Neurofuzzy Selfmade Network for Image Processing based on CNN Networks

José Antonio Medina Hernández[1,2], Felipe Gómez Castañeda[1], José Antonio Moreno Cadenas[1]
[1] Department of Electrical Engineering. CINVESTAV
Av. Instituto Politécnico Nacional 2508, 07360, México D.F., México, Phone (0155) 50613800 Ext 6262
[2] Aguascalientes Autonomous University, Department of Mathematics and Physics, Aguascalientes, México
E-mail:jamedihe@correo.uaa.mx, fgomez@cinvestav.mx, jmoreno@cinvestav.mx

***Abstract*— The Cellular Neural Network (CNN) is very efficient for image processing tasks. However, there are limitations in its processing capabilities because some tasks can not be learned by a single CNN network. In recent years it has been accepted that a set of CNNs connected in parallel can realize more complex image processing tasks than a single CNN. Also recently it has been reported an architecture of neurofuzzy adaptable network (SIMAP) able to construct its structure and membership functions using only input-output data. In this paper is described the way of associating a CNN for every fuzzy rule in the SIMAP network for making complex image processing tasks, which are impossible to do for a unique CNN.**

Keywords: Neurofuzzy system, membership function, rectangular metric, similarity function, cluster, fuzzy-ARTMAP network, SIMAP network, cellular neural network (CNN), CNN templates design.

I. INTRODUCTION

A Cellular Neural Network (CNN) is a set of locally interconnected processing units arranged in a rectangular bidimensional grid [1]-[4]. This structure is very useful for image processing tasks [2]-[6]. Two important aspects of these networks are their real time processing capacities, and facilities for VLSI implementation. The CNN network has an invariant space local interconnection. A neigborhood of radius $r = 1$ has associated two 3×3 matrices A and B, named retroalimentation and control templates, and a bias parameter z. The cloning template is the set $T = \{\mathbf{A}, \mathbf{B}, z\}$. Designing a 19 parameters cloning template is often a hard task [8]-[12].

In recent years it has been accepted that a set of CNN connected in parallel over a neural network can realize more complex image processing tasks that a single CNN [7],[13]-[15]. There are several self-made neural networks able to make identification of structure and parameters using only the input-output data [16]-[18].

In this paper we describe a neurofuzzy system containing a CNN for every fuzzy rule. It can realize more complex image processing task that those composed by a single CNN. The neurofuzzy structure is extracted from the SIMAP network [18]. The paper is organized in the following way: In section II we describe the architecture and the performance. In section III the training and recall algorithms are given. In section IV some experimental results are shown. Conclusions are drawn in section V.

II. SIMTCNN NETWORK ARCHITECTURE

Let I and I_d be two images with $M \times N$ pixels. I is a grey level input image and I_d is the desired black and white output image. We shall transform the input image I in the desired binary output image I_d. To reach this aim, we divide the input and output images in disjoint windows of dimension 3×3. Next, we use the SIMAP network to compress these input and output window-sets, creating input and output clusters and making a mapping between them. A CNN network is associated to every fuzzy rule of the map such that a training input window is transformed in the respective desired output window. Thus, for every fuzzy rule we obtain the matrices A, B and a bias parameter z of its corresponding CNN. Briefly, a map between two sets of clusters is stablished, transforming each input windows to a unique output window using a set of CNN networks connected in parallel. The resulting network is named SIMTCNN (SImilarity Map Transformed by CNN). Its architecture is shown in Fig. 1.

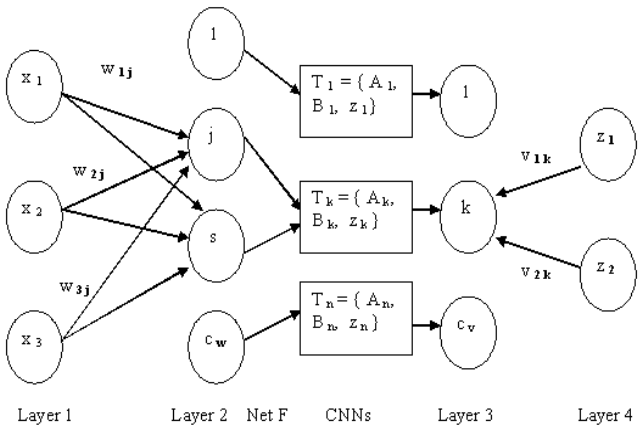

Fig. 1. SIMTCNN Architecture.

Weights w_{ij} from layer 1 to unit j of layer 2 form a vector $\mathbf{w_j}$ such that we can interpret $\mathbf{w_j}$ as the center of a cluster represented by unit j of layer 2. We name the vector $\mathbf{w_j}$ an input cluster. Weights v_{ik} from layer 4 arriving to unit k of layer 3 form a vector $\mathbf{v_k}$ such that we can interpret $\mathbf{v_k}$ as the center of a cluster represented by unit k of layer 3. We name the vector $\mathbf{v_k}$ an output cluster.

The interconnecting network F is used to associate every input-cluster to only one output-cluster. All the vectors belonging to the same cluster have some level of similarity.

The layers 1 and 4 contain the input and output nodes, receiving the pairs of training vectors $(\mathbf{x_i}, \mathbf{z_i})$. The input vector $\mathbf{x_i}$ is assigned to some input-cluster, and the output vector $\mathbf{z_i}$ is assigned to some output-cluster. The way of assigning an input or output vector to a cluster is measuring the similarity of the vector with the cluster prototype. This measure lets take decisions to create new clusters and grow the structure.

We denote with $\mathbf{w_j}$ the input-cluster at which input-vector $\mathbf{x}$ has greatest similarity and denote with $\mathbf{v_k}$ the output-cluster at which the output-vector $\mathbf{z}$ has greatest similarity. For storing the pair $(\mathbf{x}, \mathbf{z})$ in the network, we should verify that the input-cluster $\mathbf{w_j}$ is not associated via F-connections with an output-cluster different to $\mathbf{v_k}$. If $\mathbf{w_j}$ is not associated with $\mathbf{v_k}$, we increment the similarity level ρ_w in the input-clusters, to prevent a input-cluster be associated with two different output-clusters. In such a case, a rise of value in the similarity-parameter ρ_w yields a new cluster $\mathbf{w}$ in layer 2. The corresponding output-vector $\mathbf{z_j}$ can be absorbed by the cluster $\mathbf{v_k}$ or a new category in layer 3 can be produced, depending on the similarity of $\mathbf{z_j}$ with the prototype $\mathbf{v_k}$. This is the creating-structure mechanism used by SIMAP [18]. During the creation of structure the CNN networks are invisible. After the creation of structure, the CNN netwoks are trained to transform an input-window, similar to the input cluster $\mathbf{w_j}$, in an output-window, similar to the output cluster $\mathbf{w_j}$.

In the recall-phase, every input-vector $\mathbf{x}$ has associated an output-vector $\mathbf{z}$, calculated using the information contained in the input-weights $\mathbf{w_j}$, the output-weights $\mathbf{v_k}$, the connections of the net F and the templates T_j. The final result is a non-linear continuous map, with adjustable softness. The complete process requires additional units not shown in Fig. 1. In the next section we describe in detail the training and recall algorithms for the SIMTCNN architecture.

III. TRAINING AND RECALL ALGORITHMS

A. SIMILARITY MEASURES

We define as a *similarity measure* a function $S : \Re^N \times \Re^N \to \Re$ such that when $S(\mathbf{x}, \mathbf{w})$ takes high values there is a high likeness between the vectors $\mathbf{x}$ and $\mathbf{w}$ of $\Re^N$, and when $S(\mathbf{x}, \mathbf{w})$ takes values near to zero there is a great difference between vectors $\mathbf{x}$ and $\mathbf{w}$. In the following argument we consider only vectors $\mathbf{x}$ in $[-1, 1]^N$. When this is not the case we can transformate linearly the data to satisfy such condition. We denote with $|\mathbf{x}| = \sum_{i=1}^{N} |x_i|$ the rectangular metric of the vector $\mathbf{x} = (x_1, x_2, ..., x_N) \in \Re^N$. To define a similarity function we consider the rectangular metric

$$d(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N} |x_i - w_i|$$

If $\mathbf{x}$ and $\mathbf{w}$ are in $[-1, 1]^N$, we can define the standard similarity $S_k(\mathbf{x}, \mathbf{w_j}) = \frac{2N - d(\mathbf{x}, \mathbf{w_j})}{2N}$ contained in the interval [0,1].

B. TRAINING ALGORITHM FOR SIMTCNN NETWORK

In the following we use two standard similarities S_w and S_v as defined in previous section. The training algorithm for the SIMTCNN network is similar to the used by SIMAP [18], but the training of CNNs described in section C is incorporated:

Terminology.

c_w= total number of input-clusters

ρ_w=minimum required relative similarity between a candidate input-cluster $\mathbf{w}$ and a input-vector $\mathbf{x}$ to incorporate $\mathbf{x}$ to $\mathbf{w}$.

n_j=number of input vectors represented by the input-cluster $\mathbf{w_j}$

g_w=position of the candidate input-cluster where the input-vector $\mathbf{x_i}$ is going to be assigned.

ϵ=small positive number

c_v=total number of output-clusters

ρ_v=minimum required relative similarity between a candidate output-cluster $\mathbf{v}$ and output-vector $\mathbf{z}$ to incorporate $\mathbf{z}$ to $\mathbf{v}$.

m_k=number of output vectors represented by the output-cluster $\mathbf{v_k}$

g_v=position of the candidate output-cluster where the output-vector $\mathbf{z_i}$ is going to be assigned.

Algorithm.

1. Generate the first input-cluster $\mathbf{w_1}$ and the first output-cluster $\mathbf{v_1}$ using the first training pair $(\mathbf{x}, \mathbf{z})$. Take $\mathbf{w_1} = \mathbf{x}$ and $\mathbf{v_1} = \mathbf{z}$.
2. For every training pair $(\mathbf{x}, \mathbf{z})$ do steps 3-5.
3. Find the input-cluster $\mathbf{w_j}$ and the output-cluster $\mathbf{v_k}$ such that $S(\mathbf{x}, \mathbf{w_j})$ and $S(\mathbf{z}, \mathbf{v_k})$ are maximum.
4. If $S(\mathbf{x}, \mathbf{w_j}) < \rho_w$ open a new cluster $c_w + 1$ for $\mathbf{x}$, doing $c_w = c_w + 1; g_w = c_w; n_{g_w} = 1; \mathbf{w}_{g_w} = \mathbf{x}$

Also do steps 4.1 to 4.3:

4.1. If $S(\mathbf{z}, \mathbf{v_k}) < \rho_v$ open a new cluster $c_v + 1$ for $\mathbf{z}$, doing $c_v = c_v + 1; g_v = c_v; m_{g_v} = 1; \mathbf{v}_{g_v} = \mathbf{z}$;

4.2. If $S(\mathbf{z}, \mathbf{v_k}) \geq \rho_v$ apply the learning rule:

$\mathbf{v_k} = (m_k \mathbf{v_k} + \mathbf{z})/(m_k + 1)$; take $m_k = m_k + 1; g_v = k$

4.3. Stablish a connection between unit g_w in layer 2 and unit g_v in layer 3 via the F network, and go to step 2.

5. If $S(\mathbf{x}, \mathbf{w_j}) \geq \rho_w$ do steps 5.1. to 5.2.

5.1. If unit j of layer 2 is connected to unit k of layer 3, apply learning rules for $\mathbf{w_j}$ and $\mathbf{v_k}$ in the following way:

$\mathbf{w_j} = (n_j \mathbf{w_j} + \mathbf{x})/(n_j + 1); n_j = n_j + 1;$

$\mathbf{v_k} = (m_k \mathbf{v_k} + \mathbf{z})/(m_k + 1); m_k = m_k + 1;$

Go to step 2.

5.2. If criterion in 5.1 is not satisfied, take ρ_w=$S(\mathbf{x}, \mathbf{w_j})+\epsilon$ and return to step 3.

6. For $j = 1$ to c_w apply step 6.1

6.1. For every input vector $\mathbf{x}$ assigned to the input-cluster $\mathbf{w_j}$ apply the CNN learning rules to the template $T_j = \{A_j, B_j, z_j\}$, using the training pair $(\mathbf{x}, \mathbf{z})$.

7. Finish.

C. TRAINING ALGORITHM FOR THE CNN NETWORKS

In this section, we review briefly the terminology and the results reported in [12] for training the CNNs. We start dividing the input image I and the desired output image I_d into disjoin sets of 3×3 windows $\{V_i\}$ and $\{V_i^d\}$, respectively. We suppose that the input window V_i is assigned to the input cluster j, which is associated to the output cluster k. The CNN network associated to the corresponding fuzzy rule should be able to learn two matrices A, B and a bias parameter z such that the window V_i is transformated to the window V_i^d. The learning process proceeds iteratively through time. $x_{ij}(t_n)$ and $y_{ij}(t_n)$ are the activation and output functions of the cell ij at time t_n, and y_{ij}^d is the desidered output value, where $1 \leq i \leq M$, $1 \leq j \leq N$ and t_n is the instant for transforming the n-pixel of I to the n-pixel of I_d, where $1 \leq n \leq MN$. The symbol u_{ij} is the gray level of the ijth-pixel contained on the input image. The output $y_{ij}(t_n)$ of the cell ij can be approximated by

$$y_{ij}(t_n) = \frac{2}{1 + e^{-\sigma x_{ij}(t_n)}} - 1$$

where the activation value $x_{ij}(t_n)$ can be approximated by the equation

$$\frac{dx_{ij}(t_n)}{dt} = -x_{ij}(t_n) + \sum_{kl \in V_r(ij)}^{n} A_{kl}(t_n) y_{kl}(t_n) + \sum_{kl \in V_r(ij)}^{n} B_{kl}(t_n) u_{kl} + z_{ij}(t_n) \qquad (1)$$

This activation equation is similar to the CNN activation equation, but coefficients A_{kl} and B_{kl} are functions of time t_n. Similarly, we denote with $x_{kl}(t_n)$ and $y_{kl}(t_n)$ the activation and output functions at time t_n for the cells kl contained in the neighborhood of cell ij. At time t_n we want to minimize the error function

$$\epsilon_{ij}(t_n) = \frac{1}{2}[y_{ij}(t_n) - y_{ij}^d]^2$$

respect to the parameters $A_{kl}(t_n)$, $B_{kl}(t_n)$ and $z(t_n)$, associated to the radius 1 neighborhood of cell ij, using a stochastic gradient descent method. The learning of parameters proceeds according with relations

$$\Delta A_{kl}(t_n) = -\eta \frac{\partial \epsilon_{ij}(t_n)}{\partial A_{kl}(t_n)} \quad \Delta B_{kl}(t_n) = -\eta \frac{\partial \epsilon_{ij}(t_n)}{\partial B_{kl}(t_n)}$$

$$\Delta z(t_n) = -\eta \frac{\partial \epsilon_{ij}(t_n)}{\partial z(t_n)}$$

where η is the learning rate. The resulting learning rules [12] are

$$A_{kl}(t_{n+1}) = A_{kl}(t_n) - \eta[y_{ij}(t_n) - y_{ij}^d(t_n)]\frac{\sigma}{2}[1 + y_{ij}(t_n)][1 - y_{ij}(t_n)] \cdot \left[y_{kl}(t_n) + A_{kl}(t_n)\frac{y_{kl}(t_n) - y_{kl}(t_{n-1})}{A_{kl}(t_n) - A_{kl}(t_{n-1})}\right] \qquad (2)$$

$$B_{kl}(t_{n+1}) = B_{kl}(t_n) - \eta[y_{ij}(t_n) - y_{ij}^d(t_n)]\frac{\sigma}{2}[1 + y_{ij}(t_n)][1 - y_{ij}(t_n)]u_{kl} \qquad (3)$$

$$z_{kl}(t_{n+1}) = z_{kl}(t_n) - \eta[y_{ij}(t_n) - y_{ij}^d(t_n)]\frac{\sigma}{2}[1 + y_{ij}(t_n)][1 - y_{ij}(t_n)] \qquad (4)$$

The rules (2),(3) and (4) for the production of the CNN templates should be used in every cell kl contained in the neighborhood of cell ij. At the beginning of the learning, the values $y_{kl}(t_0)$, $A_{kl}(t_0)$ and $A_{kl}(t_1)$ can be taken equal to zero. Rules (2), (3) and (4) should be used for every training pair ij of the input and output pixels. For an image with $N \times M$ pixels there should be the same number of learning steps at every iteration. The number of iterations is defined as the number of times the input and output images are used. The number of iterations is determined by a stop criterion. There are two stop criteria:
1) Compute the norm between two cloning templates of consecutive learning steps. If the norm is below a constant level TOL, the algorithm stops.
2) If the number of iterations is greater than an interger NMAX, the algorithm stops.

Training Algorithm for the CNNs

Input: Training images.
Output: Matrices A, B and z parameter.
1)Initiate A, B and z to zero.
2)While stop criterion is not satisfied do step 2.1:
2.1)For every pair of corresponding input and output pixels, use the learning rules (2), (3) and (4).
3)Stop.

The first stop criterion is desirable, but convergence is not always present, so we must use second criterion. A problem of lack in convergence is present when:
i)We try that the CNN learns a task imposible to learn by a single CNN, or
ii)There are incompatibilities in the training images (for example, when we are specifying two different desired output-pixel sets for the same set of input pixels).

D. RECALL ALGORITHM FOR THE SIMTCNN NETWORK

When we have obtained the structure of the SIMAP network and the CNNs have been trained, we can extract the joint performance of the system. The recall algorithm is the following:

Terminology.

k=Number of input clusters in the training phase.
$\mathbf{x}$=Input vector in the recall phase.
$\mathbf{w_i}$=Representing vector of the $i-th$ input cluster.
$\mathbf{v_i}$=Representing vector of the $j-th$ output cluster (asociated to the input cluster $\mathbf{w_i}$).
$S_i = S_w(\mathbf{x}, \mathbf{w_i})$=relative similarity between vectors

$\mathbf{x}$ and $\mathbf{w_i}$
$\mathbf{z}$=Output vector computed by the SIMTCNN network.

Algorithm.

0. Set a value $p > 1$ (a very large p value corresponds to very crisp membership functions [18]).
1. For every input vector $\mathbf{x}$ apply steps 2 to 3.
2. For every input prototype $\mathbf{w_i}$ calculate the relative similarity $S_i = S_w(\mathbf{x}, \mathbf{w}_i)$, $i = 1, 2, .., k$.
3. Process the input window $\mathbf{x}$ using the CNN networks associated to the output clusters, generating the output windows $\mathbf{V_1}, ..., \mathbf{V_k}$ (no necessarily different)
4. Compute the output network

$$\mathbf{z} = \frac{\sum_{i=1}^{k} S_i^p \mathbf{V_i}}{\sum_{i=1}^{k} S_i^p} \qquad \text{p} > 1$$

5. Finish.

The mechanism of inference can be tuned using a similar criterion to the used for SIMAP network.

IV. EXPERIMENTAL RESULTS

In this section we present several examples to ilustrate the performance of the SIMTCNN network.

Example 1. In this example we see the capacity of the SIMTCNN network to detect defects in color filters of TFT's for liquid control displays [13]-[15]. This is a challenging task because the defects shoud be detected from an image having a background with many different gray values. A SIMTCNN network was used to detect these defects using the training images at Fig. 2. Every image in Fig. 2 contains 1288 disjoint windows of dimension 3×3. Using the vigilance parameters $\rho_w = \rho_v = 0.2$ the network yields 64 input clusters and 5 output clusters. The network also associates 5 cloning templates:

$$A_1 = \begin{pmatrix} 0.28 & 0.53 & -0.16 \\ 0.50 & -0.43 & -0.26 \\ 0.36 & -0.24 & -0.14 \end{pmatrix};$$

$$B_1 = \begin{pmatrix} -0.06 & -0.26 & -0.18 \\ -0.13 & -0.48 & -0.30 \\ -0.09 & -0.27 & -0.17 \end{pmatrix};\ z_1 = -14.91$$

$$A_2 = \begin{pmatrix} -1.56 & 1.44 & 0.22 \\ -1.53 & -2.09 & 1.87 \\ -0.09 & 1.75 & 1.26 \end{pmatrix};$$

$$B_2 = \begin{pmatrix} -0.51 & 1.63 & -0.45 \\ 0.02 & 0.37 & -0.66 \\ -0.26 & 1.15 & -0.06 \end{pmatrix};\ z_2 = 8.11$$

$$A_3 = \begin{pmatrix} 0.07 & -0.06 & 0.16 \\ 0.27 & -1.56 & -0.32 \\ 0.90 & -0.31 & -0.41 \end{pmatrix};$$

$$B_3 = \begin{pmatrix} -0.58 & -0.23 & 0.17 \\ -0.22 & -1.03 & -0.47 \\ -0.12 & -0.17 & -0.13 \end{pmatrix};\ z_3 = 11.36$$

$$A_4 = \begin{pmatrix} 1.19 & -2.04 & 0.62 \\ 1.38 & -2.56 & 0.56 \\ -0.29 & -0.45 & -1.34 \end{pmatrix};$$

$$B_4 = \begin{pmatrix} 1.12 & 1.80 & 1.13 \\ 0.03 & 1.35 & -0.13 \\ -1.73 & -2.66 & -2.24 \end{pmatrix};\ z_4 = 15$$

$$A_5 = \begin{pmatrix} 0.29 & 1.51 & 0.05 \\ 2.01 & -0.08 & 0.97 \\ -1.11 & -1.91 & -1.35 \end{pmatrix};$$

$$B_5 = \begin{pmatrix} -1.23 & -2.54 & -1.78 \\ 2.08 & 4.29 & 2.28 \\ -0.77 & -0.85 & -0.53 \end{pmatrix};\ z_5 = -7.57$$

Fig. 3 shows the transformation storaged by the network. In Figs. 4, 5 and 6 is shown the detection of defects produced by the SIMTCNN network for three test images. It can be appreciated that the output images of the SIMTCNN system indicate satisfactorily the position of the defects on the LCD displays.

Example 2. In this example is shown the capacity of the SIMTCNN network to detect and isolate several obstacles in a highway. The system acts as a filter, ignoring the lateral information, but detecting objects on the way. In Fig. 7 are shown two training images, containing 322 disjoint 3×3 windows. Fig. 8 shows the transformation storaged by the SIMTCNN network when $\rho_w = 0.4$, $\rho_v = 0.4$ and $p = 50$. The network yields 28 input clusters and 6 output clusters, along with the following templates:

$$A_1 = \begin{pmatrix} 0.45 & 0.87 & 0.29 \\ 0.50 & -0.88 & 0.31 \\ 0.29 & -0.14 & 0.08 \end{pmatrix};$$

$$B_1 = \begin{pmatrix} 0.18 & 0.08 & 0.25 \\ 0.10 & -0.93 & 0.25 \\ 0.06 & -0.15 & 0.05 \end{pmatrix};\ z_1 = 15.74$$

$$A_2 = \begin{pmatrix} 0.18 & 0.12 & 0.36 \\ 0.51 & -0.74 & 0.58 \\ 0.53 & 0.35 & 0.62 \end{pmatrix};$$

$$B_2 = \begin{pmatrix} 0.18 & -0.07 & 0.36 \\ 0.44 & -0.84 & 0.57 \\ 0.51 & 0.29 & 0.59 \end{pmatrix};\ z_2 = -11.28$$

$$A_3 = \begin{pmatrix} 0.47 & 0.22 & -0.09 \\ 0.25 & -0.24 & -0.07 \\ 0.11 & 0.67 & -0.04 \end{pmatrix};$$

$$B_3 = \begin{pmatrix} 0.52 & 0.74 & -0.27 \\ 0.22 & 1.19 & -0.04 \\ 0.15 & 0.96 & 0.07 \end{pmatrix};\ z_3 = 0.54$$

$$A_4 = \begin{pmatrix} -0.29 & 1.11 & -0.04 \\ 0.09 & -0.67 & 0.05 \\ -0.03 & 1.23 & -0.88 \end{pmatrix};$$

$$B_4 = \begin{pmatrix} -0.51 & 1.92 & 0.05 \\ 0.56 & 1.00 & -0.31 \\ 0.43 & 1.02 & -0.63 \end{pmatrix}; \; z_4 = 0.4104$$

$$A_5 = \begin{pmatrix} -0.10 & -0.04 & 0.11 \\ 0.14 & 0.64 & 0.20 \\ 0.10 & -0.03 & -0.09 \end{pmatrix};$$

$$B_5 = \begin{pmatrix} -0.12 & -0.05 & 0.11 \\ 0.13 & 0.75 & 0.21 \\ 0.12 & -0.04 & -0.10 \end{pmatrix}; \; z_5 = 0.02$$

$$A_6 = \begin{pmatrix} -0.05 & 0.08 & -0.12 \\ 0.08 & 0.64 & 0.16 \\ -0.06 & 0.16 & -0.02 \end{pmatrix};$$

$$B_6 = \begin{pmatrix} -0.07 & 0.11 & -0.16 \\ 0.12 & 0.74 & 0.19 \\ -0.08 & 0.20 & -0.03 \end{pmatrix}; \; z_6 = -0.04$$

In Figs. 9 and 10 are shown the results by the network for two test-images. It can be observed that the network is able to recognize very different obstacles and the highway contour. These results indicate that arbitrary objects in an homogoneous highway can be detected by the SIMTCNN network.

V. CONCLUSIONS

In this paper we have used the SIMAP architecture to implement a visual processing system named SIMTCNN. Using the information contained in the pairs $\{(\mathbf{x_i}, \mathbf{z_i})\}_{i=1}^{n}$ of training windows we can construct a non-linear map $\mathbf{f}(\mathbf{x})$ such that $\mathbf{z_i} \approx f(\mathbf{x_i})$. The architecture contains a set of cellular neural networks for transforming the input windows $\mathbf{x_i}$ into the output windows $\mathbf{z_i}$. SIMTCNN is a modification of the SIMAP network such that it can learn automatically the network structure, the input and output prototypes, and the cloning templates. The problem of defining the input and output fuzzy sets, and the fuzzy rules, is resolved automatically by the SIMTCNN network using the same algorithms of the SIMAP system. The training of the cellular neural networks is done using this neurofuzzy structure. The structure and output of the SIMTCNN network can be adjusted modifying the vigilance parameters and the inference parameter p, respectively [18].

We have presented two examples showing how the SIMTCNN network is useful for the detection of objects in scenes, containing homogeneous or heterogeneous backgrounds.

ACKNOWLEDGEMENTS

The authors thank the funding of The National Council for Science and Technology, CONACYT-Mexico under grant 82579.

References

[1] L. Chua and L. Yang, "Cellular Neural Networks: Theory", *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1272-1290, Oct. 1988.

[2] L. Chua and L. Yang, "Cellular Neural Networks: Applications", *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1272-1290, Oct. 1988.

[3] L. Chua, *CNN: A Paradigm for Complexity*, World Scientific, 1998.

[4] T. Roska and L. O. Chua, "The CNN Universal Machine", *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 163-173, Mar. 1993.

[5] T. Roska, L. Kek, L. Nemes, A. Zarandy, M. Brendel, and P. Szolgay, *CNN Software Library*, in CADETWin, Budapest, Hungary: Computer and Automation Institute of the Hungarian Academy of Sciences, 1998.

[6] T. Roska, L. Kék, L. Nemes, A. Zarandy, M. Brendel, and P. Szolgay, *CADETWin*, Budapest, Hungary: Computer and Automation Institute of the Hungarian Academy of Sciences, 1998.

[7] I. Szatmari, D. Balya, G. Timar, C. Rekeczky and T. Roska, "Multi-Channel Spatio-Temporal Topographic Processing for Visual Search and Navigation", *Proc. SPIE Microtechnologies for the New Millenium*, Gran Canaria, Spain, May 2003, Paper 5119 38.

[8] T. Kozek, T. Roska, and L. O. Chua, "Genetic Algorithm for CNN Template Learning", *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 392-402, Mar. 1993.

[9] J. A. Nossek, "Design and Learning with Cellular Neural Networks", *Int. J. Circuit Theory Applicat.*, vol. 24, pp. 15-24, 1996.

[10] A. Zarandy, "The Art of CNN Template Design", Int. J. Circuit Theory Applicat., vol. 27, no. 1, pp. 5-23, 1999.

[11] M. Hanggi and G. S. Moschytz, "An Exact and Direct Analytical Method for The Design of Optimally Robust CNN Templates", *IEEE Trans. Circuits Syst.*, vol. 46, pp. 304-311, Feb. 1999.

[12] J. A. Medina-Hernandez, F. Gomez-Casteneda and J. A. Moreno-Cadenas, "A Method for Designing CNN Templates", *Proc. 2007 4th Int. Conf. on Elec. and Electronic Eng.*, 2007, pp. 153-156

[13] Chin-Teng Lin, Chun-Lung Chang and Jen-Feng Chung, "New Horizon for CNN: With Fuzzy Paradigms for Multimedia", *IEEE Circuits and Systems Magazine*, pp 20-35, Second Quarter 2005.

[14] Chin-Teng Lin, Chun-Lung Chang and Wen-Chang Cheng, "A Recurrent Fuzzy Cellular Neural Network System with Automatic Structure and Template Learning", *IEEE Transactions on Circuits and Systems*, Vol. 51, No 5, pp 1024-1035, May 2004.

[15] Chin-Teng Lin, Chun-Lung Chang and Wen-Chang Cheng, "A Recurrent Fuzzy Coupled Cellular Neural Network System with Automatic Structure and Template Learning", *IEEE Transactions on Circuits and Systems II*, Vol. 53, No 8, pp 602-606, August 2006.

[16] C. F. Juang and C. T. Lin, "An Online Self-Constructing Neural Fuzzy Inference Network and Its Applications", *IEEE Trans. Fuzzy Syst.*, vol. 6, pp 12-32, Feb. 1998.

[17] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hell, 1996.

[18] J. A. Medina-Hernandez, F. Gomez-Castaneda, J. A. Moreno-Cadenas, "An Evolving Fuzzy Neural Network Based on the Mapping of Similarities", *IEEE Trans. Fuzzy Syst.*, vol. 17, pp 1379-1396, Dec. 2009.

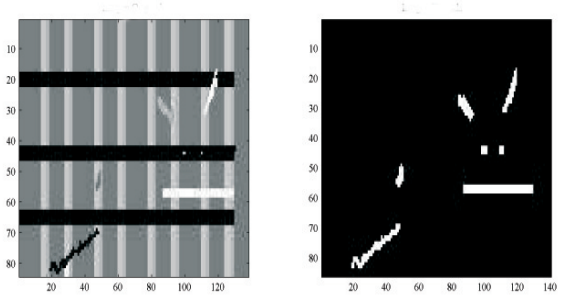

Fig. 2. Input and output images used for training of the SIMTCNN.

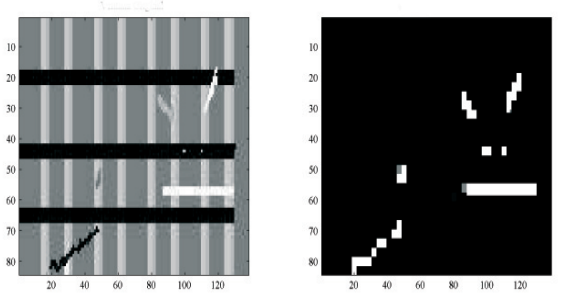

Fig. 3. Transformation storaged by the SIMTCNN network.

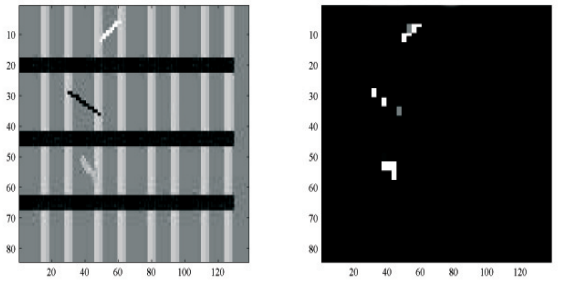

Fig. 4. Response of SIMTCNN for the first set of defects.

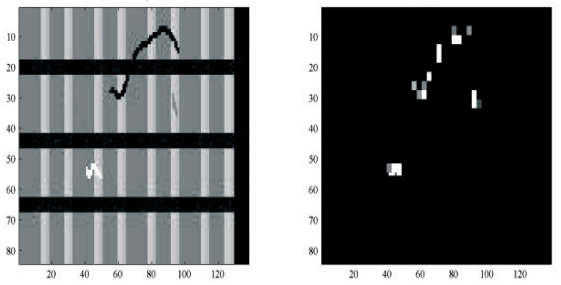

Fig. 5. Response of SIMTCNN for the second set of defects.

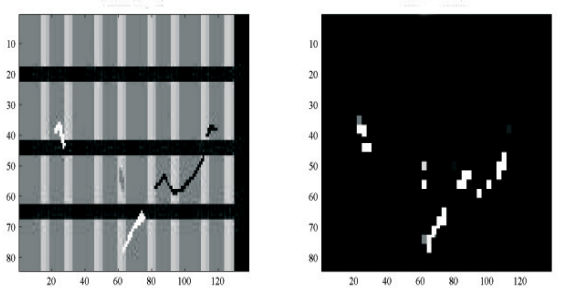

Fig. 6. Response of SIMTCNN for the third set of defects.

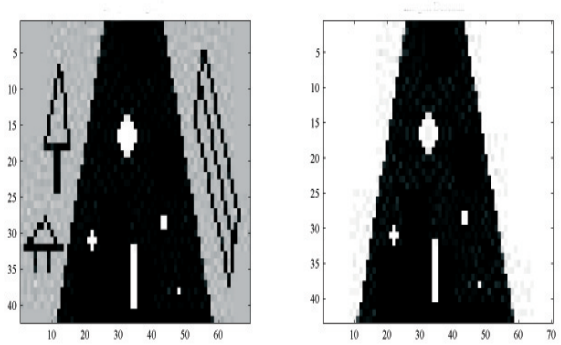

Fig. 7. Input and output images used for training.

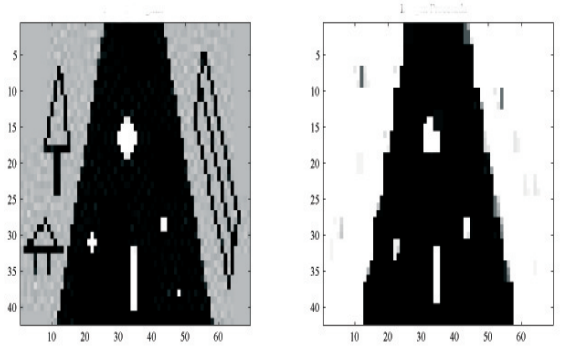

Fig. 8. Transformation stored by the network.

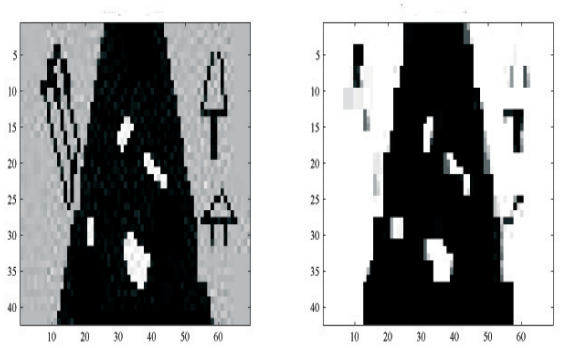

Fig. 9. Recognition of several obstacles.

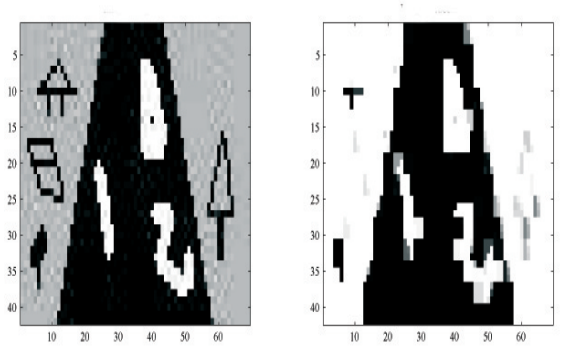

Fig. 10. Recognition of several obstacles.