APPLICATIONS OF CELLULAR NEURAL NETWORKS TO IMAGE UNDERSTANDING

Mariofanna Milanova
Paulo E. M. Almeida
Marcelo Godoy Simões

ABSTRACT

The Cellular Neural Networks (CNN) model is now a paradigm of cellular analog programmable multidimensional processor array with distributed local logic and memory. CNNs consist of many parallel analogue processors computing in real time. One desirable feature is that these processors arranged in a two dimensional grid, only have local connections, which lead themselves easily to VLSI implementations. The connections between these processors are determined by a cloning template, which describes the strength of nearest-neighbour interconnections. The cloning templates are space-invariant, meaning that all the processors have the same relative connections.

In this paper first we describe the architecture of CNN. Next, a new application of CNN using them for the 3D scene analysis is studied.

KEYWORDS

Cellular neural networks, associative memory, object recognition, image sequences, optical flow.

1 INTRODUCTION

The contrast between artificial and natural vision systems is due to the inherent parallelism and continuous time and signal values of the latter. In particular, the cells of the natural retina combine photo transduction and collective parallel processing for the realization of low-level image processing operations (feature extraction, motion analysis, etc.) concurrently with the acquisition of the image. Having collected of spatio-temporal information from the imagery, there exist spatial representations of this information that allow us to extract parameters necessary for 3D object recognition. The Cellular Neural Network paradigm is considered as a unifying model for spatio-temporal properties of the visual system [1],[2].

This paper is organised as follows. In Section II we review briefly the architecture of CNN. In Section III the applications of CNN for low-level image processing is presented. In Section IV design approach of CNNs for associative memories is implemented. Designing in an CNN can be done in a systematic manner by a synthesis procedure which stores all desired memory patterns as reachable memory vectors. In Section V we show a new application of CNNs, when using them for the recognition of optical flow field characteristics. Experimental results are presented in Section VI.

2 ARCHITECTURE OF CELLULAR NEURAL NETWORKS

Cellular Neural Networks (CNN) and the CNN universal machine (CNN –UM) were invented in 1988 and 1992, respectively [1]-[3]. The most general definition of such networks is that they are arrays of identical dynamical systems, the cells, that are only locally connected. In the original Chua and Yang model each cell is a one-dimensional dynamical system. It is the basic unit of a CNN. Any cell is connected only to its neighbour cells, i.e. adjacent cells interact directly with each other. Cells not in the immediate neighbourhood have indirect effect because of the propagation effects of the dynamics of the
network. The cell located in the position \((i,j)\) of a two-dimensional \(M \times N\) array is denoted by \(C_{ij}\), and its \(r\)-neighbourhood \(N'_r\) is defined by

\[
N'_r = \{ C_{ij} | \max\{|k-i|,|l-j|\} \leq r, 1 \leq k \leq M, 1 \leq l \leq N \}
\]  

(1)

where the size of the neighbourhood \(r\) is a positive integer number.

Each cell has a state \(x\), a constant external input \(u\), and output \(y\). The equivalent block diagram of a continuous time cell is shown in Figure 1. The first-order non-linear differential equation defining the dynamics of a cellular neural network can be written as follows:

\[
\frac{\partial x_{ij}(t)}{\partial t} = -\frac{1}{R} x_{ij}(t) + \sum_{C_{kl} \in N'_r} A(i,j;k,l) y_{kl}(t) - \sum_{C_{kl} \in N'_r} B(i,j;k,l) u_{kl} + I
\]

\[
y_{ij}(t) = \frac{1}{2} \left( |x_{ij}(t) + 1| - |x_{ij}(t) - 1| \right)
\]  

(2)

where \(x_0\) is the state of cell \(C_{ij}\), \(x_0(0)\) is the initial condition of the cell, \(C\) and \(R\) conform the integration time constant of the system, and \(I\) is an independent bias constant.

From [2] \(y_{ij}(t) = f(x_{ij}(t))\), where \(f\) can be any convenient non-linear function.

![Figure 1: Block diagram of one cell.](image)

The matrices \(A(.)\) and \(B(.)\) are known as cloning templates. \(A(.)\) acts on the output of neighbouring cells and is referred to as the feedback operator. \(B(.)\) in turn affects the input control and is referred to as the control operator. Of cause, \(A(.)\) and \(B(.)\) are application dependent. A constant bias \(I\) and the cloning templates determine the transient behaviour of the cellular non-linear network. (In general, the cloning templates do not have to be space invariant, they can be, but it is not a necessity). A significant feature of CNN is that it has two independent input capabilities: the generic input and the initial state of the cells. Normally they are bounded by:

\[
-u_{ij}(t) \leq 1 \text{ and } x_{ij}(0) \leq 1
\]

Similarly, if \(|f(\cdot)| \leq 1\) then \(y_{ij}(t) \leq 1\).

When used as an array processing device, the CNN performs a mapping

\[
x_{ij}(0) \quad \rightarrow \quad y_{ij}(t)
\]

\[
u_{ij}(t) \quad \rightarrow \quad y_{ij}(t)
\]

where \(F\) is a function of the cloning template \((A, B, I)\).

The functionality of the CNN array can be controlled by the cloning template \((A, B, I)\), where in 2D cellular neural network \(A\) and \(B\) are \((2r+1) \times (2r+1)\) real matrices and \(I\) is a scalar number. In many applications \(A(i,j;k,l)\) and \(B(i,j;k,l)\) are space invariant. If \(A(i,j;k,l) = A(k,l,i,j)\), then the CNN is called symmetrical or reciprocal.

There are two main cases: continuous-time (CT-CNN) and discrete-time (DT-CNN) cellular neural networks. The equations for each cell of a DT-CNN are

\[
x_{ij}(0) = \sum_{C_{kl} \in N'_r} A(i,j;k,l) y_{kl}(k) + \sum_{C_{kl} \in N'_r} B(i,j;k,l) u_{kl}(k) + I
\]

\[
y_{ij}(t) = f(x_{ij}(t-k))
\]

\[
f(x) = \text{sat}(x)
\]

(3)

A special class of two-dimensional cellular neural networks is described by ordinary differential equations of the form (see 2, [5]).

\[
\frac{\partial x_{ij}(t)}{\partial t} = -a_{ij} x_{ij}(t) + \sum_{k,l} f_{ij,k,l} \text{sat}(x_{kl}(t)) + I
\]

\[
y_{ij}(t) = \text{sat}(x_{ij}(t))
\]  

(4)

where \(1 \leq i \leq M, 1 \leq j \leq N, a_{ij} = 1/Rc > 0, \) and \(x_0\) and \(y_0\) are the states and the outputs of the network, respectively.

\[
\text{sat}(x) = \begin{cases} 
1 & x > 1 \\
0 & -1 \leq x < 1 \\
-1 & x \leq -1
\end{cases}
\]

and \(\text{sat}(\cdot)\) represents the activation function.

We consider zero inputs \((u_{ij} = 0\) for all \(i\) and \(j\)) and a constant bias vector \(I = [I_{11}, I_{12}, \ldots, I_{MN}]^T\).

Under these circumstances, we will refer to (4) as a zero-input non-symmetric cellular neural network where the \(n\) neurons are arranged in a \(M \times N\) array (if \(n = M \times N\) and the interconnection structure is confined to local neighborhoods of radius \(r\).

System (4) is a variant of the analog Hopfield model with activation function \(\text{sat}(\cdot)\), perhaps the best known of the associative neural network memories. The Hopfield networks, in general, are completely connected. Therefore, the number of connections scales as the square of the number of units. This presents a serious problem in the VLSI implementation of large networks. This limitation has been overcome by adopting both CNN models -
3 Cellular Neural Network for Low-Level Image Processing

The most popular application for CNN has been in image processing, especially because of their analog feature and sparse connections, which are conductive to real-time processing [1], [8], [9].

A two-dimensional CNN can be viewed as a parallel non-linear two-dimensional filter and have already been applied for noise removal, shape extraction, edge detection, etc.

Let us first approximate the integral equation (3) by a difference equation. Let \( t = nh \), where \( h \) is a constant time step, and approximate the derivative of \( x_i(n) \) by its corresponding difference form [1]

\[
\frac{C}{h} [x_i(n+1)h] = x_i(nh) + \sum_{C(i,j,k,l) \neq i,j} \lambda_{i,j,k,l} x_{ij}(nh) + 1 \leq i \leq M; 1 \leq j \leq N
\]

(5a)

\[
y_i(nh) = 0.5 (x_i(nh) + 1) = f(x_i(nh))
\]

(5b)

Let

\[
I_i = \sum_{C(i,j,k,l) \neq i,j} \beta_{i,j,k,l} x_{ij}(nh) + 1 \leq i \leq M; 1 \leq j \leq N
\]

(5c)

We can recast (5a) and (5b) into the form

\[
x_i(n+1) = x_i(n) + \frac{1}{C} R x_i(n) + \sum_{C(i,j,k,l)} \lambda_{i,j,k,l} f(x_i(n)) + I_i
\]

(6)

Equation (6) can be interpreted as a two-dimensional filter for transforming an image, represented by \( x(n) \), into another one, represented by \( x(n+1) \). The filter is non-linear because \( f(x_i(n)) \) in (6) is a non-linear function. Usually, the filter is space invariant for image processing. The property of the filter is determined by the parameters in (6). To find a set of parameters (coefficients, synaptic weights) so that a network performs according to a given task is one of the problems in CNN. For the one-step filter in (6), the pixel values, \( x_i(n+1) \) of the image are determined directly from the pixel values, \( x_i(n) \) in the corresponding neighborhood \( N_r(i,j) \) (3 x 3).

Therefore, a one-step filter can only make use of the local properties of images. When the global properties of an image is important the above one-step filter can be iterated \( n \) times to extract additional global information from the image. Well know property of an iterative filter is the so-called propagation property. This property can be observed by substituting \( x_i(n) \) in (6) iteratively down to \( x_i(0) \), which coincides with the input image.

\[
x_i(n) = \sum_{C(i,j,k,l)} \beta_{i,j,k,l} x_i(0) + 1 \leq i \leq M; 1 \leq j \leq N
\]

(7)

Therefore, the propagation property of iterative filters makes it possible to extract some global features in images. The image at time \( t \) depends on the initial image \( x_i(0) \) and the dynamic rules of the cellular neural network. Therefore, we can use a cellular neural network to obtain a dynamic transform of an initial image at any time \( t \).

The template coefficient (weights) of a CNN which will give a desired performance can either be found by design, learning or mapping. In this paper, one application of global learning approach and one application of design approach are presented. We investigated various visual task using global learning approach for CNN. All variants of global learning algorithms are based on the idea that an cost function is defined which measures how well the network maps a set of input images onto the desired output image in section IV we also show design approach for CNN associative memory. Design with CNNs [10] can have many different faces, e.g. programming the network to have some desired fixed points or to evolve along a prescribed trajectory from a given initial condition to some desired fixed point under control of a given input or specifying some desired local dynamics.

Many visual tasks are related to visual reconstruction and can be cast as optimization problems. Examples are shape from shading, edge detection, motion analysis, structure from motion, and surface interpolation. These ill-posed, inverse problem yield a solution through minimization techniques. From Poggio, 1985 [11] (Table I), we see that various early and intermediate level computer vision tasks are obtained by energy minimization of various functionals. As shown by Koch [12], quadratic variational problems can be solved by linear, analog electrical, or chemical networks using regularization techniques, additive models, Markov random field (MRF). However, quadratic variational principles have limitations. The main problem is the degree of smoothness required for the unknown function that is to be recovered. For instance, the surface interpolation scheme outlined above smooths over edges and thus fails to detect discontinuities.

Hopfield and Tank have shown that networks of nonlinear analog "neurons" can be effective in computing the solution of optimization problems. (traveling salesman problem TSP, stereo matching problem [13], [14]. As shown by Bose and Liang [15], CNN is an analog Hopfield network in which the connections are limited to units in local
### Table 1: Problems and the Corresponding Functionals

<table>
<thead>
<tr>
<th>Problem</th>
<th>Regularization principle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge detection</td>
<td>[ \int [(Sf-i)^2 + \lambda(f_x)^2] , dx ]</td>
</tr>
<tr>
<td>Area based Optical flow</td>
<td>[ \int [(u+iv+i)^2 + \lambda(u_x^2 + v_y^2 + v_x^2)] , dxdy ]</td>
</tr>
<tr>
<td>Contour based Optical flow</td>
<td>[ \int [(v-N-V^n)^2 + \lambda (dV/dx)] ]</td>
</tr>
<tr>
<td>Surface reconstruction</td>
<td>[ \int [(S.f-d^2 + \lambda (f_x^2 + 2f_y^2 + f_y^2)] , dxdy ]</td>
</tr>
<tr>
<td>Spatio-temporal approximation</td>
<td>[ \int [(S.f-i)^2 + \lambda(\nabla f \cdot V + \nabla f)^2] , dxdydt</td>
</tr>
<tr>
<td>Color</td>
<td>[</td>
</tr>
<tr>
<td>Shape from shading</td>
<td>[ \int [(E-R(f,g))^2 + \lambda(f_x^2 + f_y^2 + g_x^2 + g_y^2)] , dxdy ]</td>
</tr>
<tr>
<td>Stereo</td>
<td>[ \int \left{ \left[ \nabla G^m(U(x,y)-R(x+d(x,y),y)) \right]^2 + \lambda(\nabla f)^2 \right} , dxdy ]</td>
</tr>
<tr>
<td>Contours</td>
<td>[ \int E_{snake} \left( v(s) \right) , ds ]</td>
</tr>
</tbody>
</table>

...neighborhood of individual units with bi-directional signal paths...

Hopfield’s idea was to solve combinatorial optimization problems by allowing the binary variable to vary continuously between 0 and 1 and to introduce terms in the energy function that forced the final solution to one of the corners of the hypercube [0,1]^N. Briefly, let the output variable \( y_i(t) \) for neuron \( i \) have the range \( 0 < y_i(t) < 1 \) and be a continuous and monotonic increasing function of the internal state variable \( x_i(t) \) of the neuron \( i \): \( y_i = f(x_i) \). The output is then given as (a sigmoid-like function):

\[
y_i = \frac{1}{2} \left( 1 + \tanh \frac{x_i}{x_0} \right) = \frac{1}{1 + e^{-2x_i/x_0}}
\]

where \( x_0 \) determines the steepness of the gain function.

The dynamics of the CNN network are described by a system of nonlinear ordinary differential equations (4) and by an associated computation energy function (called the Lyapunov Function) which is minimized during the computation process.

The resulting changing equation that determines the rate of change \( x_y \) is

\[
C_y \frac{dx_y}{dt} = -\frac{x_y(t)}{R_y} + \sum T_{y,i}y_i(t) + I_y
\]

where \( y_i(t) = f(x_i(t)) \).

We replace the sign-type non-linearity in (4) by a sigmoidal non-linearity. In this case the system becomes a continuously valued dynamical system where gradients are well defined and classical optimization algorithms can be applied.

The Lyapunov function, \( E(t) \), of the cellular neural network is

\[
E(t) = -\frac{1}{2} \sum_{(i,j)} T_{y,i}y_i(t)y_j(t) + \frac{1}{2R_x} \sum_{(i,j)} y_i(t)^2 - \sum_{(i,j)} f_y^2(t)
\]

By using an appropriately defined energy function, stability of the CNN can be proved in the same way as an analog or continuous Hopfield network. Hopfield and Tank [16] investigated the analogy between finding a solution to a given optimization problem and setting an appropriate Lyapunov function corresponding to the additive neural model.
The analog network has two advantages. First the function E is Lyapunov, while for a digital network it is not. Second, deeper minima of energy have generally larger basins of attraction. A randomly selected starting state has a higher probability of falling inside the basins of attraction of a deeper minimum.

4 DESIGN OF CNNS FOR ASSOCIATIVE MEMORIES

The goal of associative memories is to store a set of desired patterns as stable memories such that a stored pattern can be retrieved when the input pattern contains sufficient information about that stored pattern. In a first phase some number p of patterns $\Xi^p$, $1 < p < p$, is stored during a learning process. In a second phase, the recall step, one of these p patterns (possibly corrupted by noise) is presented to the associative memory. Its output must then converge towards this pattern.

An associative memory can be implemented as a continuous-time or discrete-time dynamical system

$$\frac{dx}{dt} = F_\circ(x(t), u) \quad \text{or} \quad x(n+1) = F_\circ(x(n), u)$$

(10)

parameterised by a matrix $W$ encoding the patterns $\Xi^p$. The input $u$ can be presented as an initial condition $x(0)$ and as an independent input.

The most popular way to store patterns $\Xi^p$ is to store them as stable equilibrium points of the N-dimensional dynamical system (10).

A first attempt at developing a design method for associative memories using DTCNNs was made in [17], where the well-known Hebbian rule was used to determine the connection weights. However, serious limitations were found relating to the kind of patterns to be stored. The Hebbian training signal will not typically be optimal for learning-invariant object recognition due to erroneous classifications made by the neuron to spatially similar images from different objects and spatially dissimilar images derived from the same object.

We use the synthesis procedure presented by Liu [5] for the design of a cloning template for CNN. He considers a class of two-dimensional discrete-time cellular neural networks described by equations of the form

$$\frac{dx_{ij}}{dt} = -Ax_{ij} + T\text{sat}(x_{ij}) + I_{ij}$$

(11)

$$y_{ij} = \text{sat}(x_{ij}) \quad \text{with} \quad 1 \leq i \leq m; \quad 1 \leq j \leq n.$$

$x_{ij}$ and $y_{ij}$ are the states and outputs of the network respectively, and:

$$A = \text{diag}[a_1, ..., a_n]$$

$$T = [T_{ij}]$$ represents the feedback cloning template

$$I = [I_{ij}]$$ is the bias vector and

$$\text{sat}(x_{ij}) = \begin{cases} 1 & x_{ij} \geq 1 \\ 0 & -1 < x_{ij} < 1 \\ -1 & x_{ij} \leq -1 \end{cases}$$

$$\text{sat}(x_{ij}) = [\text{sat}(x_{11}), ..., \text{sat}(x_{mn})]^T$$ with

$\text{sat}()$ represents the activation function.

Among the synthesis techniques of CNN's for associative memories, the eigenstructure method appears to be especially effective. This method has successfully been applied to the synthesis of neural networks defined on hypercubes, the Hopfield model and iterative algorithms. The key idea is to make a proper choice of the interconnection matrix $T$. Next, we present the synthesis problem and the synthesis procedure.

Suppose that $\beta$ is an asymptotically stable equilibrium point and $\alpha \equiv \text{sat}(\beta)$ is a memory vector of system (11) with parameters $A, T,$ and $I$ The synthesis problem is as follows: Given $m$ vectors in $B^s$ (the desired memory patterns), say $\alpha_1, ..., \alpha_n$. How can we properly choose $A, T,$ and $I$ so that the resulting synthesised system (11) has the properties of an associative memory [5].

4.1 Synthesis Procedure

We choose vectors $\beta_j$ for $i = 1, ..., m$ and a diagonal matrix $A$ with positive diagonal elements, such that $A\beta = \mu \alpha_j$, where $\mu > 0$, i.e., choose $\beta_j = [\beta_j^1, ..., \beta_j^m]^T$ with $\beta_j^i/\alpha_j > 1$; $i = 1, ..., m$ and $f = 1, ..., n$. $A = \text{diag}[^\alpha_1, ..., \alpha_n]$ with $\alpha_j > 0$ for $j = 1, ..., n$ and $\mu > \max |\alpha_j|$ such that $\beta_j = \mu \alpha_j$. We use $A = \text{diag}[^1, ..., ^1]$ and $\mu = 10$.

Compute the $n \times (m-1)$ matrix

$$Y = [y_1, ..., y_m]^T = [\alpha^1-\alpha^n, ..., \alpha^{m-1}-\alpha^n]$$

(12)

Perform a singular value decomposition of $Y = USV^T$, where $U$ and $V$ are unitary matrices and $S$ is a diagonal matrix with the singular values of $Y$ on its diagonal.

Compute $T^+ = [T_{ij}] = \Sigma_i u^i (u^i)^T$

(13)

where $i = 1, ..., p$ and $p = \text{rank}(Y)$

$$T^* = [T_{ij}] = \Sigma_i u^i (u^i)^T$$

(14)

where $i = p + 1, ..., n$
Choose a positive value for the parameter $\tau$ and compute
\[ T = \mu T^+ - \tau T^- \]  
\[ I = \mu \alpha^+ - T \alpha^- \]  
\[ (15) \]

We use $\tau = 0.95$.

Then $\alpha^1, ..., \alpha^n$ will be stored as memory vectors in the system (4). The states $\beta^i$ corresponding to $\alpha^i$, $i = 1, ..., m$, will be asymptotically stable equilibrium points of system (11). There are several advantages using the eigenstructure method:

As is well known, the outer design method does not guarantee that every desired memory pattern will be stored as an equilibrium point (memory point) of the synthesized system when the desired patterns are not mutually orthogonal. The network designed by the eigenstructure method is capable of storing equilibrium points, which by far may outnumber the order of the network. (For example of a neural network of dimension $n=81$ which stores 151 vector as equilibrium points, refer to [6].)

In a network designed by the eigenstructure method, all $\alpha^i$ the desired patterns are guaranteed to be stored as asymptotically stable equilibrium points. System (11) is a variant of the recurrent Hopfield model with activation function $\text{sat}(\cdot)$. There are also several differences from the Hopfield model: 1) The Hopfield model requires that $T$ is symmetric. We do not make this assumption for $T$. 2) The Hopfield model is allowed to operate asynchronously, but the present model is required to operate in a synchronous mode. 3) In a Hopfield network used as an associative memory the weights are computed by a Hebb rule correlating the prototype vectors to be stored, while the connections in the Cellular Neural Network are only local. For example, a CNN of the form (11) with $M=N=9$ and $r=3$, has 2601 total interconnections, while a fully connected NN with $n=81$ will have a total of 6561 interconnections.

5 CELLULAR NEURAL NETWORKS APPLICATION IN 3D OBJECT RECOGNITION SYSTEM

In this section we are now concentrating on an application of CNN associative memory for 3D object recognition, that is based on the following idea. While the robot is moving from one viewpoint to another to gather characteristic views of an object, an image sequence is taken and analysed on the way. Obviously, such a sequence contains a lot of additional information. It implicitly codes the 3D structure of the object for the price of a huge amount of data. For the reduction of data and of processing time, we downsample the images in the sequence to a size of 32x32 pixels. The processing of the image sequence consists mainly of three steps:

1. computing the optical flow;
2. extracting features from the flow;
3. classification of these features.

5.1 Selecting Feature Vectors for Recognition

In any pattern recognition problem feature detection is of utmost importance. The features extracted from different views must have a reasonable degree of invariance against shift, rotation, scale and other variations. When an object cannot be identified on the basis of information about shape, other types of information will play a critical role—motion, spatial properties (size, location), texture [18], [19],[20],[21], [22], [25].

Our work in the spirit of Little and Boyd 23, 24, is a model-free approach making no attempt to recover a structural model of the 3D object. Therefore, it describes the shape of the motion of the object with a set of features. We derive features from dense optical flow data $(u(x,y), v(x,y))$. In contrast to Little and Boyd we aim on image sequences of a static complex object, taken by a moving camera. We determine a range of scale-independent scalar features of each flow image that characterise the spatial distribution of the flow. The features are invariant to scale and do not require identification of reference points on the moving camera. The flow diagram of the system that creates our motion features is presented in Figure 2.

The steps in the system are, from top to bottom:

1. The system begins with a motion sequence of $n+1$ images (frames) of an object.
2. The optical flow algorithm is sensitive to brightness changes caused by reflections, shadows, and changes of illumination, so we first filter the images by a Laplacian of Gaussian to reduce the additive effects.

3. We compute the optical flow of the motion sequence to get \( n \) images \((u,v)\) data, where \( u \) is the \( x \)-direction flow and \( v \) is the \( y \)-direction flow. We use the method presented by Bülthoff, Little, and Poggio [23]. The dense optical flow is generated by minimizing the sum of absolute differences between image patches. We compute the flow only in a box surrounding the object. The result is a set of moving points. Let \( T(u,v) \) be defined as:

\[
T(u,v) = \begin{cases} 
1, & \text{if } |(u,v)| \geq 1 \\
0, & \text{otherwise}
\end{cases}
\]

\( T(u,v) \) segments moving pixels from non-moving pixels.

4. For each frame of the flow, we compute a set of scalars that characterizes the shape of the flow in that frame. We use all the points in the flow and analyze their spatial distribution. The shape of motion is the distribution of flow, characterized by several sets of measures of the flow. Similar to Little and Boyd, we compute the following scalars:

\[
\begin{align*}
\text{centx} & : \text{x coordinate of centroid of moving region} \\
\text{centy} & : \text{y coordinate of centroid of moving region} \\
\text{wcentx} & : \text{x coordinate of centroid of moving region weighted by } |u| \\
\text{wcenty} & : \text{y coordinate of centroid of moving region weighted by } |u| \\
\text{waspct} & : \text{aspect ratio of moving region} \\
\text{dcentx} & = \text{wcentx} - \text{centx} \\
\text{dcenty} & = \text{wcenty} - \text{centy} \\
\text{aspect} & : \text{aspect ratio of moving region} \\
\text{waspect} & : \text{aspect ratio of moving region weighted by } |(u,v)| \\
\text{daspect} & = \text{aspect}-\text{waspect}
\end{align*}
\]

5. Each image \( I_j \) in a sequence of \( n \) images generates \( m=13 \) scalar values, \( s_{ij} \), where \( i \) varies from 1 to \( m \), and \( j \) from 1 to \( n \). We rearrange the scalars to form one time series for each scalar - \( S_i=[s_{i1},...,s_{in}] \).

These time series of scalars are then used as feature vectors for the classification step, which is implemented by using a Cellular Neural Network. Figures 3-5 show some images of a sequence and some flow images. Figure 6 visualizes the feature vector of the whole sequence.
6 EXPERIMENTS

A series of experiments was done using a set of images of the Columbia image database. This database consists of image sequences of objects, which were placed on a turntable when taking images every 5°. The background is uniform and the objects are centered. To speed up the flow computation and to handle the amount of data, we reduced the image resolution to 32x32 pixels and used only every second image. Thus, our image sequences consist of 36 low resolution images taken every 10°. The features of the image sequences of ten different objects of the database were used. Fig. 7 shows the learned objects. Fig. 8 shows an image sequence of object number two.

Figure 7: Images of the ten objects

Figure 8: Image sequence of object number two

Figure 9: Features of five test objects shown as gray value images
The resulting feature vectors for the time series of the feature vectors of five objects are shown in figure 9. For a better visualization they are shown as normalized gray images. We show the different features as described in sect. V in x-direction, the time in y-direction. One can see that they form characteristic patterns that are used for an unambiguous recognition of these objects. The associative memory is used to restore incomplete sequences and to classify them. Obviously, recognition of a previously learnt object would not be a problem at all, since the CNN is designed such that each stored feature vector $x_i$ is an equilibrium point of the CNN.

In our experiments we wanted to measure the influence of two important parameters: what happens if the object is not correctly centered in the images and how sensitive is the system to noisy input images?

Thus, we modified the image sequences by translating the objects by $t$ pixels in x- and y-direction each and by adding a random uniformly distributed noise of maximal value $n$ to each of the images. In our test $t$ varies from 0 to 6 and $n$ varies from 0 to 30. Consider, that the images are only of size $32 \times 32$ pixels. Thus a translation of the object of up to 20% of the image’s size is possible. For each value of the parameters $(t, n)$ the experiments were repeated four times – the noise was of random nature. Table 2 shows the achieved recognition rates for our test objects.

**Table 2: Recognition rates for varying translation and noise**

<table>
<thead>
<tr>
<th>Translation $t$ (pixels)</th>
<th>noise $n$</th>
<th>recognition rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>100,00</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>97,50</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>92,50</td>
</tr>
<tr>
<td>0</td>
<td>15</td>
<td>87,50</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>65,00</td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>60,00</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>62,50</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>100,00</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>100,00</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>92,50</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>87,50</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>67,50</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>87,50</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>65,00</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>100,00</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>97,50</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>87,50</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>82,50</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>72,50</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>65,00</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>65,00</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>80,00</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>97,50</td>
</tr>
</tbody>
</table>

In addition, Fig. 10 shows the average recognition rates for $t < t_i$ and $n < n_i$.

In addition, we compared the CNN associative memory to a nearest neighbor classifier. The Euclidean distance was used as the distance measure between the feature vectors. The main disadvantage of the nearest neighbor classifier compared with an associative memory is that it relies on a distance measure that holds no information on sub-patterns and the kind of distance between two different feature vectors.

Thus, the recognition rates of the nearest neighbor classifier are below those of the CNN associative memory. Fig. 11 shows the achieved recognition rates for the nearest neighbor classifier. Fig. 12 compares the recognition rates for a fixed value of $t$ and variable noise $n$.

**Figure 10: Average recognition rates of the CNN**

**Figure 11: Recognition rates of the nearest neighbor classifier**
Figure 12: Recognition rates of CNN and NN for constant translation \( r \) and variable noise \( n \)

Further experiments will be done concerning different resolutions of the images and concerning variable lengths of the image sequences.

7 CONCLUSION

We have presented applications of CNNs, in image processing and 3D object recognition. The systematic steps towards design and learning with CNNs provide powerful techniques for finding the template coefficients (synaptic weights) to perform a desired task. The nearest neighbour interactive property of CNN makes them much more amenable to VLSI implementation.

REFERENCES

25. M. Milanova, Recovering and representing three dimensional objects for computer vision or computer graphic applications. in: Proceedings of the DSP‘95, (Cyprus, 1998) 544-552.