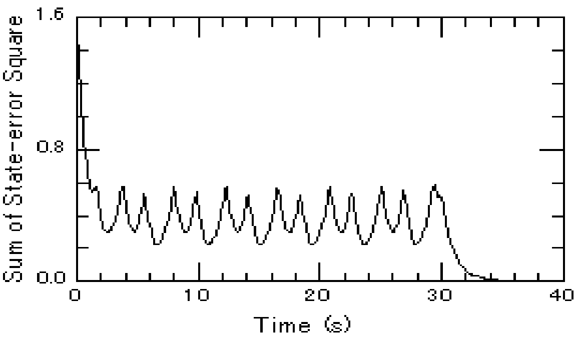

Fig. 8. Graph of sum of State-error Square via time for center point detection.

corresponding to Figs. 7(c)–7(e), but the output of the first layer remain the same in this time. Figure 7(e) is the same as Fig. 7(a), except for both the end pixels which have been peeled off. So far one cycle of the peeling-off is completed. This cycle continues automatically until the case corresponding to Fig. 6(i) appears, where the outputs of both layers do not change again and the transient has arrived at the steady state corresponding to Fig. 7(f). Figure 7 shows the simulation for center point detection of a line, which agrees with our analysis. This dynamic process can be observed from the graph of the sum of State-error Square varying with the integration time shown in Fig. 8. The sum of State-error Square is defined as:

$$S_{\mathrm{CNN}}(t) = \sum_{i=1}^{M}\sum_{j=1}^{N}((x_{1,ij}(t+\Delta t) - x_{1,ij}(t))^2 + (x_{2,ij}(t+\Delta t) - x_{2,ij}(t))^2)\,. \tag{9}$$

The first peak shows the state corresponding to Fig. 7(c), the second peak is corresponding to Fig. 7(e) and so on. When the line is completely peeled into center point, the sum of State-errors Square decays to zero. The two layer stable outputs are shown by Fig. 7(f), in this simulation, the zero-fixed boundary condition is adopted. From the above simulation result, we have shown that the problem of center point detection of a line can be solved in single step.

4. Applications

It is an important problem to detect the center point and the skeleton of a given object or character, because many image processing and pattern recognition applications are based on them. In this section, we will apply the above method to center point detection, skeletonization, etc.

4.1. *Center point detection*

It is very important to detect the center point of a given object, because the point can be used as the reference position of an object. Unfortunately, the definition

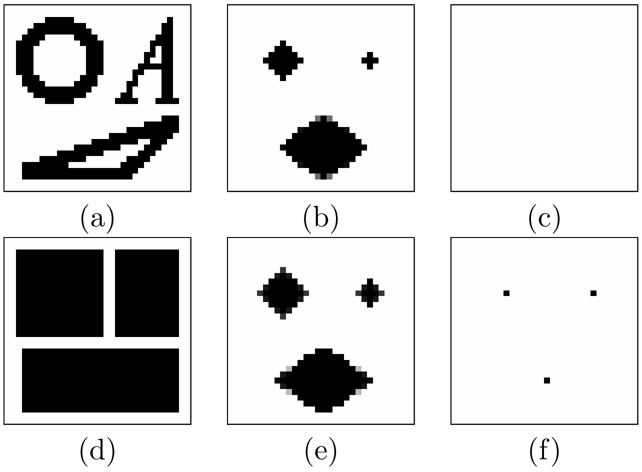

Fig. 9. An example of detecting the center points of three objects with different shapes.

of the center point is ambiguous, because there are various objects such as convex, concave, disk with embedded holes, etc. In this paper, we define "the center point as located halfway from the furthermost points of a given object in the horizontal and vertical directions". Therefore, without a loss of generality, we can consider that the object image is a rectangular block such as Fig. 9(d), because arbitrary shape objects such as Fig. 9(a) can be changed into rectangular blocks with the same length and width by using the shadow template.[9] Thus, the center point detection is reduced to the problem of finding out the center point of a rectangular object, and we can easily solve this problem with the same technique proposed in the last section.

It is obvious that we can obtain the solution in two steps, i.e., the first step is to detect the center line of the object by the template (6), and the second is to detect the center point of the above centerline by the transposed template (6). Moreover, if we consider simultaneously peeling-off pixels from four corner directions of the object, then the solution can be obtained in one step. A simulation for center point detection is shown in Fig. 9, in which a given image shown by Fig. 9(a) includes three objects with different shape. The way of operation is: Firstly, the three object are respectively changed into three rectangular blocks shown by Fig. 9(d), then the image including three blocks is set as the initial states of both CNN layers, and the template (10) is adopted. Thus, after the CNN attains the steady-state, we can obtain the expected result from the output of the second layer shown by Fig. 9(f), where three points are the solutions corresponding to three objects, respectively. Figures 9(b) and 9(e) are respectively the transient outputs of the first and second layers. The first layer becomes the empty stable output as shown by Fig. 9(c).

$$A_1 = 2,\ A_2 = 2,\ B_1 = 0,\ B_2 = 0,\ I_1 = -3.5,\ I_2 = -0.5\,,$$
$$C_1 = \begin{bmatrix} 0 & 0.5 & 0 \\ 0.5 & 2 & 0.5 \\ 0 & 0.5 & 0 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 0 & -0.5 & 0 \\ -0.5 & 2 & -0.5 \\ 0 & -0.5 & 0 \end{bmatrix}. \tag{10}$$

4.2. *Skeletonization*

The problem of skeletonization is also a very important task. For example, it is used as a subroutine of several image processing and pattern recognition problems, such as character recognition. Like the center point of an object, the definition of the skeleton of objects is also ambiguous. Here, the solution of skeletonization is located in the central position of strokes or lines. Thus, we can simultaneously peel off pixels from the object in all directions with a method similar to the one proposed above. The solution can be obtained in one step. Figure 10 shows an example for skeletonization, in which the given binary image shown by Fig. 10(a) includes two characters — the Chinese character "Wang" with seven pixels width and the English character "A" with 5 pixels. Similarly, we set the binary image as the initial states of both two CNN layers, and adopt the template (11). After the CNN attains the steady-state, the expected solution can be obtained from the second layer stable output shown by Fig. 10(h). Transients of the simulation can be observed from Fig. 10, in which Figs. 10(b) and 10(e) is a couple of the transient outputs of the first and second layers at a time, and Figs. 10(c) and 10(g) is another couple of the transients. Figure 10(d) is the stable output of the first layer.

$$A_1 = 2,\ A_2 = 2,\ B_1 = 0,\ B_2 = 0,\ I_1 = 1.8,\ I_2 = -1.8,$$
$$C_1 = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & -1.7 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}, \quad C_2 = \begin{bmatrix} -0.5 & -0.5 & -0.5 \\ -0.5 & 4.2 & -0.5 \\ -0.5 & -0.5 & -0.5 \end{bmatrix}. \tag{11}$$

Figure 11 shows another example for skeletonizing a handwritten Chinese character "Ma", in which Fig. 11(a) is the original image, Fig. 11(b) is the binary image which is set as the initial states of both layers. Figures 11(c)–11(e) are three transients observed from the second layer output through progressing time.

Through the above two applications, we have shown that the mutually coupled two-layer CNNs can efficiently solve some problems of image processing compared to the single layer CNN. On the other hand, in the above two-layer CNNs, all of their B templates are set to zero. Observe that the C templates work as B templates in each layer, because one layer output transfers to the other layers input through the template C. Therefore, it is possible to realize the above CNNUMs based on the current single layer CNNUM chip.

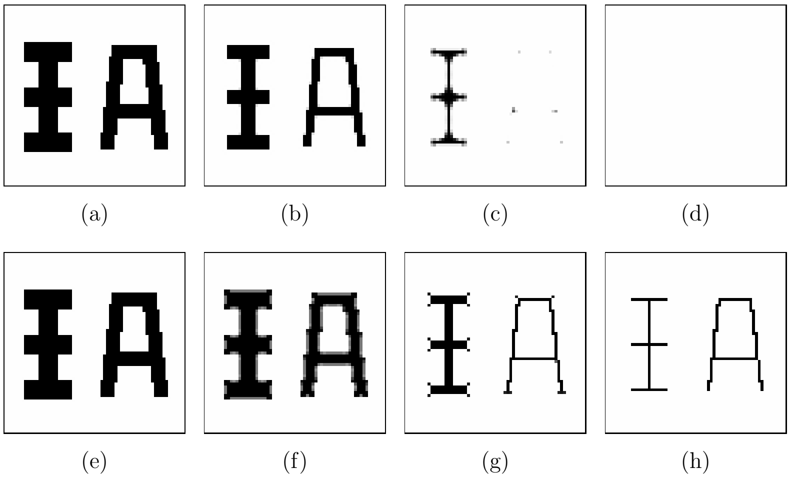
Fig. 10. An example of skeletonization.

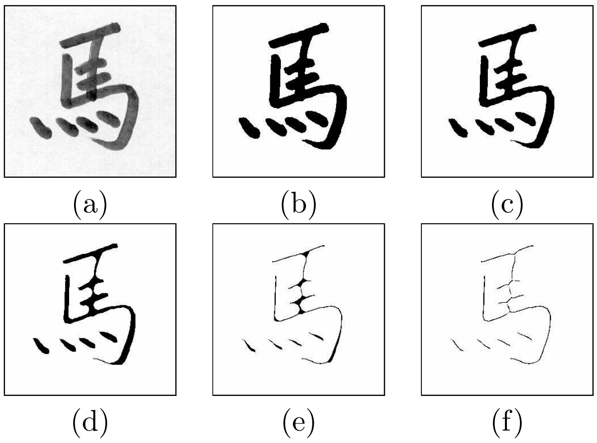
Fig. 11. An example of skeletonizing a handwritten Chinese character.

4.3. *Dividing object into halves*

Now let us consider an interesting problem to divide object into the halves (or two parts with the same area) that cannot be solved by the single layer CNNs. This task can be executed similarly based on the mutually coupled characteristic of the two-layer CNN. The basic idea of the algorithm is that, the two layers of the CNN are respectively and simultaneously used to peel the pixels from two sides of the object. After the process reaches the centerline, the two-layer CNN arrives at the steady state, thus the object is divided into two parts. The template are given as

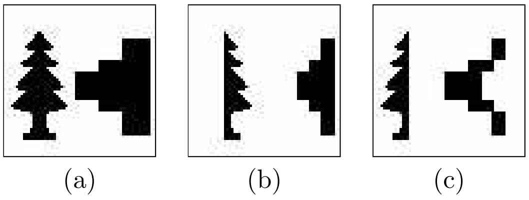
Fig. 12. An example of dividing object into halves.

the follows:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}, \ B_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \ C_1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}, \ I_1 = -2,$$
$$A_2 = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \ B_2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ C_2 = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \ I_2 = -2\,. \tag{12}$$

In this case, the given binary image shown in Fig. 12(a) is set to all the inputs and initial states of the CNN. Thus, we obtain the two different parts with the same area in the horizontal from the given object as the two layer outputs of the CNN, which are shown in Figs. 12(b) and 12(c). Moreover, we can see from the figures that the right object in Fig. 12(b) is the solution obtained by compressing the original object toward the right direction without changing its outline. These properties may offer some interesting applications in the future.

5. Stability of the Mutually Coupled Two-Layer CNNs

In most applications of CNNs such as image processing, the CNNs are required to be completely stable. Therefore, stability is one of the most important dynamical properties of CNNs. In the last decade, the studies on complete stability have been vigorously discussed and many criteria have been obtained in single-layer CNNs and DCNNs.[1,10,22–25] We need to discuss the stability of the mutually coupled two-layer CNNs for image processing. We have already been discussed the stability of two-layer CNNs in previous works[10] with Lyapunov's method, and found that the obtained stability conditions are rather restricted for some special applications. Thus, it is important to obtain milder conditions for complete stability of two-layer CNNs. In this section, we will analyze the stability of two-layer CNNs with mutually coupled symmetric templates by using the decoupling technique.[13]

Firstly, let us consider a simple two-layer CNN without introducing inputs and bias currents. Its equations are described in the following form:

$$\begin{aligned} \dot{x}_{1,ij} &= -x_{1,ij} + (a_1+1)y_{1,ij} + c_1 y_{2,ij} + D_1\nabla^2 y_{2,ij}\,, \\ \dot{x}_{2,ij} &= -x_{2,ij} + (a_2+1)y_{2,ij} + c_2 y_{1,ij} + D_2\nabla^2 y_{1,ij}\,, \end{aligned} \tag{13}$$

where a_1, a_2, c_1, c_2, D_1 and D_2 are parameters. By expanding the Laplacian operator in discrete form, Eq. (13) can be rewritten as:

$$\begin{aligned}\dot{x}_{1;ij} &= -x_{1;ij} + (a_1+1)y_{1;ij} + c_1 y_{2;ij}\\ &\qquad + D_1(y_{2;i-1,j} + y_{2;i+1,j} + y_{2;i,j-1} + y_{2;i,j+1} - 4y_{2;ij})\,,\\ \dot{x}_{2;ij} &= -x_{2;ij} + (a_2+1)y_{2;ij} + c_2 y_{1;ij}\\ &\qquad + D_2(y_{1;i-1,j} + y_{1;i+1,j} + y_{1;i,j-1} + y_{1;i,j+1} - 4y_{1;ij})\,.\end{aligned} \tag{14}$$

Comparing it with Eq. (1), the two-layer CNN has the following template:

$$\begin{aligned} A_1 &= \begin{bmatrix}0 & 0 & 0\\ 0 & a_1+1 & 0\\ 0 & 0 & 0\end{bmatrix}, \; B_1 = 0\,, \quad C_1 = \begin{bmatrix}0 & D_1 & 0\\ D_1 & c_1-4D_2 & D_1\\ 0 & D_1 & 0\end{bmatrix}, \; I_1 = 0\,,\\ A_2 &= \begin{bmatrix}0 & 0 & 0\\ 0 & a_2+1 & 0\\ 0 & 0 & 0\end{bmatrix}, \; B_2 = 0\,, \quad C_2 = \begin{bmatrix}0 & D_2 & 0\\ D_2 & c_2-4D_2 & D_2\\ 0 & D_2 & 0\end{bmatrix}, \; I_2 = 0\,.\end{aligned} \tag{15}$$

On the other hand, the Laplacian operator in Eq. (13) acts on the output cell variables instead of the state variables, and each output cell is a piecewise-linear nonlinear function of the cell state. If we assume that all initial states of the system at the beginning are always in the linear region, the Laplacian operators can be considered to be directly used to modulate the state variables, and the state Eqs. (13) are rewritten as follows:

$$\begin{aligned}\dot{x}_{1,ij} &= a_1 x_{1,ij} + c_1 x_{2,ij} + D_1 \nabla^2 x_{2,ij}\,,\\ \dot{x}_{2,ij} &= a_2 x_{2,ij} + c_2 x_{1,ij} + D_2 \nabla^2 x_{1,ij}\,,\end{aligned} \tag{16}$$

where $x_{1;ij}$, $x_{2;ij}$ are in linear region, and $1 \le i \le M$, $1 \le j \le N$.

Thus, the above linear differential equation can be solved by decoupling it into MN-decoupled systems of two first-order linear differential equations, and considering that the MN orthonormal space-dependent eigenfunction $\phi_{MN}(m,n;i,j)$ of the discrete Laplacian operator can be assumed as follows for most boundary conditions:

$$\nabla^2 \phi_{MN}(m,n;i,j) = -k_{MN}^2 \phi_{MN}(m,n;i,j)\,, \tag{17}$$

where M and N are the CNN dimensions, m and n are the summation indexes for the current space variables i and j ($i = 0,1,\ldots,M-1$; $j = 0,1,\ldots,N-1$), and k_{mn}^2 are the corresponding spatial eigenvalues. In particular, for the zero-flux boundary condition, the spatial eigenfunction and eigenvalue can be assumed to be in following form:

$$\nabla^2 \phi_{MN}(m,n;i,j) = \cos\frac{(2i+1)m\pi}{2M} \cos\frac{(2j+1)n\pi}{2N} \tag{18}$$

and

$$k_{mn}^2 = 4\left(\sin^2\frac{m\pi}{2M} + \sin^2\frac{n\pi}{2N}\right). \tag{19}$$

Then, the expected solution of Eq. (16) can be expressed as a weighted sum of $M \times N$ orthogonal space-dependent eigenfunctions $\phi_{MN}(m, n; i, j)$ in the following form:

$$\begin{aligned} x_{1,ij}(t) &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(\alpha_{mn}e^{\lambda_{mn1}t} + \beta_{mn}e^{\lambda_{mn2}t})\phi_{MN}(m,n;i,j)\,, \\ x_{2,ij}(t) &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(\gamma_{mn}e^{\lambda_{mn1}t} + \delta_{mn}e^{\lambda_{mn2}t})\phi_{MN}(m,n;i,j)\,, \end{aligned} \tag{20}$$

where α_{mn}, β_{mn}, γ_{mn}, δ_{mn} are constants depending on the initial conditions. λ_{mn1}, λ_{mn2} are the roots of the following characteristic equation (21), which are influenced by spatial eigenvalue k_{mn}^2 corresponding to spatial eigenfunction.

$$\det\left|\lambda_{mn}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}a_1 - 1 & c_1\\ c_2 & a_2 - 1\end{bmatrix} - k_{mn}^2\begin{bmatrix}0 & D_1\\ D_2 & 0\end{bmatrix}\right| = 0 \tag{21}$$

and

$$\lambda_{mn}[k_{mn}^2] = \frac{1}{2}[(a_1 + a_2) \pm \sqrt{(a_1 - a_2)^2 + 4(k_{mn}^2 D_2 - c_2)(k_{mn}^2 D_1 - c_1)}]\,. \tag{22}$$

From the above Eqs. (20) and (22), it can be derived that, if we suitably select the template parameters to make all of temporal eigenvalues have negative real parts, the CNN is completely stable. Unfortunately, this situation is not suitable for most applications of image processing, because any cell states in the CNN will decay to zero which corresponds to a homogeneous output pattern. On the other hand, if one of the temporal eigenvalues has a positive real part, the two-layer CNN becomes unstable in this linear region, and the cell states do arise and enter into the nonlinear region (i.e., saturation region or partial saturation region). After that, if the isolated cells have real stable equilibrium points, then the two-layer CNN will be stable for image processing. This mechanism is similar to the one used in Turing Patterns in arrays of coupled circuits.[13,14]

Let us consider the two-layer CNN equation described by Eq. (13) without any coupling among the cells. Thus, each cell in the CNN behaves as an isolated cell. The dynamic two first-order differential equations of the isolated cell are described in the following form:

$$\begin{aligned} \dot{x}_{1,ij} &= -x_{1,ij} + a_1 y_{1,ij} + c_1 y_{2,ij}\,, \\ \dot{x}_{2,ij} &= -x_{2,ij} + a_2 y_{2,ij} + c_2 y_{1,ij}\,, \end{aligned} \tag{23}$$

with piecewise-linear nonlinear output function. Similar state equations have been dealt with for different purposes in several literature, Refs. 13, 26 and 27, where some restrictions on parameters were assumed. Here, we simply discuss the above equation of isolated cell without any restriction on those parameters. Due to the uniform piecewise-linear nonlinearity of output function, it is reasonable to divide the two state spaces into the different types of sub-space shown in Fig. 13, where either both states of the cell are unsaturated (linear region 1), or they are fully

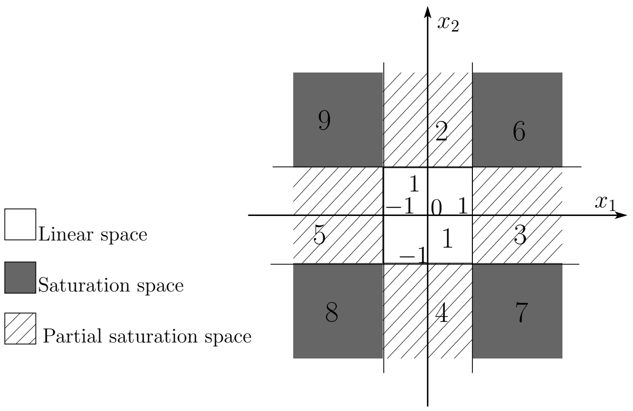

Fig. 13. Different state sub-spaces of an isolated cell in phase plane.

Table 3. The equilibrium point and its stability conditions.

Region	Equilibrium point $(\overline{x}_1, \overline{x}_2)$	Stability Conditions
1	$0, 0$	$a_1 + a_2 < 0, (a_1 + a_2)^2 > (a_1 - a_2)^2 + 4c_1c_2$
2	$\frac{-c_1}{a_1}, a_2 + 1 - \frac{c_1c_2}{a_1}$	$a_1 < 0, a_1a_2 < c_1c_2, \left\lvert\frac{c_1}{a_1}\right\rvert < 1$
3	$a_1 + 1 - \frac{c_1c_2}{a_2}, \frac{-c_2}{a_2}$	$a_2 < 0, a_1a_2 < c_1c_2, \left\lvert\frac{c_2}{a_2}\right\rvert < 1$
4	$\frac{c_1}{a_1}, -a_2 - 1 + \frac{c_1c_2}{a_1}$	$a_1 < 0, a_1a_2 < c_1c_2, \left\lvert\frac{c_1}{a_1}\right\rvert < 1$
5	$-a_1 - 1 + \frac{c_1c_2}{a_2}, \frac{c_2}{a_2}$	$a_2 < 0, a_1a_2 < c_1c_2, \left\lvert\frac{c_2}{a_2}\right\rvert < 1$
6	$a_1 + c_1 + 1, a_2 + c_2 + 1$	$a_1 + c_1 > 0, a_2 + c_2 > 0$
7	$a_1 - c_1 + 1, -a_2 + c_2 - 1$	$a_1 - c_1 > 0, a_2 + c_2 > 0$
8	$-a_1 - c_1 - 1, -a_2 - c_2 - 1$	$a_1 + c_1 > 0, a_2 + c_2 > 0$
9	$-a_1 + c_1 - 1, a_2 - c_2 + 1$	$a_1 - c_1 > 0, a_2 + c_2 > 0$

saturated (saturated regions 6, 7, 8 and 9), or one of them is unsaturated and the other one is saturated (partial saturated regions 2, 3, 4 and 5). Thus, the equilibrium in each of such sub-spaces can easily be obtained by $\dot{x}_{1,ij} = 0$, $\dot{x}_{2,ij} = 0$ and the corresponding stability conditions by the eigenvalues of the Jacobian matrix at the equilibrium point. The obtained real equilibrium points and their stability conditions are listed in Table 3. As we can see the isolated cell has only one stable equilibrium point in every sub-space if the parameters (a_1, a_2, c_1 and c_2) are suitably selected. Thus, we obtain the second necessary condition. It is worth mentioning that, in the above analysis, even if we introduce the terms of inputs, templates (B_1, B_2) and bias current (I_1, I_2), because all of them in general are constants, they do not modify the Jacobian eigenvalues and do not alter the number and stability of the equilibrium points, but shift the positions of equilibrium points in the phase plane. The obtained stability conditions are supported by the above simulations shown in Secs. 3 and 4. Besides the above mutually coupled two-layer CNNs described by Eq. (12), we can also analyze the two-layer CNNs with the same technique for considering symmetric templates A_1 and A_2, only the process is more complicated.

6. Conclusions

In this paper, we have presented mutually coupled two-layer cellular neural networks, and discussed the applications and stability. Although the two examples of center point detection and skeletonization can be solved by single layer CNN, the applications of the proposed new method in this paper to these problems is very efficient compared to those in other CNNs. Moreover, we have also found a special example of the mutually coupled two-layer CNN — dividing object into halves which cannot be solved by the single layer CNNs. Through our simulation examples, the two-layer CNNs show a real potential of the structure. We have also analyzed the stability for the two-layer CNNs with mutually coupled symmetric templates, and obtained the necessary conditions using decoupling technique.

References

1. L. O. Chua and L. Yang, "Cellular neural networks: Theory and applications", *IEEE Trans. Circuits Syst.* **35**, 10 (1988) 1257–1290.
2. T. Roska and L. O. Chua, "The CNN universal machine", *IEEE Trans. Circuits Syst.* **40**, 3 (1993) 163–173.
3. L. O. Chua and T. Roska, "The CNN paradigm", *IEEE Trans. Circuits Syst.* **40**, 3 (1993) 147–156.
4. F. Werblin, T. Roska, and L. O. Chua, "The analogic cellular neural network as bionic eye", *Int. J. Circuit Theor. Appl.* **23**, 6 (1995) 541–569.
5. H. Harrer and J. A. Nossek, "Skeletonization: A new application for discrete-time cellular neural networks using time-variant templates", *Proc. IEEE Int. Symp. Circuits & Syst.*, 1992, pp. 2897–2900.
6. D. Yu, C. Ho, X. Yu, and S. Mori, "On the application of cellular automata to image thinning with cellular neural network", *Proc. IEEE Int. Workshop Cellular Neural Networks and Their Applicat.*, CNNA-92, 1992, pp. 210–215.
7. T. Matsumoto, T. Yokohama, H. Suzuki, R. Furukawa, A. Oshimoto, T. Shimmi, Y. Matsushida, T. Seo, and L. O. Chua, "Several image processing examples by CNN", *Proc. IEEE Int. Workshop Cellular Neural Networks and Their Applicat.*, CNNA-90, 1990, pp. 100–112.
8. P. L. Ventianer, F. Werblin, T. Roska, and L. O. Chua, "Analogic CNN algorithms for some image compression and restoration tasks", *IEEE Trans. Circuits Syst.* **42**, 5 (1995) 278–284.
9. T. Roska, L. Kek, L. Nemes, A. Zarandy, and P. Szolgay, "CNN software library (templates and algorithms)", Analogical and Neural Computing Laboratory, Computer and Automation Institute of the Hungarian Academy of Sciences, Version 7.3, 1999.

10. Z. H. Yang, Y. Nishio, and A. Ushida, "Templates and algorithms for two-layer cellular neural networks", *Proc. 2002 Int. Joint Conf. on Neural Networks*, 2002, pp. 1946–1951.
11. L. O. Chua, M. Hasler, G. S. Mochytz, and J. Neirynck, "Autonomous cellular neural networks: A unified paradigm for pattern formation and active wave propagation", *IEEE Trans. Circuits Syst.* **42**, 10 (1995) 559–577.
12. Z. H. Yang, Y. Nishio, and A. Ushida, "A model of nonlinear phenomena: A unified framework two layer CNN", *Proc. 2001 Int. Technical Conf. on Circuits/Systems, Computer and Communications*, 2001, pp. 1288–1291.
13. L. Goras, L. O. Chua, D. M. W. Leenaerts, and L. Pivka, "Turing patterns in CNNs part I–III", *IEEE Trans. Circuits Syst.* **42**, 10 (1995) 602–637.
14. G. Manganaro, P. Arena, and L. Fortuna, *Cellular Neural Networks*, Springer-Verlag, Berlin, Heidelberg, 1999.
15. M. J. Ogorzalek, Z. Galias, A. M. Dabrowski, and W. R. Dabrowki, "Chaotic waves and spatio-temporal patterns in large arrays of doubly-coupled Chua's circuits", *IEEE Trans. Circuits Syst.* **42**, 10 (1995) 706–714.
16. L. Pivka, "Autowave and spatio-temporal chaos in CNNs part I and II: A tutorial", *IEEE Trans. Circuits Syst.* **42**, 10 (1995) 638–664.
17. P. Thiran, K. R. Crounse, L. O. Chua, and M. Hasler, "Pattern formation properties of autonomous cellular neural networks", *IEEE Trans. Circuits Syst.* **42**, 10 (1995) 757–774.
18. S. Majorana and L. O. Chua, "A unified framework for multi-layer high order CNN", *Int. J. Circuit Theor. Appl.* **26**, 6 (1998) 567–592.
19. C. W. Wu, L. O. Chua, and T. Roska, "A two-layer radon transform cellular neural network", *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing* **39**, 7 (1992) 488–489.
20. C. Botoca, "An associative memory using a two layer cellular neural network, part I", *Proc. the International Conference SCS 95*, 1995, pp. 121–124.
21. B. Siemiatkowska, "A highly parallel method for mapping and navigation of an autonomous mobile robot", *Proc. Int. Conf. on Robotics and Automation*, San Diego, California, May 1994, pp. 2796–2801.
22. C. W. Wu and L. O. Chua, "More rigorous proof of complete stability of cellular neural networks", *IEEE Trans. Circuits Syst.* **44**, 4 (1997) 370–371.
23. L. O. Chua and T. Roska, "Stability of a class of nonreciprocal cellular neural networks", *IEEE Trans. Circuits Syst.* **37**, 12 (1990) 1520–1527.
24. L. O. Chua and C. W. Wu, "On the universe of stable cellular neural networks", *Int. J. Circuit Theor. Appl.* **20**, 4 (1992) 497–517.
25. N. Takahashi and L. O. Chua, "On the complete stability of nonsymmetric cellular neural networks", *IEEE Trans. Circuits Syst.* **45**, 7 (1998) 754–758.
26. T. Hu and Z. Lin, "A complete stability analysis of planar linear systems under saturation", *IEEE Trans. Circuits Syst.* **47**, 4 (2000) 498–512.
27. F. Zou and J. A. Nossek, "Bifurcation and chaos in cellular neural networks", *IEEE Trans. Circuits Syst.* **40**, 3 (1993) 166–173.