INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS, VOL. 20, 453–467 (1992)

# DISCRETE-TIME CELLULAR NEURAL NETWORKS

HUBERT HARRER AND JOSEF A. NOSSEK
*Institute for Network Theory and Circuit Design, Technical University Munich, Arcisstr. 21, D-8000 Munich 2, Germany*

SUMMARY

A network structure called a discrete-time cellular neural network is introduced. It is derived from cellular neural networks and feedback threshold networks. The architecture is discussed and its advantages are presented. Convergence is proved for a large class of templates and applications are given for the following image-processing tasks: linear thresholding, connected component detection, hole filling, concentric contouring, increasing and decreasing objects step by step, searching for objects with minimal distance, and oscillation.

## 1. INTRODUCTION

Cellular neural networks (CNNs) as introduced in Reference 1 have recently received growing attention. They are an efficient architecture for performing image processing and pattern recognition. The simple structure of a single cell and the local connectivity make CNNs well suited for analogue VLSI implementation. The network is described by a system of non-linear differential equations[2]

$$\frac{dx^c}{dt} = -x^c(t) + \sum_{d \in N_r(c)} a_d^c y^d(t) + \sum_{d \in N_r(c)} b_d^c u^d + i^c \tag{1}$$

$$y^c(t) = \tfrac{1}{2}(|x^c(t)+1| - |x^c(t)-1|) \tag{2}$$

The variable $x^c$ denotes the state of the cell $c$, $y^c$ its output and $u^c$ its input. The state of a single cell is controlled by the inputs and outputs of adjacent cells within an $r$-neighbourhood $N_r(c)$. The outputs are fed back multiplied by the coefficients $a_d^c$, and the inputs are multiplied by the control parameters $b_d^c$. The value of $i^c$ is constant and is used for adjusting a threshold. A set of feedback, control coefficients and the thresholds is called a template. The template coefficients are translationally invariant. The $r$-neighbourhood $N_r(c)$ is the set of all cells within a distance $r$ including cell $c$. There are problems concerning the boundary of the grid, because not all neighbouring cells exist. To resolve these, dummy cells are inserted which possess constant values for $u^c$ and $y^c$. Below, $u^c = -1$ and $y^c = -1$ are used for dummy cells, which means that the input and initial array are surrounded by a ring of white pixels. An $r$-neighbourhood can be defined similarly on a hexagonal grid.

The output $y^c$ is obtained by a piecewise linear function of the cell state $x^c$ in (2). Depending on the initial state $\mathbf{x}(t_0)$, the system converges to a final state $\mathbf{x}(t_\infty)$ with binary outputs if the feedback coefficients are symmetric and the self-feedback is large enough ($a_c^c > 1$).

When applying the Einstein summation convention, a summation is performed automatically for identical indices in the lower and upper positions of variables in a product term. A summation is not done if the indices are parenthesized. This abbreviation will be used throughout this text. The index $d$ is always applied within the $r$-neighbourhood of a cell $c$, where the index $c$ is used for all cells on the grid. With that, (1) is written as

$$\frac{dx^c}{dt} = -x^c(t) + a_d^c y^d(t) + b_d^c u^d + i^c \tag{3}$$

Complex dynamics can be applied to image processing. If global tasks are performed by a wave-like

0098–9886/92/050453–15$12.50

*Received 9 April 1991*
*Revised 20 January 1992*

propagation of the signals, the system can solve such *global* tasks with only *local* interconnections. Many useful applications of this type have been found recently, e.g. the connected component detector,[3] the hole filler,[4] the shadow detector[5] and linear thresholding.[6] A big problem of CNNs is the design of the template coefficients. Normally they are determined only by a trial-and-error method. The learning algorithm described in Reference 7 fails if the system dynamics is too complex. Hence a system is required which solves global tasks despite local connectivity, where the coefficients can be derived by a logic formulation of the problem. This system should also possess reliable convergence properties[1,2,8,9] and a simple cell structure to make it well suited for VLSI implementation.

## 2. ARCHITECTURE

Discrete-time cellular neural networks (DTCNNs) are a special kind of feedback threshold network[10-12] where the local interconnections and the translationally invariant weights are transferred from CNNs. They are completely described by a recursive algorithm. The dynamic behaviour is based on the feedback of clocked and binary outputs. A single cell is influenced by the inputs and outputs of adjacent cells within an $r$-neighbourhood as defined for CNNs. The architecture is closely related to cellular automata[13,14] but differs from them in having continuous-valued inputs and weights.

*Definition 1. Discrete-time cellular neural networks*

Discrete-time cellular neural networks (DTCNNs) are defined by the following algorithm using the index $d$ within the $r$-neighbourhood of cell $c$:

$$x^c(k) = a_d^c y^d(k) + b_d^c u^d + i^c \tag{4}$$

$$y^c(k) = f(x^c(k-1)) = \begin{cases} 1 \text{ if } x^c(k-1) > 0 \\ -1 \text{ if } x^c(k-1) < 0 \end{cases} \tag{5}$$

In contrast to CNNs, the system is clocked and only binary values $(-1, +1)$ are weighted by the coefficients $a_d^c \in \mathbb{R}$ of the feedback operator. Before an iteration is started, the initial values $y^c(0) \in \{-1, 1\}$ have to be defined. They are of crucial importance for the dynamic behaviour of the system and can be considered as a second cell input. The value $+1$ denotes a black pixel, $-1$ a white pixel. The inputs $u^c \in D^c$ are restricted to a continuous definition set, which is necessary for the convergence proof in Section 3. The inputs are multiplied by the control parameters $b_d^c \in \mathbb{R}$ for all neighbouring cells $d$ within $N_r(c)$. A constant value $i^c \in \mathbb{R}$ is added to adjust a threshold of cell $c$. A non-indexed value $i \in \mathbb{R}$ means that there are equal values for all cells. The continuous template coefficients and inputs are assumed to be time-invariant, but an extension to time-dependent parameters[15] is also possible. The binary output state $y^c$ is determined by the sign of $x^c$ in the previous iteration step. It is undefined for $x^c = 0$. In practice there is always some noise, which implies that $|x^c| > 0$.

If the template coefficients are chosen such that

$$\Delta = \min_{c,k} |a_d^c y^d(k) + b_d^c u^d + i^c| \tag{6}$$

is large enough, then the algorithm is relatively insensitive to parameter tolerances of $a_d^c$, $b_d^c$ or $i^c$ smaller than $\Delta$, because individually they cannot cause any changes in the output states.

Abbreviating the notation, the time-independent part of (4) is defined as the cell bias $k^c(\mathbf{u})$, which again is a function of the constant inputs and the threshold values:

$$k^c = k^c(\mathbf{u}) = b_d^c u^d + i^c \tag{7}$$

The advantage of DTCNNs is the description of the next output state through a set of linear inequalities. The template coefficients are computed by solving this system of linear inequalities, where in each time step the subsequent output state is postulated. This can be done iteratively using the design

algorithm described in Reference 16. The simulation on a digital computer is very simple, since the system is time-discrete and a sophisticated integration algorithm is not necessary.

For the algorithm (4), (5) the network structure of a single cell is given in Figure 1. The variables are substituted by voltages and the iteration steps by discrete-time instances.

The next iteration step of the system is obtained after one period of the clock signals $\varphi_1$ and $\varphi_2$ (Figure 2). Neglecting parasitic transients, the time instance $kT$ can be chosen arbitrarily during the 'high' phase of $\varphi_1$.

The multiplications are done by linear voltage-controlled current sources. The total current sum is transformed into a voltage $v_x^c(kT)$ by the resistor $R_x$ with

$$v_x^c(kT) = R_x(A_d^c v_y^d(kT) + B_d^c v_u^d + I^c) \tag{8}$$

The voltage-controlled voltage sources $F(v_x^c(kT))$ and $F(v_z^c(kT))$ have the non-linear characteristic

$$F(v) = \begin{cases} V_{sat} & \text{if } v > 0 \\ -V_{sat} & \text{if } v < 0 \end{cases} \tag{9}$$

Here the next output state is determined and the capacitor $C_1$ is charged during the 'high' phase of $\varphi_1$ to $V_{sat}$ or $-V_{sat}$. During the 'high' phase of $\varphi_2$ ($\varphi_1$ 'low') the capacitors $C_1$ and $C_2$ are connected in parallel. The voltage $v_z^c(kT)$ has settled after the transient to

$$v_z^c(kT) = \frac{C_1 F(v_x^c((k-1)T)) + C_2 v_z^c((k-1)T)}{C_1 + C_2} \tag{10}$$

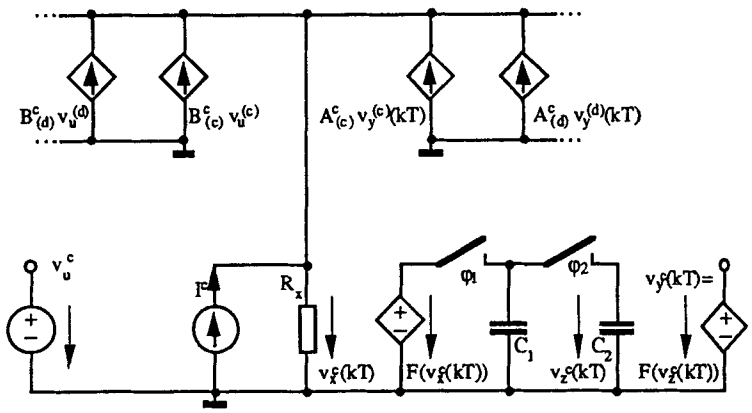



Figure 1. Network structure of a single cell

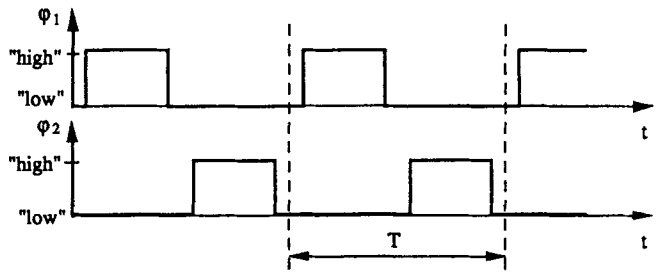



Figure 2. Clock diagram of discrete-time cellular neural networks

Table I. Normalization of the variables and network elements

| | |
|---|---|
| $u^c = v_u^c / V_{sat}$ | $a_d^c = A_d^c R_x$ |
| $x^c(k) = v_x^c(kT) / V_{sat}$ | $b_d^c = B_d^c R$ |
| $z^c(k) = v_z^c(kT) / V_{sat}$ | $i^c = I^c R_x / V_{sat}$ |
| $y^c(k) = v_y^c(kT) / V_{sat}$ | |

Because $| F(v_x^c) | = V_{sat}$ and $| v_z^c | \leqslant V_{sat}$ $\forall k$, it follows for $C_1 > C_2$ that

$$v_y^c(kT) = F(v_x^c((k-1)T)) = \begin{cases} V_{sat} & \text{if } v_x^c((k-1)T) > 0 \\ -V_{sat} & \text{if } v_x^c((k-1)T) < 0 \end{cases} \tag{11}$$

Inserting (8) into (11) leads to

$$v_y^c(kT) = F(R_x(A_d^c v_y^d((k-1)T) + B_d^c v_u^d + I^c)) \tag{12}$$

Applying the normalization in Table I, the algorithm (4), (5) is obtained.

Compared to CNNs, an analogue realization of DTCNNs has some important advantages.

(i) Because of the binary outputs, the interconnection of several chips is very simple. For CNNs, continuous output signals have to be propagated during the transient.
(ii) The parameter insensitivity of (6) makes the network robust against fabrication tolerances if the templates are designed appropriately.
(iii) The propagation speed can be controlled within a large range only by changing clock rate. This also simplifies the testability of a chip.

Efficient analogue circuit structures are given in References 17 and 18 for this architecture.

*Remarks* A DTCNN containing a large enough neighbourhood can be considered as a single-layer perceptron with a comparator non-linearity and feedback connections or a discrete Hopfield net which is updated in parallel. However, large neighbourhoods are not well suited for simple VLSI implementation. For the latter, strong restrictions are given for the weight coefficients (local connectivity and translational invariance). If translational invariance is abolished, the network structure can be used for an associative memory trained by the Hebb rule.[19]

## 3. CONVERGENCE

In the following sub-sections theoretical aspects concerning the convergence of DTCNNs are given. Until now, general conditions to obtain a convergent output pattern have not been found. However, for a large class of templates, convergent behaviour can be shown. It is also proved that in contrast to CNNs, oscillation is possible for symmetric feedback coefficients and large enough self-feedback. In this case the cycle length is limited to a period of two.

### *3.1. Eigendominant templates*

Convergence can be proved for dominant self-feedback and cell bias of a DTCNN. Such templates are used for a large class of applications, where a transition from white to black or from black to white cannot be undone. Examples of such tasks are the hole filler, increasing and decreasing of objects or the search for objects with minimal distance in Section 4. First, the following definitions are necessary.

*Definition 2. Convergence*

A DTCNN is said to be convergent if

$$\exists k_0: \quad (\forall k > k_0: \mathbf{y}(k) = \mathbf{y}(k_0)) \tag{13}$$

The obtained output state $\mathbf{y}(k_0)$ is called a fixed point.

*Remark* It is obvious that such a fixed point is reached for any two subsequent iterations with identical output states for *all* cells and $\Delta > 0$. The latter condition makes sure that no cell state can be zero, which is always assumed throughout this paper.

*Definition 3. Minimum absolute cell bias*

A minimum absolute cell bias $k^c_{\min}$ is defined as the smallest absolute value of the cell bias $k^c(\mathbf{u})$ for all possible input patterns:

$$k^c_{\min} = \min_{\mathbf{u} \in D} | k^c(\mathbf{u}) | \tag{14}$$

*Remark* If the inputs $u^c$ are not restricted to a definition set $D^c$, it is usually possible to choose a combination for $\mathbf{u}$ that implies $\forall_c$: $k^c = 0$ for any $b^c_d \neq 0$.

*Definition 4. Eigendominance*

A DTCNN is said to be eigendominant for a template if

$$\forall_c: \quad (a^{\{c\}}_{\{c\}} + k^c_{\min}) > \sum_{\substack{d \in N_r(c) \\ d \neq c}} | a^c_d | \tag{15}$$

*Remarks* Eigendominant templates with $k^c_{\min} = 0$ imply the diagonal dominance

$$a^{\{c\}}_{\{c\}} > \sum_{\substack{d \in N_r(c) \\ d \neq c}} | a^c_d | \tag{16}$$

If $\forall_c$: $k^c = 0$, the next output state is determined by the self-feedback coefficient and no cell can change its output state. The initial pattern is also the convergent output pattern and therefore such networks make no sense for image processing.

*Theorem 1. Convergence for eigendominant templates*

An eigendominant DTCNN converges in $n$ or fewer iterations, where $n$ denotes the number of cells on the grid.

*Proof* To prove the theorem, let us first consider any cell with $k^c > 0$ ($k^c > k^c_{\min}$). For the output state we have two cases.

Case 1: $y^c(k) = +1$

The sign of $x^c(k)$ is positive according to Definition 4, because

$$a^{\{c\}}_{\{c\}} y^c(k) + k^c > - \sum_{\substack{d \in N_r(c) \\ d \neq c}} a^c_d y^d(k) \tag{15}$$

for any $y^d \in \{-1, 1\}$. Hence the output state of cell $c$ cannot be influenced by adjacent cells and the output state is $+1$ for *all* subsequent iterations.

Case 2: $y^c(k) = -1$

$$\text{(a)}\quad x^c(k) < 0 \Rightarrow y^c(k+1) = -1$$

The white output state in the subsequent iteration step is caused by the neighbouring cells.

$$\text{(b)}\quad x^c(k) > 0 \Rightarrow y^c(k+1) = +1$$

The output state changes from white to black. For the following iterations Case 1 applies and the transition cannot be undone.

Similar considerations apply sign-symmetrically to a cell with $k^c < 0$ ($k^c \leqslant -k^c_{\min}$). In this case a transition is only allowed once from black to white and cannot be undone afterwards. Because each cell can change its output state only *once* during the transient, an eigendominant DTCNN is forced to converge in $n$ or fewer iterations. It is not necessary to satisfy Case 1 ($k^c > 0$, $y^c(k) = +1$) or the sign-symmetric case ($k^c < 0$, $y^c(k) = -1$) for a fixed point, because convergence is also obtained if an output state is determined by Case 2(a) and *no* cell changes its output value in the subsequent iteration. □

Examples of eigendominant templates are given in Section 4. It is possible to embed the condition of eigendominance in the learning algorithm described in Reference 16 to make sure that only convergent templates are utilized. However, the class of tasks for which a solution exists at all is then crucially restricted. The proof can also be applied to general feedback threshold networks as perceptrons with feedback connections or discrete Hopfield nets.

### *3.2. Symmetric feedback templates*

It is proved in Reference 10 that feedback threshold networks with symmetric weights are either convergent or oscillate with a cycle length of two. This can also be concluded for DTCNNs containing symmetric feedback coefficients.

*Theorem 2. Symmetric feedback templates*

A DTCNN with symmetric feedback coefficients

$$a^c_d = a^d_c \tag{17}$$

is either convergent or oscillates with a period of two.

*Proof* The idea of the proof is based on the Lyapunov theory for discrete systems. The function

$$E(k) = -y_c(k)a^c_d y^d(k-1) - (y_c(k) + y_c(k-1))k^c \tag{18}$$

with

$$y_c = y^c \tag{19}$$

is used as an energy function and can only decrease during the transient. It is restricted to $|E(k)| \leqslant E_{\max}$, where

$$E_{\max} = n\,|a^c_d|\,1^d + 2n(u_{\max}\,|b^c_d|\,1^d + i_{\max}) \tag{20}$$

$$u_{\max} = \max_c |u^c|, \qquad i_{\max} = \max_c |i^c| \tag{21}$$

Then we obtain for two subsequent iterations

$$\begin{aligned}\Delta E &= E(k+1) - E(k)\\ &= -y_c(k+1)a^c_d y^d(k) - (y_c(k+1) + y_c(k))k^c + y_c(k)a^c_d y^d(k-1) + (y_c(k) + y_c(k-1))k^c\end{aligned} \tag{22}$$

Using the symmetry condition (17), it is easy to see that

$$y_c(k)a_d^c y^d(k-1) = y_c(k-1)a_d^c y^d(k)$$

and thus it follows immediately that

$$\Delta E = -(y_c(k+1) - y_c(k-1))(a_d^c y^d(k) + k^c) = -(y_c(k+1) - y_c(k-1))x^c(k)$$

Because the sign of $x^c(k)$ determines the value of the following output state, we obtain with (5)

$$x^c(k) = |x^{(c)}(k)|\, y_{(c)}(k+1)$$
$$\Delta E = -|x^c(k)|\,(y_c(k+1)^2 - y_c(k-1)y_c(k+1)) \leqslant 0$$

Hence *E decreases* if at least one cell changes its output state from iteration $k-1$ to $k+1$. Thus $\Delta E = 0$ is only possible if

$$\forall c: \quad y^c(k-1) = y^c(k+1) \tag{23}$$

Then two cases have to be considered:

$$\text{(a)} \quad \mathbf{y}(k) = \mathbf{y}(k-1)$$

This is a fixed point in accordance with Theorem 2.

$$\text{(b)} \quad \mathbf{y}(k) \neq \mathbf{y}(k-1)$$

Here we have with (23) an oscillating system of period two. Larger cycle lengths cannot appear, because the energy function is restricted by (20) and cannot steadily decrease. □

The number of iterations before a fixed point or a limit cycle is obtained is estimated in Reference 10.

## 4. APPLICATIONS

Most applications of CNNs[3-6,20] could be performed by DTCNNs too. A novel task of image processing is the transformation of objects to their concentric contours and the search for objects with minimal distance to a fixed point. For CNNs no template has been found for these tasks up to now. As an example for an oscillating system, a template with symmetric feedback coefficients is introduced. The applications are defined on a square grid but can easily be mapped onto a hexagonal grid.

The following definitions are used throughout this section.

*Definition 5. Four-connected*

Two pixels $(i, j)$ and $(m, n)$ of the same colour are called four-connected if

$$|i-m| + |j-n| \leqslant 1 \tag{24}$$

*Definition 6. Eight-connected*

Two pixels $(i, j)$ and $(m, n)$ of the same colour are called eight-connected if

$$|i-m| \leqslant 1 \wedge |j-n| \leqslant 1 \tag{25}$$

*Definition 7. Four-connected path*

A four-connected path $P$ is defined as a set of $r$ pixels where two following pixels $p_i$ and $p_{i+1}$ for $i = 1, \ldots, r-1$ are four-connected.

*Definition 8. Four-connected object*

A four-connected object $O$ is defined as the maximum set of identical coloured pixels where for any two pixels $(i, j) \in O$ and $(m, n) \in O$ a four-connected path $P \subseteq O$ does exist.

*Remarks* The 1-neighbourhood was defined for eight-connected pixels by Definition 6. A four-connected neighbourhood consists of cells which are only influenced by four-connected pixels. Four-connected pixels are always eight-connected. For one-dimensional problems or on a hexagonal grid, this distinction is superfluous. The definition of an eight-connected object is similar to Definition 8.

### *4.1. Linear thresholding*

*Problem description* For linear thresholding, only local features of the image to be processed are of importance. It can be considered as a convolution. The feedback coefficients are set to zero and the image is applied to the cell inputs. The control coefficients are used for extracting particular features of the input image within the $r$ neighbourhood of each cell. This corresponds to the cell bias $k^c$ performing a discrete convolution with a kernel **b**:

$$x^c = k^c = b^c_d u^d + i^c \tag{26}$$

The value of $i^c$ adjusts a threshold for accepting the decision of the detected feature.[2,6,20]

*Convergence proof* A DTCNN used as a linear threshold network is convergent after a single iteration for any parameters $b^c_d$, $u^c$ and $i^c$, because the network has *no feedback*. The subsequent output state depends only on the inputs.

As an example for local image processing, the edge detection of black eight-connected objects is demonstrated. This task can be solved by the template

$$a = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}, \qquad b = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4{\cdot}5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}, \qquad i = -1{\cdot}5 \tag{27}$$

The condition for obtaining a black output state of cell $i$, $j$ is

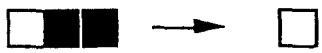
$$4{\cdot}5u^{i,j} - u^{i-1,j} - u^{i,j-1} - u^{i,j+1} - u^{i+1,j} - 1{\cdot}5 > 0 \tag{28}$$

i.e. a cell only stays black if there is at least one white neighbouring cell by Definition 5. This is true for all pixels in the edge set. It is proved in the Appendix that the edges of noisy input patterns are recognized correctly if they are restricted to a definition set $D = \{u^c \mid 0{\cdot}89 < |u^c| < 1{\cdot}11\}$.

### *4.2. Connected component detector (CCD)*

*Problem description* For pattern recognition and data compression, black objects within a single line should be compressed to the size of one pixel and moved to the right boundary, where black and white pixels are alternating depending on the number of objects. The black pixels in the final output pattern correspond to the number of connected components in the initial pattern.[4]

For this, the DTCNN has to be programmed as a cellular automaton. The output state of the middle cell is *only* allowed to change if one of the following situations occurs:



(29)

(30)

Table II. Computation of $x^c(k)$ for all combinations $y^{i,j-1}$, $y^{i,j}$ and $y^{i,j+1}$

| $y^{i,j-1}(k)$ | $y^{i,j}(k)$ | $y^{i,j+1}(k)$ | $x^{i,j}(k)$ | $y^{i,j}(k+1)$ |
|---|---|---|---|---|
| −1 | −1 | −1 | −1 | −1 |
| −1 | −1 | +1 | −3 | −1 |
| −1 | +1 | −1 | +1 | +1 |
| −1 | +1 | +1 | −1 | −1 |
| +1 | −1 | −1 | +1 | +1 |
| +1 | −1 | +1 | −1 | −1 |
| +1 | +1 | −1 | +3 | +1 |
| +1 | +1 | +1 | +1 | +1 |

Initializing the state $\mathbf{y}(0)$ with the pattern to be analysed, the task can be solved by the template

$$a = \boxed{+1 \mid +1 \mid -1}, \qquad b = \boxed{0 \mid 0 \mid 0}, \qquad i = 0 \tag{31}$$

This is proved by computing the sign of $x^c(k)$ for all possible combinations of $y^{i,j-1}$, $y^{i,j} = y^c$ and $y^{i,j+1}$ (Table II).

The CCD is an example of global image processing with only local cell connectivity.

*Convergence proof* A DTCNN with the template coefficients of (31) is convergent for all initial conditions, because a changing output state (from black to white or from white to black) can *only* be propagated to the right side. After $n$ or fewer iterations a fixed point is achieved because of the bounded size of the field. Considering the change of cell $b$ from black to white in Figure 3, cell $a$ has to remain white. The transition (30) is only allowed if the right neighbour of cell $a$ is white. Cell $c$ must stay black, because (29) is *only* possible if the left neighbour is white. Therefore white pixels can only move to the right. Similar considerations can be made for a transition from white to black.

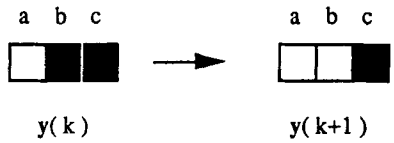



Figure 3. Transition from black to white

*Remark* A rotation of the parameters in the feedback operator by a multiple of 45° allows connected component detection in different directions.

### 4.3. *Hole filler*

*Problem description* White pixels completely enclosed by a black eight-connected object $O$ should be black in the final output state.[5]

This task can be solved by a propagation starting at the boundaries of the grid. The initial state is black (+1). The pattern is used as input of the cells. Transitions from black to white in the output are only allowed if the cell input is white and there is a four-connected white neighbour cell by Definition 5. This is satisfied for

$$y^{i,j}(k+1) = -1 \Rightarrow u^{i,j} = -1 \wedge y^{i-1,j}(k) + y^{i,j-1}(k) + y^{i,j+1}(k) + y^{i+1,j}(k) < +4 \tag{32}$$

$$y^{i,j}(k+1) = +1 \Rightarrow u^{i,j} = +1 \vee y^{i-1,j}(k) + y^{i,j-1}(k) + y^{i,j+1}(k) + y^{i+1,j}(k) = +4 \tag{33}$$

and can be done using the template

$$a = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & 2 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}, \qquad b = \begin{array}{|c|c|c|}\hline 0 & 4{\cdot}5 & 0 \\ \hline \end{array}, \qquad i = -1 \tag{34}$$

*Convergence proof* A DTCNN using the template (34) is convergent by Theorem 1 if the inputs are restricted to a definition set

$$D = \{u^c \mid |u^c| > \tfrac{2}{3}\} \tag{35}$$

The template is eigendominant by Definition 4 because $k^c_{\min} > 2$ and the condition

$$\min_c(2 + k^c_{\min}) > 4$$

is satisfied for all combinations of the input pattern.

### 4.4. *Concentric contours*

*Problem description* A concentric contour of a black eight-connected object $O$ is defined as a set of alternating black and white rings starting from the boundary to the interior. Notice that complex object structures may have more than one interior point. Figure 4 shows a black square and its concentric contour.

This can be used for a special kind of size-independent object recognition. Objects of different size but with identical interior contour rings could be detected by linear thresholding.

A DTCNN solves this task by extracting the outer contour ring of the object and inverting its interior pixels. The contour ring remains stable during the following iterations. A new concentric ring is built up every iteration step. The arising rings can be interpreted as a propagation to the interior of the objects. A cell output $y^{i,j}$ is only allowed to change its state if there are four neighbours $y^{i-1,j}$, $y^{i,j-1}$, $y^{i,j+1}$ and $y^{i+1,j}$ by Definition 5 which have the same colour. Applying the pattern to the initial state $\mathbf{y}(0)$ *and* the cell inputs $\mathbf{u}$, this task is performed by the template

$$a = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 3{\cdot}5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}, \qquad b = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 4 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}, \qquad i = -4 \tag{36}$$

Choosing $b^{i,j}_{i,j} = 4$ and $i = -4$ assures that $x^{i,j}$ can never be positive if $u^{i,j} = -1$. On top of this the following conditions are required:

$$u^{i,j} = +1 \wedge y^{i,j}(k) = y^{i-1,j}(k) = y^{i,j-1}(k) = y^{i,j+1}(k) = y^{i+1,j}(k) = +1 \Rightarrow x^{i,j} < 0 \tag{37}$$

$$u^{i,j} = +1 \wedge y^{i,j}(k) = y^{i-1,j}(k) = y^{i,j-1}(k) = y^{i,j+1}(k) = y^{i+1,j}(k) = -1 \Rightarrow x^{i,j} > 0 \tag{38}$$

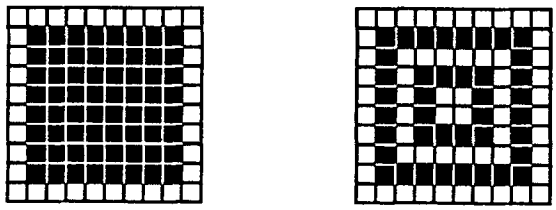


Figure 4. Square and its concentric contour

*Convergence proof* A DTCNN with the template (36) is convergent for all input patterns. In every iteration step the number of cells possessing four equal neighbours decreases, because the extracted contour ring always becomes stable. In fewer than $n/2$ iterations there is no cell with four equal neighbours.

*Remark* For 'decomposing' an object of any size in its concentric rings, only a template of neighbourhood $r = 1$ is necessary. For this application a template for CNNs has not been found up to now.

### 4.5. *Increasing and decreasing objects step by step*

*Problem description* A black object $O$ should be increased or decreased every iteration step by a one-pixel layer.

The DTCNN has to be programmed as a cellular automaton. The pattern is used as initial state. Increasing the object, *only* a transition from white to black of the output state is allowed if the cell has at least one black neighbour. Doing this, the object is increased every iteration by a new ring of black pixels.

The problem is solved by the template

$$a = \begin{array}{|c|c|c|}\hline 0{\cdot}5 & 0{\cdot}5 & 0{\cdot}5 \\ \hline 0{\cdot}5 & 0{\cdot}5 & 0{\cdot}5 \\ \hline 0{\cdot}5 & 0{\cdot}5 & 0{\cdot}5 \\ \hline \end{array}, \qquad b = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}, \qquad i = +4 \tag{39}$$

because of

$$\exists d \in N_r(c): \quad y^d(k) = +1 \Rightarrow y^c(k+1) = +1 \tag{40}$$

*Convergence proof* A DTCNN using the template (39) is convergent by Theorem 1. The template is eigendominant by Definition 4 because $k^c_{\min} = 4$ and the condition

$$\min_c (a_{\{c\}}^{\{c\}} + k^c_{\min}) = 4{\cdot}5 > 4$$

is satisfied for all combinations of the initial pattern.

Decreasing a black object $O$, *only* transitions from black to white are allowed if the cell has at least one white neighbour. Therefore the object is decreased every iteration by a single ring of black pixels.

This is done by changing the threshold $i$ of (39) to $-4$.

Increasing and decreasing is another example of global image processing using wave propagation. Here the propagation is done orthogonal to the edges of the object. Decreasing an object, the propagation is directed towards the interior point(s); for an increase the opposite direction is used.

*Remark* The template (39) increases an object corresponding to an eight-connectivity. It can also be done with respect to a four-connectivity using the template

$$a = \begin{array}{|c|c|c|}\hline 0 & 0{\cdot}5 & 0 \\ \hline 0{\cdot}5 & 0{\cdot}5 & 0{\cdot}5 \\ \hline 0 & 0{\cdot}5 & 0 \\ \hline \end{array}, \qquad b = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}, \qquad i = +2 \tag{41}$$

*4.6 Search for objects with minimal distance*

*Problem description* The increase of an object in Section 4.5 should be modified to control the propagation by the cell inputs. A transition from white to black is only allowed if the cell has at least one black neighbour and the input of the cell is white. Therefore black pixels in the input pattern stop the propagation. The input pattern can be considered as a maze where a wave spreads through the lanes beginning from a starting point. This can be applied for the search of objects with minimal distance to a given pixel if the propagating wave is compared to the position of distributed targets on the grid, which are only allowed on white pixels in the input pattern. For this, an additional logic layer is used to perform an AND function between the propagating wave and the time-dependent positions of the targets. If an object is detected, the propagation has to be stopped. The detected object has the minimal distance to the starting point.

Figure 5 shows the time-dependent development of the output pattern for the first four iterations. Black pixels correspond to black pixels in the input pattern and grey pixels denote black pixels in the output pattern.

This can be solved by the template

$$a = \begin{array}{|c|c|c|}\hline 0{\cdot}5 & 0{\cdot}5 & 0{\cdot}5 \\ \hline 0{\cdot}5 & 0{\cdot}5 & 0{\cdot}5 \\ \hline 0{\cdot}5 & 0{\cdot}5 & 0{\cdot}5 \\ \hline \end{array}, \qquad b = \begin{array}{|c|c|c|}\hline 0 & -4 & 0 \\ \hline \end{array}, \qquad i = 0 \tag{42}$$

The starting point, which is only allowed to be a white pixel in the input pattern, is the only pixel that is black-coloured in the initial state. For the next output state the following requirements are given:

$$u^c = +1 \wedge y^c(k) = -1 \Rightarrow y^c(k+1) = -1 \tag{43}$$

$$u^c = -1 \wedge \exists d \in N_r(c)\colon\ y^d(k) = +1 \Rightarrow y^c(k+1) = +1 \tag{44}$$

*Convergence proof* A DTCNN using the template (42) is convergent by Theorem 1 if the inputs are restricted to a definition set

$$D = \{u^c \mid |u^c| > \tfrac{7}{8}\} \tag{45}$$

The template is eigendominant by Definition 4 because $k^c_{\min} > 3{\cdot}5$ and the condition

$$\min_c(0{\cdot}5 + k^c_{\min}) > 4$$

is satisfied for all combinations of the input pattern.

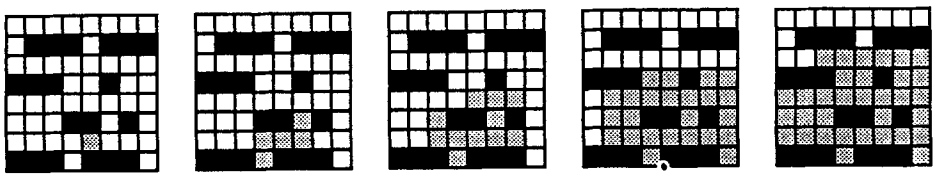



Figure 5. Time-dependent development of the output pattern (grey pixels) for the illustrated input pattern (black pixels)

*4.7. Example of a limit cycle using a symmetric template*

Convergence cannot be guaranteed for DTCNNs with symmetric feedback templates and a self-feedback coefficient larger than unity as proved for Theorem 2. Using the template

$$a = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 1{\cdot}5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}, \qquad b = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline \end{array}, \qquad i = 0 \tag{46}$$

an oscillating output pattern of period two is obtained for the initial pattern in Figure 6.

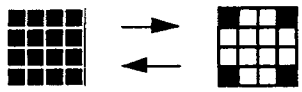


Figure 6. Oscillating output pattern

## 5. CONCLUSIONS

In contrast to CNNs, discrete-time cellular neural networks have clocked variables and a comparator characteristic for the non-linear function. The distinctions from cellular automata are the continuous-valued weight coefficients and inputs in the linear part of the cell which allow linear thresholding and processing of noisy or grey-level input signals.

Global image processing with only local cell connectivity is obtained by clocked and binary output states which are fed back within an $r$-neighbourhood. In contrast, CNNs do this by way of an analogue dynamic transient using continuous feedback.

For CNNs the propagation speed depends crucially on the template coefficients and the input or output signals, because these influence the derivative of the state variable $x^c$ in (1). Large values of the derivative can change the state variables within a short time and cause fast propagation. Hence the CNN structure is able to do image-processing tasks *where the propagation speed is controlled by both the input and initial pattern*. Such tasks cannot be done by DTCNNs because of the clocked and binary outputs.

However, for some applications a constant signal propagation is required. The search for objects with minimal distance described in Section 4.6 cannot be done by a CNN, because there propagation through wide lanes of the maze is faster than through small lanes. This is caused by larger derivatives of the state variables depending on the number of black neighbours to a cell.

The DTCNN architecture guarantees a constant propagation speed. Here *only the sign* of the variable $x^c$ is of importance and *not its absolute value*.

Another example is the concentric contouring described in Section 4.4. The DTCNN is able to decide in a *binary* fashion if a cell has only neighbours of the same colour. For the CNN the output of a cell has to change *continuously*. This change in the output values is *not equal* for all 'interior' pixels, because corner cells of a new concentric ring possess more neighbours with constant output values and therefore change their values at different speeds. This leads to 'disturbed' rings for complex object structures because of the different propagation speeds.

Further applications-oriented research has to be done, especially to decide which classes of tasks can *only* be solved by CNNs and which only by DTCNNs.

Simulation on digital computers is far less expensive, because a numerical integration algorithm is not needed.

A learning algorithm[16] based on the relaxation method performs the computation of the template coefficients if the logical problem formulation is given. This method is well suited for DTCNNs, because

the following output state of a cell is described by an inequality. Another advantage is the insensitivity of the parameters if the coefficients are chosen appropriately.

Analogue implementation of the DTCNN has the properties of being insensitive against fabrication tolerances, simple interconnection of several chips, control of the propagation speed by the clock rate and good testability.

A comparison with cellular automata would be of importance, because most applications are taken outside this area. For special problems they are certainly more efficient and simpler to implement. However, the advantage of CNNs and DTCNNs is their flexibility. The programmable structure can solve a large class of problems simply by changing the parameters.

ACKNOWLEDGEMENT

The authors are very grateful to Professor Dr. Tamás Roska for stimulating discussions and helpful comments.

## APPENDIX: DETERMINATION OF THE DEFINITION SET FOR THE EDGE DETECTOR

The choice of the definition set for the inputs has to guarantee that the value of the state variable $x^c$ cannot be zero. This is identical to the condition of $k^c_{\min} > 0$, because we have no feedback for edge detection. Hence we must restrict the absolute values of the inputs to a lower and an upper bound by

$$D = \{u^c \mid u_{\min} < |u^c| < u_{\min}\} \tag{47}$$

where $u_{\min} = 1 - \delta$ and $u_{\max} = 1 + \delta$ has to be satisfied for symmetric limits. This implies

$$u_{\min} + u_{\max} = 2 \tag{48}$$

For the 'worst case' (only *one* white neighbour, e.g. $u^{i-1,j}$) the condition

$$4 \cdot 5\,|u^{i,j}| + |u^{i-1,j}| - |u^{i,j-1}| - |u^{i,j+1}| - |u^{i+1,j}| - 1 \cdot 5 > 0 \tag{49}$$

has to be true. This is achieved by transforming (49) to an equality where positive inputs are replaced by the lower bound $u_{\min}$ and negative inputs by the upper bound $u_{\max}$.

$$4 \cdot 5u_{\min} + u_{\min} - u_{\max} - u_{\max} - u_{\max} - 1 \cdot 5 = 0, \quad \text{i.e. } 5 \cdot 5u_{\min} - 3u_{\max} - 1 \cdot 5 = 0 \tag{50}$$

Inserting (48) into (50) leads to

$$u_{\min} = 7 \cdot 5/8 \cdot 5 \approx 0 \cdot 89, \qquad u_{\max} = 9 \cdot 5/8 \cdot 5 \approx 1 \cdot 11 \tag{51}$$

## REFERENCES

1. L. O. Chua and L. Yang, 'Cellular neural networks: theory', *IEEE Trans. Circuits and Systems*, **CAS-35**, 1257–1272 (1988).
2. J. A. Nossek, G. Seiler, T. Roska and L. O. Chua, 'Cellular neural networks: theory and circuit design', *Int j. circ. theor. appl.*, **20**, 523–543, (1992).
3. T. Matsumoto, L. O. Chua and H. Suzuki, 'CNN cloning template: shadow detector', *IEEE Trans. Circuits and Systems*, **CAS-37**, 1070–1073 (1990).
4. T. Matsumoto, L. O. Chua and H. Suzuki, 'CNN cloning template: connected component detector', *IEEE Trans. Circuits and Systems*, **CAS-37**, 633–635 (1990).
5. T. Matsumoto, L. O. Chua and R. Furukawa, 'CNN cloning template: hole-filler', *IEEE Trans. Circuits and Systems*, **CAS-37**, 635–638 (1990).
6. L. O. Chua and B. E. Shi 'Exploiting cellular automata in the design of cellular neural networks for binary image processing', *ERL Memo UCB/ERL M89/130*, University of California, Berkeley, November 1989.
7. F. Zou, S. Schwarz and J. A. Nossek, 'Cellular neural network design using a learning algorithm', *Proc. First IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA-90*, Budapest, 1990, IEEE, New York, 1991, pp. 73–81.
8. L. O. Chua and T. Roska 'Stability of a class of nonreciprocal cellular neural networks', *IEEE Trans. Circuits and Systems*, **CAS-37**, 1520–1527 (1990).

9. T. Roska, 'On the qualitative and quantitative relationships between the analog and digital realizations of "neural" computing circuits', *Report 2/89*, Hungarian Academy of Sciences, Budapest, 1989.
10. E. Goles, F. Fogelman and D. Pellegrin, 'Decreasing energy functions as a tool studying threshold networks', *Discrete Appl. Math.*, **12**, 261–277 (1985).
11. E. Goles, 'Fixed point behaviour of threshold functions on a finite set', *SIAM J. Alg. Appl. Math.*, **3**, 529–531 (1982).
12. J. Bruck, 'On the convergence properties of the Hopfield model', *Proc. IEEE*, **78**, 1579–1585 (1990).
13. K. Preston Jr. and M. J. B. Duff, *Modern Cellular Automata: Theory and Applications*, Plenum, New York, 1984.
14. T. Toffoli and N. Margolus, *Cellular Automata Machines*, MIT Press, Cambridge, MA, 1987.
15. T. Roska and L. O. Chua, 'Cellular neural networks with nonlinear and delay-type template elements', *Proc. First IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA-90*, Budapest, 1990, IEEE, New York, 1991, pp. 12–25.
16. H. Harrer, J. A. Nossek and F. Zou, 'A learning algorithm for time-discrete cellular neural networks', *Proc. IJCNN-91*, Singapore, 1991, pp. 717–722.
17. H. Harrer, J. A. Nossek, G. Seiler and R. Stelzl, 'An analog CMOS compatible convolution circuit for analog neural networks', *Proc. Micro Neuro-91*, Munich, 1991, pp. 231–241.
18. H. Harrer, J. A. Nossek and R. Stelzl, 'An analog implementation of discrete-time cellular neural networks', *Report TUM-LNS-TR-91-14*, Technical University of Munich, June 1991.
19. S. Tan, J. Hao and J. Vandewalle, 'Cellular neural networks as a model of associative memories', *Proc. First IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA-90*, Budapest, 1990, IEEE, New York, 1991, pp. 26–35.
20. L. O. Chua and L. Yang, 'Cellular neural networks: applications', *IEEE Trans. Circuits and Systems*, **CAS-35**, 1273–1290 (1988).
21. N. Frühauf and E. Lüder, 'Realization of CNNs by optical parallel processing with spatial light valves', *Proc. First IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA-90*, Budapest, 1990, IEEE, New York, 1991, pp. 281–290.