New Model and Applications of Cellular Neural Networks in Image Processing

Radu Matei
Technical University "Gh.Asachi" of Iasi
Romania

1. Introduction

The cellular neural network (CNN) in its standard form as defined by Chua and Yang has become a paradigm of analog, high-speed computation, with various applications in parallel signal processing, especially in image processing tasks. Within two decades this domain has extensively developed both theoretically and from the applications point of view. Although the mainstream research basically followed the standard CNN model, many other CNN models and architectures were proposed, envisaging new and more complex classes of applications.
We propose here an alternative CNN model, inspired from Hopfield-type recurrent networks. This new model has the same general architecture as the standard model, while the cell structure, preserving the basic elements, is modified from a dynamical point of view. The cell non-linearity, having a major role in CNN dynamics, no longer applies only to the current state variable as it does in the standard model, but to the whole information fed to the current cell, thereby leading to a different dynamic behavior. The new cell structure and state equations are presented and the dynamics and equilibrium points are analyzed.
The advantages of this model, which we call "CNN with overall non-linearity", are a simpler design of the templates, and the fact that all voltages in the cells are limited to a fixed dynamic range regardless of template parameters; thus the boundedness problem regards the current range instead of voltage, which makes it less critical from an implementation point of view. The image processing capabilities of the proposed CNN model will be approached. Some new image processing tasks are proposed. We first analyze the behavior of the proposed CNN in processing binary images, then using so-called local rules we design the templates for several specific tasks, which are: convex corner detection, erasing thin lines, detection of closed curves, finding all paths between two selected points through a labyrinth, rotation detector, marked object reconstruction and finding the intersection points of one-pixel thin lines from two superposed binary images. Many other processing tasks both on binary or grayscale images are possible. Further research may focus on extending the range of applications of the proposed model.
Next some applications of CNNs in image linear filtering will be approached. In this range of applications, the cells of the CNN operate in the linear domain, and the system

stability must be ensured. The design of CNN linear filters with high selectivity may lead in general to large-size templates. Due to the fact that local connectivity in CNNs is an essential restriction, the decomposition of a large-size template into small-size templates is often necessary. Fast linear filtering using parallel computing systems like CNNs may be useful in early vision aplications and pre-processing tasks. We will present some original and efficient design methods for several types of two-dimensional filters: circular, maximally-flat, elliptically-shaped, orientation-selective filters etc. We also propose a class of 2D linear filters which are designed starting from a desired shape in the frequency plane, described in polar coordinates. Using this method, some interesting filters can be designed, such as concave-shaped filters presenting narrow lobes in the frequency plane. A particular class are the zero-phase 2D filters, which are extensively used in image processing due to the absence of phase distortions. For all the proposed filters, the design starts from an imposed 1D prototype and uses rational approximations and various frequency transformations to obtain the 2D frequency response. The resulted filters are efficient, at the same time of low complexity and relatively high selectivity. The design methods are entirely analytical, but they could be combined with numerical optimization techniques. The filters can be implemented either digitally or on analog processing systems like CNNs. Further research also envisages an efficient and reliable implementation of this class of filters.

2. Cellular Neural Networks with Overall Nonlinearity

2.1 Equations and Circuit Model

The state equation of the current cell *i* for a standard CNN has the form (Chua & Yang, 1988):

$$\frac{dx_i}{dt} = -x_i + \sum_{j \in N_i} A_j \cdot f(x_j) + \sum_{k \in N_i} B_k \cdot u_k + I_i \tag{1}$$

and the output is given by the piece-wise linear function:

$$y_i = f(x_i) = 0.5\left(|x_i + 1| - |x_i - 1|\right) \tag{2}$$

Here a new version of CNN is proposed (Matei, 2001), with a modified structure as compared to the standard one. The difference lies in the form of the state equation, which has the form:

$$\frac{dx_i}{dt} = -x_i + f\left(\sum_{j \in N_i} A_j \cdot x_j + \sum_{k \in N_i} B_k \cdot u_k + I_i\right) \tag{3}$$

For simplicity we used a single index for the cells, as in the case of 1D CNNs. Unlike the standard CNN, in this case the nonlinear function applies to the entire information gathered from the neighboring cells, i.e. state variables (x_j), inputs (u_k) and the bias

term (I_i). For this reason, we have called this a CNN with "overall non-linearity". This CNN version was inspired by the well-known dynamical model of a neuron which is the elementary unit of so-called recurrent neural networks or Hopfield networks. Within this framework, the standard CNN can be regarded as a particular Hopfield network with local connectivity. The neuron's output is related to the current state ("activation potential") through a nonlinear activation function with various shapes (logistic, piece-wise linear,etc.). There are two basic dynamical neuron models, related by a linear and generally invertible transformation (Haykin, 1994). The proposed CNN corresponds to the second recurrent model. Therefore, the proposed CNN model and the Chua-Yang model are equivalent, being described by similar equations if a linear transformation is applied.
Let us denote by z_i the argument of the piece-wise linear function f in (3):

$$z_i = \sum_{j \in N_i} A_j \cdot x_j + \sum_{k \in N_i} B_k \cdot u_k + I_i = \sum_{j \in N_i} A_j \cdot x_j + k_i \tag{4}$$

where the inputs and bias are constant and were put together into the term k_i.
Multiplying both members of (3) by A_j, summing over $i \in N_i$ (N_i denotes the current cell neighborhood) and shifting again the indices *i* and *j*, we finally obtain (Matei, 2001):

$$\frac{dz_i}{dt} = -z_i + \sum_{j \in N_i} A_j \cdot f(z_j) + \sum_{k \in N_i} B_k \cdot u_k + I_i \tag{5}$$

having a form similar to (1), but in the new variable z_i. Equation (4) defines the linear transformation which relates the two CNN models.

2.2 System and Circuit Description

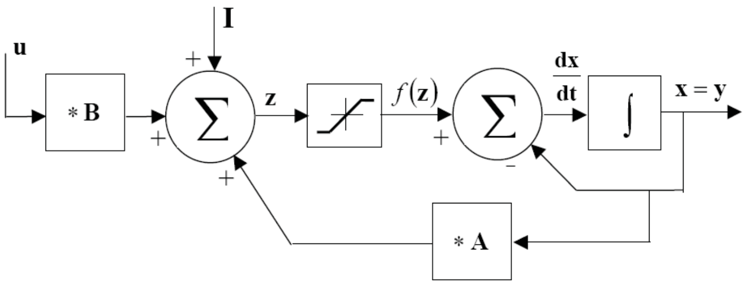
The modified CNN will be analytically described from dynamical point of view, as well as at circuit level. Figure 1 shows a block diagram representation of the dynamic behavior of the modified CNN, which can be compared to the standard CNN block diagram (Matei, 2001).
In this block diagram all the variables are vectors or matrices. The correlation sums in the state equation can be written as convolutions. The variable **z** at the input of the nonlinear block is:

$$\mathbf{z} = \mathbf{A} * \mathbf{x} + \mathbf{B} * \mathbf{u} + \mathbf{I} \tag{6}$$

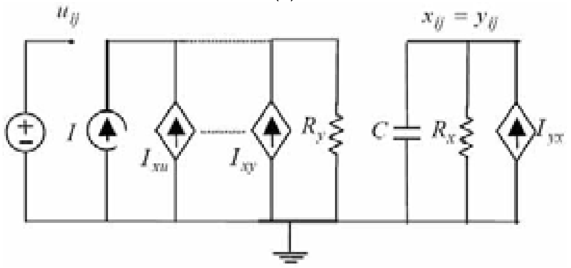
and the matrix state equation of the modified CNN will be:

$$d\mathbf{x}/dt = -\mathbf{x} + f\left(\mathbf{A} * \mathbf{x} + \mathbf{B} * \mathbf{u} + \mathbf{I}\right) \tag{7}$$

Unlike the standard CNN case (where the nonlinear block is placed in the feedback loop), in



(a)



(b)

Fig. 1.(a) Block diagram of proposed CNN model; (b) Circuit structure of a cell

the modified CNN it operates in the forward path. Here we must emphasize an important feature of the proposed CNN: the output is identical to the state variable $\mathbf{y}(t) = \mathbf{x}(t)$ throughout the operating domain – a significant advantage in the CNN implementation.
In Figure 1(b) we present the cell circuit of the modified CNN, which preserves the same elements as the standard CNN, but has a different structure. The major difference between the two structures lies in the location of the nonlinear element. In the standard CNN case, the nonlinear function applies to the state variable x_i, which is a voltage, and gives the output voltage $y_i = f(x_i)$. In the modified CNN equation (3), the non-linearity applies to the amount denoted z_{ij}, summing up the entire information from the states and the inputs of the neighboring cells. The variable z_{ij} is a current (given by several linear controlled current sources), and $f(z_i)$ is a current as well. For the piecewise-linear current-controlled current source we can write $I_{yx} = f(z_i)$. The resistor denoted R_y has the single role of summing the currents given by the linear sources, while in the standard CNN cell it converts voltage into current. Unlike the Chua-Yang model, in this case the nonlinear current source feeds the current directly into the capacitor C and the resistor R_x. On the cell circuit, considering normalized values for C and R_x we can also derive the state equation (3).

2.3 Dynamics and Equilibrium Points

Using the Lyapunov method and the linear mapping (4) it can be proven that the modified CNN also has the property of convergence to a constant steady state following any transient. The stability properties are also established in Hopfield network theory (Haykin, 1994).

For standard CNNs, the state equation (1) is linear as long as all the state variables are less than one in absolute value, $x_i \in [-1,1]$. This domain defines the linear region of the state space. For the modified CNN, the linear region of the state space in variable x_i depends on the feedback state information, the input information and the bias term, as shows (3). In the linear region of the state space, as long as no cell reached saturation, equations (1) and (3) have a similar form, since in this domain we have $f(x) = x$, which implies that the linear dynamics of the standard CNN and modified CNN are identical. The difference lies in the way in which the two types of CNNs enter the nonlinear region and reach equilibrium.
In the following paragraph we proceed to an analysis of the dynamic behavior and a study of the equilibrium points of the modified CNN. We will use the so-called driving-point plot, which gives the state derivative dx_i/dt vs. state variable x_i (Chua & Yang, 1988). We will emphasize the dynamic routes and the possible equilibrium points the system can reach.
Using equation (5) which corresponds to a standard CNN in the state variable z_i, we can first represent the driving-point plot ($dz_i/dt, z_i$) in the new variable z_i as in Figure 2 (a). The drawing of this plot is shown in detail in (Matei, 2001). In order to derive the driving point plot in the original variable x_i, we use the linear mapping (4) in the form:

$$z_i(t) = p \cdot x_i(t) + g_i(t) \tag{8}$$

$$g_i(t) = \sum_{j \in N_i, j \neq i} A_j \cdot x_j + \sum_{k \in N_i} B_k \cdot u_k + I_i = \sum_{j \in N_i, j \neq i} A_j \cdot x_j + k_i \tag{9}$$

In (8), $p = A_0$ is the central element of the feedback template. Supposing for the moment a fixed, constant value for $g_i(t)$, denoted g_i, we obtain for $p \neq 0$:

$$x_i(t) = (1/p) \cdot (z_i(t) - g_i) \tag{10}$$

$$dz_i/dt = p \cdot dx_i/dt \tag{11}$$

The case $p = 0$ should be analyzed separately. Through a simple graphical construction shown in Figure 2 (a) we can draw the corresponding driving point plot in the original variable x_i. This can be done using two auxiliary plots: (a) the plot (x_i, z_i), given by (10), represented by a straight line with slope of value $1/p$ and offset value g_i; (b) the plot $(dz_i/dt, dx_i/dt)$, given by (11), consisting in a straight line with slope p crossing the origin .
The plots depicted in Figure 2 (b) were drawn for the particular values $p = 2$, $g_i = 0.5$.
The segment A_1B_1 with slope ($p - 1$), given by:

$$dz_i/dt = (p-1) \cdot z_i + g_i \tag{12}$$

has an identical form in x_i:

$$dx_i/dt = (p-1) \cdot x_i + g_i \tag{13}$$

The points A_1, B_1, given by the coordinates: $A_1(1, p-1+g_i)$, $B_1(-1, -p+1+g_i)$ become:

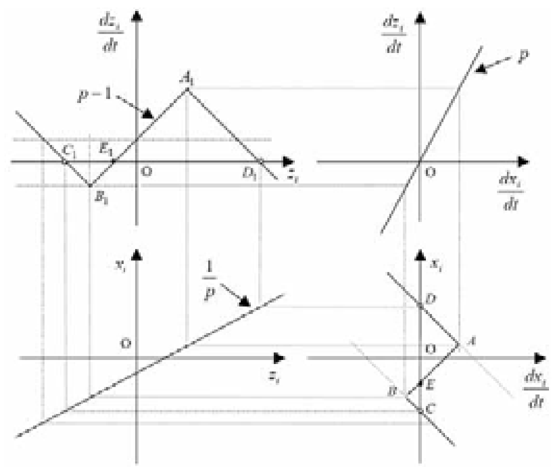
$$A\left(\frac{1-g_i}{p},\frac{p-1+g_i}{p}\right),\ B\left(\frac{-1-g_i}{p},\frac{-p+1+g_i}{p}\right) \tag{14}$$

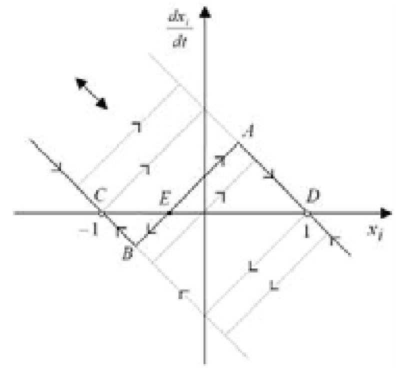
The lines A_1D_1 and B_1C_1, described by the equations:

$$dz_i/dt=-z_i+p+g_i \qquad dz_i/dt=-z_i-p+g_i \tag{15}$$

become AD and BC respectively, given by:

$$dx_i/dt=-x_i+1 \qquad dx_i/dt=-x_i-1 \tag{16}$$





(a) (b)

Fig. 2. (a) Graphical construction for determining the driving point plot; (b) Driving point plot, dynamic routes and equilibrium points for different values of g_i

In the diagram $(dx_i/dt, x_i)$ the points C and D result from the intersection of the straight lines BC and AD with axis x_i, imposing the condition: $dx_i/dt=0$.
We finally obtain: $x_i(C)=-1$ and $x_i(D)=1$. Points $C(-1,0)$ and $D(1,0)$ will therefore be fixed equilibrium points of the modified CNN, independent of the values of *p* and g_i. For different values of the amount g_i, the diagram $(dz_i/dt, z_i)$ shifts up or down. Correspondingly, the $(dx_i/dt, x_i)$ plot slides along the two parallel dashed lines crossing the two fixed points C and D. In Figure 2(b) one typical driving point plot is shown with continuous line and the dynamic routes are indicated by arrows. The number of equilibrium points and their nature for different values of *p* and g_i are analyzed in (Matei, 2001).

2.4 Dynamic Range of Modified CNNs

For standard CNNs the following result has been proven in (Chua & Yang, 1988):

Theorem: All states x_i in a CNN are bounded for $t > 0$ and the bound v_{max} is given by:

$$v_{max} = 1 + |I| + \max_i \sum_i \left(|A_i| + |B_i|\right) \tag{17}$$

if we consider the normalized resistance value $R_x = 1$.

The possibility of estimating the maximum dynamic range is very important from the implementation point of view, because it allows one to optimally design the circuit parameters in order to keep the voltage values within reasonable bounds.

In (Rodriguez-Vasquez et al., 1993) a new CNN model, called "full-range" model was introduced, adding a nonlinear term to the state equation, which confines the state value to the range [-1,1]. This simplifies the design, allowing for reduced area and power consumption in VLSI implementation.

Our approach gives in fact another solution to this problem, forcing the state and output values to be equal. Therefore, the proposed CNN with "overall non-linearity" is in fact a "full-range" model as well. In the modified CNN, this problem is solved by the form of the state equation itself. Indeed, at any equilibrium point, the current cell state variable is $x_i = f(z_i)$ and consequently its variation is bounded to the range $[-1,1]$. Moreover, the output variable is identical to the state variable at any time: $y_i(t) = x_i(t)$.

Referring again to the circuit structure, the argument $z_i(t)$ of the nonlinear function $f(z_i)$ given by (4) is physically a current. However, since equation (5) is similar to the state equation of a standard CNN cell, but written in variable z_i, equation (17) may be formally applied to evaluate a maximum bound I_{max} for the cell currents.

Therefore in the case of proposed CNN with overall nonlinearity, the boundedness problem regards the current range instead of voltage, which makes it less critical from an implementation point of view. All voltage variations in the modified CNN are limited to a fixed range corresponding to the extreme pixel values, regardless of template parameters.

3. Applications of CNNs with Overall Nonlinearity

In this section we approach the image processing capabilities of the proposed CNN model. Some processing tasks will be proposed on binary (black and white) images. We first analyze the behavior of the proposed CNN model in processing binary images, then we design the templates for a few specific tasks, namely: convex corner detection, erasing thin lines, detection of closed curves, finding all paths between two selected points through a labyrinth, rotation detector, marked object reconstruction and finding the intersection points of one-pixel thin lines from two superposed binary images. Many other processing tasks on binary or grayscale images are possible.

A vast reference for different classes of standard CNN tasks is the CNN Software Library, a template catalog for a large range of analog processing operations on binary and grayscale images. These also include more complex analog algorithms which consist of sequences of elementary tasks which are fulfilled under digital control.

We use here the convention that the pixel value -1 corresponds to white, and the pixel value +1 corresponds to black. Generally, one binary image is loaded into the CNN as initial state,

while another binary image may be applied to the array input. In this range of applications the system must behave as a bipolar CNN. A standard CNN is said to be bipolar if $\mathbf{u},\mathbf{y} \in \mathbb{B}^{MN}$, $\mathbf{x}(0) \in \mathbb{B}^{MN}$, where $\mathbb{B} = \{-1,1\}$, and *M*, *N* are the number of rows and columns of the CNN array (Hänggi & Moschytz, 2000).
Based on the information contained in the two images, the CNN must accomplish a given task upon the state image, according to a set of local rules for which the feedback and control templates are designed. If a given binary image is loaded as initial state, this implies that all cells are initially saturated. Depending on the chosen template parameters, the initial state of a given cell may be stable or unstable. If the state is stable, the corresponding pixel will remain unchanged (black or white); if on the contrary the state is unstable, the pixel may switch to the opposite state, following a transient evolution. Taking into account that we want to obtain a binary output image in response to the processed image, we impose that the cells which are initially unstable switch into the opposite state, therefore for a generic cell $C(i,j)$ we will have: $\dot{x}_{ij}(0) \neq 0$. These cells will switch into the opposite saturated state which is supposed to be stable according to local rules. The switching of a cell between two saturated states implies a dynamic evolution through the middle linear region. As stated from the beginning in (3), the dynamics of the current cell is described by:

$$dx_i(t)/dt = -x_i(t) + f\left(z_i(t)\right) \tag{18}$$

Suppose that a given pixel is initially black, i.e. $x_i(0) = 1$. If it is to remain black, this implies $\dot{x}_i(0) = 0$ and from (18) we must have: $f\left(z_i(0)\right) = 1$, therefore $z_i(0) > 1$. If the black pixel must switch to white, we impose: $z_i(0) < -1$, so $f(z_i(0)) = -1$; the initial state derivative is:

$$\dot{x}_i(0) = -x_i(0) - 1 = -2 < 0 \tag{19}$$

The cell state value decreases exponentially from 1 to -1:

$$x_i(\infty) = -1 \qquad \dot{x}_i(\infty) = -x_i(\infty) - 1 = 0 \tag{20}$$

therefore the cell finally reaches a stable equilibrium point.
In the following paragraphs we present several tasks on binary images in order to show the image processing capabilities of the proposed CNN model with overall nonlinearity. For each of these tasks, the templates are designed and simulation results are provided.

3.1 Convex Corner Detection

Although the convex corner detection on binary images is implemented as well on standard CNNs, we show here the template design method for the proposed model. We assume a feedback template *A* of the general symmetric form:

$$A = \begin{bmatrix} s & s & s \\ s & p & s \\ s & s & s \end{bmatrix} \tag{21}$$

The first step is to find the local rules according to the desired result. We have to take into account all the possible combinations of black and white pixels within a 3×3 frame. Although the total number of combinations is quite large, due to template *A* symmetry the number of distinct combinations is largely reduced. All inner black pixels must turn white, while any edge pixel must remain black if it locates a convex corner or turn white otherwise. In Figure 3(a), the combinations (a)–(d) represent convex corners, while cases (e)-(h) do not. Assuming for *s* a negative value, we obtain a system of inequalities which reduces to the set:

$$p + 2s + I < -1 \qquad p + I > 1 \qquad p + 8s - I > 1 \tag{22}$$

Finding a solution to the above system is a linear programming problem which can be solved graphically or numerically. We finally reach the set of parameters: $p = 7.5$; $s = -1.5$; $I = -6$. A simulation of this task is shown in Figure 3(b), (c). The result of convex corner detection is shown, resulted from the test image (b) loaded as initial state. We remark that all convex corners are correctly detected, while the concave corners do not appear in the output image (c). A thin line emerging from a convex corner is also detected.

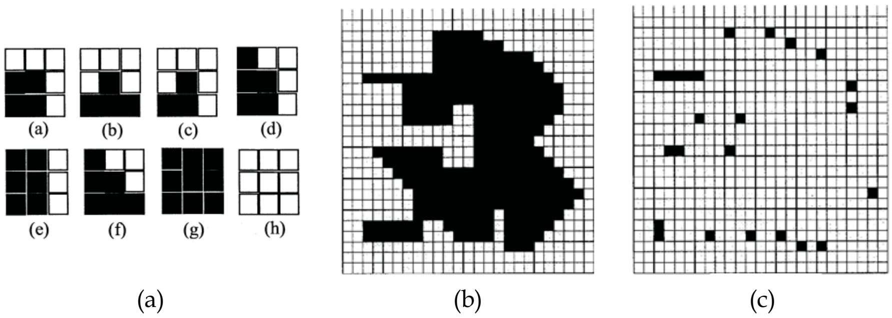


Fig. 3. (a) Possible pixel combinations in convex corner detection; (b) Image loaded as initial state; (c) Output images containing detected corners

3.2 Erasing Thin Lines

We intend to design a template which erases every thin line (one-pixel thick) from a binary image, leaving all other objects unchanged. For example, the test image in Figure 4(a) is loaded into the CNN as initial condition. We look for the feedback template *A* in the form (21), this time assuming $s > 0$; writing the local rules, we reach the following system:

$$p - 4s + I < -1 \qquad p - 2s + I > 1 \qquad p - I > 1 \tag{23}$$

and we choose a convenient set of parameter values: $p = 8$; $s = 2$; $I = -2$. The CNN input is not used. The final result of the processing is shown in Figure 4(c). We notice that the two thin curves from the left, as well as the spiral and the lines emerging from the black square have been erased. The lines perpendicular to the square edges may leave a single residual pixel. In Figure 4(b) a snapshot of an intermediate state is shown, in which the progressive deletion of thin lines is noticeable. The compact objects present in the image remain unchanged.

3.3 Detection of Closed Curves

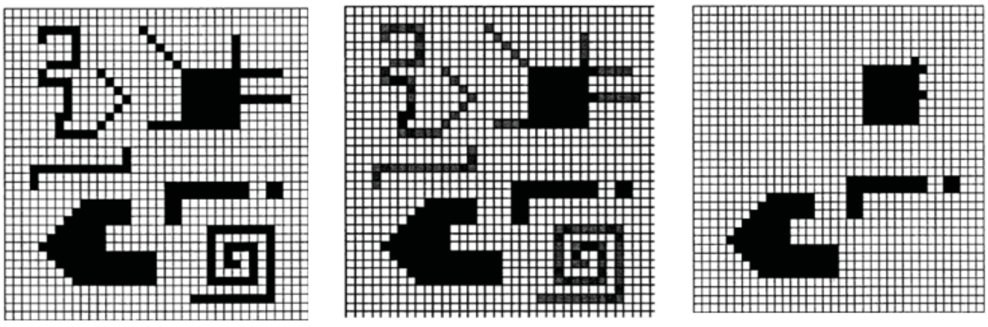
In some applications it might be useful to discern between closed and open curves. These curves in turn may result from a previous processing, such as edge detection etc.
We design a CNN task which detects the one-pixel thin closed curves and deletes the open curves from a binary image. The absence of at least one pixel from a closed curve turns it into an open curve. Both the input and the initial state are used. The same binary image **P** is loaded as initial state and applied to input: $U(t) = P$; $X(0) = P$. The templates are chosen as:

$$A = \begin{bmatrix} s & s & s \\ s & p & s \\ s & s & s \end{bmatrix}; \ B = \begin{bmatrix} r & r & r \\ r & p & r \\ r & r & r \end{bmatrix}; \ r = -0.5s \qquad (24)$$

Applying the corresponding local rules, the set of inequalities is derived:

$$2p - 2s + I > 1 \qquad 2p - 3s - I > 1 \qquad 2p - 2.5s + I < -1 \qquad (25)$$

A convenient choice of parameters is: $p = 9$; $s = 6$; $I = -4.5$. In this task we observed the 8-pixel connectivity, i.e. the current black pixel is considered connected to any other black pixel lying in its 3×3 neighborhood. A simulation is shown in Figure 5. The test image (a) contains six thin curves, which feature all types of possible edges and corners. There is also a compact concave object in the image. The image (b) shows the CNN state at an intermediate moment, and the final image (c) contains just the closed curves and the compact object, which was preserved. The process of erasing open curves relies on a dynamic propagation starting from the free ends. The black pixels from the ends turn white according to the imposed local rules until the entire curve is erased.


(a) (b) (c)
Fig. 4. (a) Binary image; (b) Transient state; (c) Steady state output image

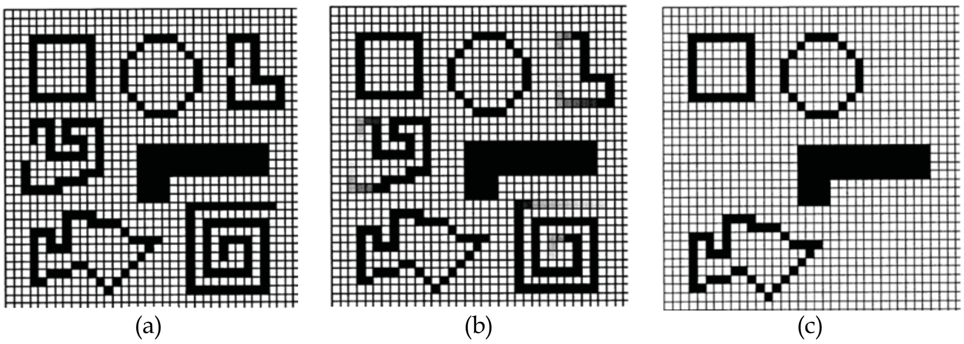


Fig. 5. Closed curve detection: (a) binary image; (b) transient state; (c) output image

3.4 Finding All Paths Between Two Selected Points Through a Labyrinth

A processing task related to the previous one is presented below. We consider a labyrinth made up of white one-pixel thin curves against a black background. In this case we suppose valid the convention that one pixel is possibly connected only to four neighbors (up, down, left, right). The target of the proposed task is to find all the existing paths between two given points situated in the labyrinth, which are marked by white pixels in the input image. A point is represented by one pixel, either on the input or state image.

The binary image **P** representing the labyrinth is loaded into the CNN as initial state ($X(0) = \mathbf{P}$). The input image may be zero everywhere (average grey level) except for two white pixels (of value -1) which mark the points selected in the labyrinth.

The dynamic process which finds the routes between the chosen points is similar to the one used in application 3.3. The template parameters are designed such that all labyrinth routes with unmarked ends are gradually erased starting from end pixels which turn from black to white (cell outputs switch from -1 to +1). The templates are searched in the form:

$$A = \begin{bmatrix} r & s & r \\ s & p & s \\ r & s & r \end{bmatrix}; \; B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{26}$$

and the following system results:

$$-p + 4r + I < -1 \quad -p + 2s - 4r + I > 1 \quad -p + 2s + 2r + I > 1 \quad -p + 2s + 4r + I - a < -1 \tag{27}$$

with convenient parameter values: $p = 12$; $s = 4$; $r = 0.5$; $a = 8$; $I = 8$.

In Figure 6 a simulation example is shown. The input image (b) contains two white pixels on a uniform gray background (zero pixel value). The two pixels locate on the labyrinth image (a) (loaded as initial state) the points between which the routes are selected. At the end of dynamic propagation shown in (c), the output image (d) results.

More generally, routes can be found among several points in the labyrinth, or a single point is chosen and we find all the closed routes containing that point. The classical problem of finding the shortest path through a labyrinth is more complex and it can be solved through a

CNN algorithm using several nonlinear templates, which are run sequentially under digital control.

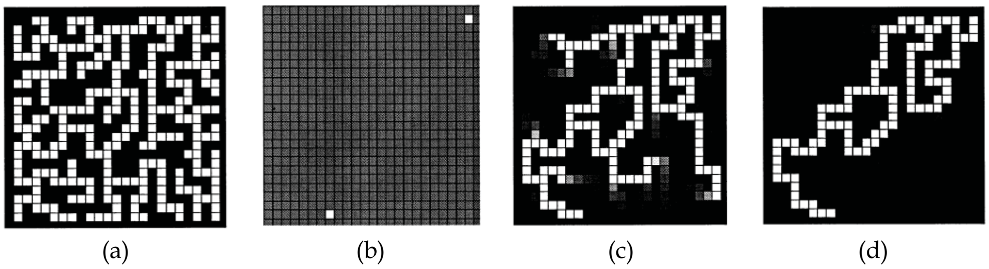


Fig. 6. (a) Labyrinth image as initial state; (b) input image; (c) transient state; (d) output image containing detected routes between selected points

3.5 Erasing Objects with Tilted Edges (Rotation Detector)

We will design a CNN task which detects in a binary image the compact convex or concave objects having only horizontal and vertical edges and removes all tilted objects or objects having at least one tilted edge. In particular the CNN can detect the rotation of such objects. This is based on the discrete representation on an array of a straight line, in particular the edge of a compact object, with a certain slope. Due to the limited spatial resolution of the CNN array, any tilted edge will be mapped as a stair-like edge. An edge with the minimum angle will map into a staircase with only two stairs, while an edge with a slope of $\pi/4$ is mapped into a staircase with a maximum number of one-pixel stairs. The binary image **P** is loaded into the CNN as initial state ($X(0) = \mathbf{P}$) and applied at the CNN input as well, $U(t) = \mathbf{P}$. We look for the templates in the general forms:

$$A = \begin{bmatrix} r & s & r \\ s & p & s \\ r & s & r \end{bmatrix}; \ B = \begin{bmatrix} 0.5r & -0.5s & 0.5r \\ -0.5s & p & -0.5s \\ 0.5r & -0.5s & 0.5r \end{bmatrix} \tag{28}$$

and finally the following convenient values are found: $p = 5$; $s = 5$; $r = -0.8$; $I = -11.2$.

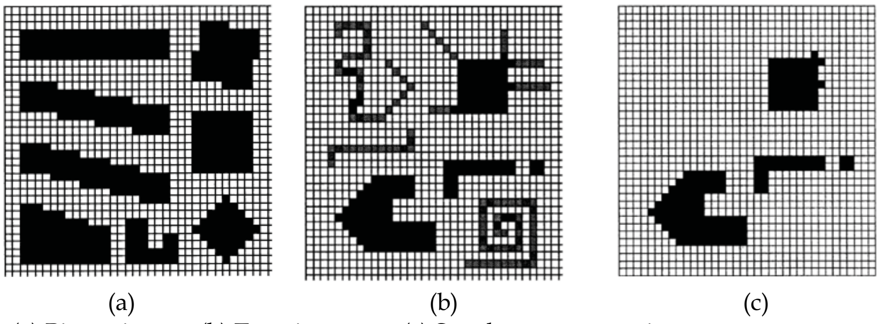


Fig. 7. (a) Binary image; (b) Transient state; (c) Steady state output image

A simulation is presented in Figure 7. Image (a) contains several compact objects: a rectangle and a square placed horizontally and their rotated versions etc. The objects with at least one tilted edge are gradually eroded through a dynamic propagation, as shows the transient snapshot (b); the final image (c) preserves only objects with horizontal and vertical edges. Therefore, this task can be used to detect the rotation of simple objects (rectangles etc.).

A simulation is presented in Figure 7. Image (a) contains several compact objects: a rectangle and a square placed horizontally and their rotated versions etc. The objects with at least one tilted edge are gradually eroded through a dynamic propagation, as shows the transient snapshot (b); the final image (c) preserves only objects with horizontal and vertical edges. Therefore, this task can be used to detect the rotation of simple objects (rectangles etc.).

3.6 Marked Object Reconstruction

A binary image **P** containing compact shapes or curves is applied to the CNN input. From this image we intend to select certain objects which we want to preserve, and delete the others. The selection of objects is done by marking them with a black pixel, in the image loaded as initial state. Therefore at the input we have $U = \mathbf{P}$ and the initial state $X(0)$ consists in black marking pixels against a white background. These can be placed anywhere within the area of selected object (in the interior or on the edge). The marked objects are reconstructed in the state image through a gradual process. Starting from the marking pixel, the white pixels within the marked object turn black. This propagation process stops on the edge of the object in the input image. The unmarked objects will not appear in the final image. We assume the feedback template of the form (21) and the control template zero, except the central element equal to *a*. With $s > 0$, $p > 0$, we finally obtain the system:

$$p - 8s + I + a > 1 \qquad p + 8s - I - a > 1 \qquad p + 6s - I - a < -1 \tag{29}$$

A convenient set of parameters results as: $p = 3$; $s = 1.5$; $a = 12$; $I = 1.5$. An example of selecting marked objects is shown in Figure 8. The image (a) contains six compact objects and curves. Using the marking points from (b) the objects from (d) were selected, while using points in (e), we selected the objects as in image (f). The image (c) is an intermediate snapshot showing the reconstruction of the selected objects around the marking points.

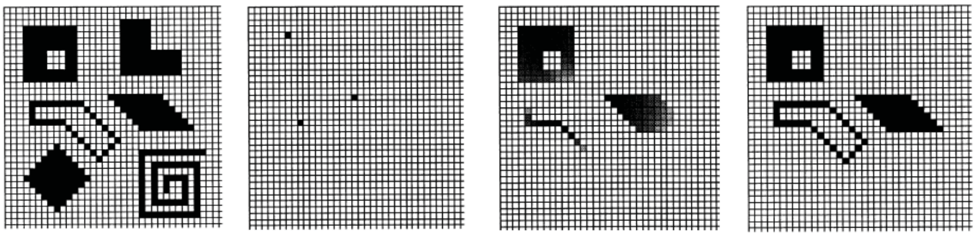


Fig. 8. (a) Test image; (b) initial state (marking points); (c) transient state; (d) output image

3.7 Finding the Intersection Points of One-Pixel Thin Lines from Two Binary Images

We consider two binary images $\mathbf{I}_1$ and $\mathbf{I}_2$ containing one-pixel thin lines or curves, among other compact objects. Superposing the two images, the objects will have overlapping areas. We propose to design the feedback and control templates such that the CNN detects only the intersection points (marked by single black pixels) of the thin lines present in the two images. All other overlapping areas will be ignored and will not appear in the output image. This task may be regarded as logic selective AND operation, since it only applies to some objects of particular shape from the images to be processed. One of the images is loaded as initial state while the other is applied to the input: $X(0) = \mathbf{I}_1$; $U = \mathbf{I}_2$. Interchanging the two images ($X(0) = \mathbf{I}_2$; $U = \mathbf{I}_1$) yields the same result. The feedback and control templates are identical, $A = B$, of the form (21). For this task we get the system:

$$2p - 8s + I > 1 \qquad 2p - 2s + I < -1 \qquad 14s - I > 1 \tag{30}$$

and a set of suitable parameter values is: $p = 3$; $s = -0.5$; $I = -8.5$. The described task is shown in Figure 9. The binary images (a) and (b) contain thin lines and various other objects. The first image is loaded as initial state, the second is applied to the input. The output result is the image (c) which contains only the intersection points of thin lines.

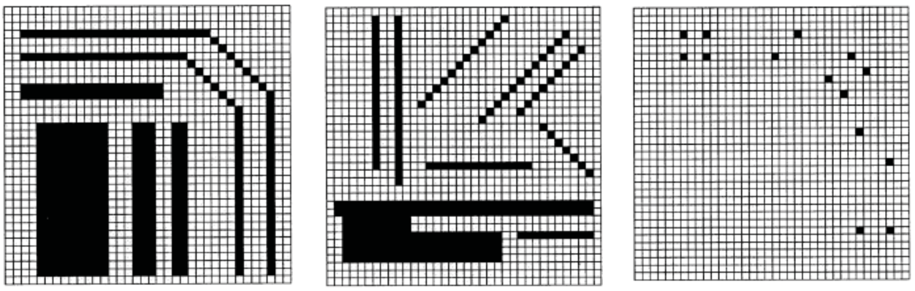

(a) (b) (c)

Fig. 9. (a) Image I1 as initial state; (b) Input image I2; (c) Resulted output image

4. Applications of CNNs in Image Linear Filtering

Although cellular neural networks are basically designed for nonlinear operation, an important class of applications of CNNs in image processing is also linear filtering (Crounse & Chua, 1995). Generally, the two-dimensional spatial filters are most commonly designed in the frequency domain, by imposing a set of specifications, or starting from various 1D prototype filters with given characteristics and applying frequency transformations which finally lead to the desired 2D filter. For each specific method the filter stability must also be ensured. We will approach here only the IIR spatial filters. Generally the existing design methods of 2D IIR filters rely to a large extent on 1D analog filter prototypes, using spectral transformations from s to z plane via bilinear or Euler transformations followed by z to (z_1, z_2) transformations. The most investigated were circular, elliptic-shaped, fan filters etc.

When CNNs are used as stable linear filters, the image **I** to be filtered is usually applied at the input (**U**=**I**) and the initial state is usually zero (**X**=0). The CNN cells must not reach

saturation during operation, which implies the restriction for the cell state: $\left|x_{ij}(t)\right| < 1$, $i, j = 1...N$.
The linear CNN filter is described by the spatial transfer function (Crounse & Chua, 1995):

$$H(\omega_1,\omega_2) = -B(\omega_1,\omega_2)/A(\omega_1,\omega_2) \tag{31}$$

where $A(\omega_1,\omega_2)$, $B(\omega_1,\omega_2)$ are the 2D Discrete Space Fourier Transforms of templates **A**, **B**.
An essential constraint for CNNs is local connectivity, specific to these parallel systems. Design specifications may lead to high-order 2D filters, requiring large-size templates, that cannot be directly implemented. The templates currently implemented are no larger than 3×3 or 5×5. Larger templates can only be implemented by decomposing them into a set of elementary templates. The most convenient are the separable templates, which can be written as a convolution of small-size templates. As we will show, the templates of the 2D filters resulted from 1D prototypes can always be written as convolution products of small-size templates, therefore the filtering can be realized in several steps. The filtering function will be a product of elementary rational functions.
For an 1D recursive CNN filter of order *N*, its transfer function has the following expression, where equal degrees are taken for numerator and denominator:

$$H(\omega) = \sum_{n=1}^{N} b_n \cos^n \omega \Bigg/ \sum_{m=1}^{N} a_m \cos^m \omega = B(\omega)/A(\omega) \tag{32}$$

The numerator and denominator of (32) can be factorized into first and second order polynomials in $\cos\omega$. For instance, the numerator can be decomposed as follows:

$$B(\omega) = k \cdot \prod_{i=1}^{n} (\cos\omega + b_i) \cdot \prod_{j=1}^{m} (\cos^2\omega + b_{1j}\cos\omega + b_{2j}) \tag{33}$$

with $n + 2m = N$ (filter order). Correspondingly, the template B ($1\times N$), can be decomposed into 1×3 or 1×5 templates.
A similar expression is valid for the denominator $A(\omega)$. Coupling conveniently the factors of $A(\omega)$ and $B(\omega)$, the filter transfer function (32) can be always written as a product of elementary functions of order 1 or 2, realized with pairs of 1×3 or 1×5 templates:

$$H(\omega) = H_1(\omega)\cdot H_2(\omega)\cdot \ldots \cdot H_k(\omega) \tag{34}$$

Here we will discuss zero-phase IIR CNN filters, i.e. with real-valued transfer functions, which correspond to symmetric control and feedback templates. In IIR spatial filtering with a 2D filter $H(\omega_1,\omega_2)$, the final state can be written:

$$X(\omega_1,\omega_2) = U(\omega_1,\omega_2)\times H(\omega_1,\omega_2) = U(\omega_1,\omega_2)\times H_1(\omega_1,\omega_2)\times H_2(\omega_1,\omega_2)\ldots H_n(\omega_1,\omega_2) \tag{35}$$

The image $X_1(\omega_1,\omega_2)=U(\omega_1,\omega_2)\cdot H_1(\omega_1,\omega_2)$ resulted in the first filtering step is re-applied to CNN input, giving the second output image: $X_2(\omega_1,\omega_2)=X_1(\omega_1,\omega_2)\cdot H_2(\omega_1,\omega_2)$ and so on, until the whole filtering is achieved. The frequency response $H(\omega_1,\omega_2)$ can be written:

$$H(\omega_1,\omega_2)=\prod_k H_k(\omega_1,\omega_2)=\prod_k B_k(\omega_1,\omega_2)\Big/\prod_k A_k(\omega_1,\omega_2) \tag{36}$$

which implies a cascade interconnection of IIR filters. The separable templates **A** and **B** can be written as convolution products, where B_i, A_j are elementary (3×3 or 5×5) templates:

$$B=B_1*B_2*\ldots*B_n \qquad A=A_1*A_2*\ldots*A_m \tag{37}$$

4.1 Design Methods for Circularly-Symmetric Filters

In this section we propose an efficient design technique for 2D circularly-symmetric filters, based on 1D prototype filters. Circular filters are currently used in many image processing applications. Given a 1D filter function $H_p(\omega)$, the corresponding 2D filter $H(\omega_1,\omega_2)$ results through the frequency transformation $\omega\to\sqrt{\omega_1^2+\omega_2^2}$:

$$H(\omega_1,\omega_2)=H_p\left(\sqrt{\omega_1^2+\omega_2^2}\right) \tag{38}$$

A currently-used approximation of the function $\cos\sqrt{\omega_1^2+\omega_2^2}$ is given by:

$$\cos\sqrt{\omega_1^2+\omega_2^2}\cong C(\omega_1,\omega_2)=-0.5+0.5(\cos\omega_1+\cos\omega_2)+0.5\cos\omega_1\cos\omega_2 \tag{39}$$

and corresponds to the 3×3 template:

$$\mathbf{C}=\begin{bmatrix}0.125 & 0.25 & 0.125\\ 0.25 & -0.5 & 0.25\\ 0.125 & 0.25 & 0.125\end{bmatrix} \tag{40}$$

Let us consider a 1D filter $H_P(\omega)$ described by the symmetric templates B_P, A_P of radius *R*:

$$\begin{aligned}B_P&=[\ldots\ b_2\ \ b_1\ \ b_0\ \ b_1\ \ b_2\ \ldots]\\ A_P&=[\ldots\ a_2\ \ a_1\ \ a_0\ \ a_1\ \ a_2\ \ldots]\end{aligned} \tag{41}$$

Applying DSFT and using trigonometric identities we finally get the transfer function as a ratio of polynomials in $\cos\omega$. Its denominator has the factorized form (with $n+2m=N$, the filter order):

$$A_p(\omega) = c_0 + \sum_{k=1}^{R} c_k (\cos\omega)^k = k \cdot \prod_{i=1}^{n} (\cos\omega + a_i) \cdot \prod_{j=1}^{m} (\cos^2\omega + a_{1j}\cos\omega + a_{2j}) \tag{42}$$

Substituting $\cos\omega$ by $C(\omega_1, \omega_2)$ given by (39) in the prototype function $A_P(\omega)$, we obtain:

$$A(\omega_1, \omega_2) = A_p\left(\sqrt{\omega_1^2 + \omega_2^2}\right) = c_0 + \sum_{k=1}^{R} c_k \cdot C^k(\omega_1, \omega_2) \tag{43}$$

Therefore, the design of the 2D circular filter consists in substituting $\cos\omega$ with $C(\omega_1, \omega_2)$ in each factor of $H_P(\omega)$; $A(\omega_1, \omega_2)$ will have the form, corresponding to (42):

$$A(\omega_1, \omega_2) = k \cdot \prod_{i=1}^{n} \left(C(\omega_1, \omega_2) + a_i\right) \cdot \prod_{j=1}^{m} \left(C^2(\omega_1, \omega_2) + a_{1j} \cdot C(\omega_1, \omega_2) + a_{2j}\right) \tag{44}$$

Since the 1D prototype filter function can be factorized, the 2D circularly-symmetric filter is separable, a major advantage in implementation. The large-size template **A** corresponding to $A(\omega_1, \omega_2)$ results directly decomposed into elementary (3×3 or 5×5) templates:

$$A = k \cdot (C_1 * \ldots * C_i * \ldots C_n) * (D_1 * \ldots * D_j * \ldots D_m) \tag{45}$$

If we use the 3×3 template **C** given in (40) to approximate the 2D cosine function, each template C_i from (45) is obtained from **C** by adding a_i to the central element. Each 5×5 template D_j results as $D_j = C * C + b_{1j} \cdot C_1 + b_{2j} \cdot C_0$, where C_0 is a 5×5 zero template, with central element one; C_1 is a 5×5 template obtained by bordering B with zeros.

4.2 Maximally-Flat Zero-Phase Filters

We propose a method of designing 2D spatial filters of different types starting from approximations commonly used in temporal filters. However, we will design spatial filters which will preserve only the magnitude characteristics of their temporal prototypes, while the designed filters will be zero-phase; their frequency response will be real-valued. The design of the desired 2D filters will be achieved in several steps, detailed as follows.

The starting point is a discrete prototype filter of a given type and desired specifications, with a transfer function $H(z)$. Using this discrete filter we first derive an 1D spatial filter. This 1D filter will inherit only the magnitude characteristics of its temporal counterpart, while its phase will be zero throughout the frequency domain, namely the range $[-\pi, \pi]$. We consider the magnitude $|H(\omega)|$ of $H(z) = H(e^{j\omega})$. In order to find a zero-phase 1D filter transfer function which approximates $|H(\omega)|$, we can make the change of frequency variable:

$$\omega = \arccos x \Leftrightarrow x = \cos\omega \tag{46}$$