

Texture Classification and Segmentation by Cellular Neural Networks Using Genetic Learning*

Tamás Szirányi and Márton Csapodi

*Analogical and Neural Computing Systems Research Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences
(MTA SZTAKI), P.O. Box 63, H-1518 Budapest, Hungary
E-mail: sziranyi@lutra.sztaki.hu*

Received November 14, 1994; accepted June 17, 1997

We present a new single-chip texture classifier based on the cellular neural network (CNN) architecture. Exploiting the dynamics of a locally interconnected 2D cell array of CNNs we have developed a theoretically new method for texture classification and segmentation. This technique differs from other convolution-based feature extraction methods since we utilize feedback convolution, and we use a genetic learning algorithm to determine the optimal kernel matrices of the network. The CNN operators we have found for texture recognition may combine different early vision effects. We show how the kernel matrices can be derived from the state equations of the network for convolution/deconvolution and nonlinear effects. The whole process includes histogram equalization of the textured images, filtering with the trained kernel matrices, and decision-making based on average gray-scale or texture energy of the filtered images. We present experimental results using digital CNN simulation with sensitivity analysis for noise, rotation, and scale. We also report a tested application performed on a programmable 22×20 CNN chip with optical inputs and an execution time of a few microseconds. We have found that this CNN chip with a simple 3×3 CNN kernel can reliably classify four textures. Using more templates for decision-making, we believe that more textures can be separated and adequate texture segmentation (<1% error) can be achieved. © 1998 Academic Press

Key Words: cellular neural network; smart sensors; genetic algorithm; texture analysis; deconvolution; supervised learning; unsupervised learning; segmentation.

1. INTRODUCTION

In this paper we present a new single-chip texture classifier system based on the cellular neural (also called cellular nonlinear) network (CNN) architecture [1, 2]. CNNs are 2D cell arrays with local cell interconnections which can be used as an effective tool in many image processing tasks [3]. Their main strength is that they can be implemented on parallel VLSI along with programmable cell interconnections and optical inputs [4, 5]. Using CNN for texture recognition is theoretically new because CNNs

perform continuous time feedforward and feedback convolutions and nonlinear dynamics. The kernel matrices, or templates, containing the interconnection weights are determined through genetic learning. Textured images are first equalized; then we expect that with the learned weights the network transient will settle into a stable state where different textures can be classified according to the gray-scale average or texture energy of the output.

In Section 2 we give a brief overview of how CNN is related to other texture classification methods and lay the foundation for the expectation that CNN can be used for this task. In Section 3 we motivate the use of genetic algorithms in CNN template optimization problems. We also present some test results on a few enhancements we made to improve genetic learning. Several types of natural textures are examined in Section 4 to test our method, using digital network simulations. Experimental results are shown for 4, 8, and 16 texture classes. Sensitivity analysis for noise, rotation, and scale is included. As some of the first programmable CNN chips are being tested in our laboratory, we report the results of on-chip experiments. The chip we use has optical inputs, but pixel values are black and white. Using the same learning algorithm, we are able to find templates for classification of natural textures projected onto the optical input array of the chip. The main results are summarized in Section 5. In the Conclusions we compare our results to standard feedforward convolution methods.

2. CELLULAR NEURAL NETWORKS AND THEIR USE IN TEXTURE RECOGNITION

Cellular neural networks were introduced by Chua and Yang in 1988 [1, 2]. CNNs are 2D cell arrays with local, shift-invariant cell interconnections and analog, nonlinear cell dynamics. CNNs can be implemented as VLSI chips with optical inputs and programmable cell interconnections, providing an effective tool for real-time image processing. A CNN's normalized differential state-equation can be described by matrix-convolution operators

$$\frac{dX_t}{dt} = -X_t + A * Y_t + B * U + J, \quad (1)$$

* A short part of this paper has been published in the *Proceedings of the 12th ICPR, Jerusalem, 1994*.

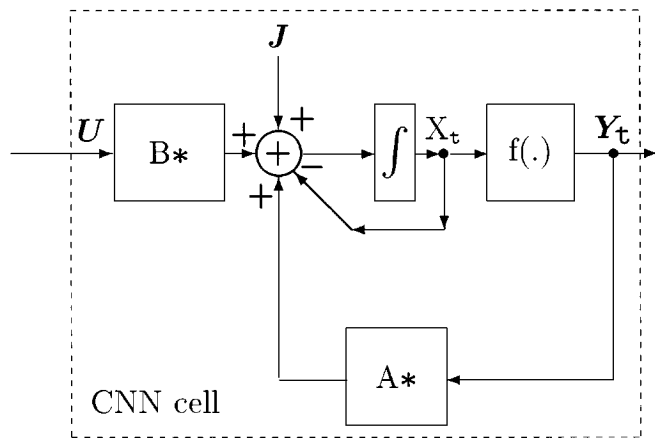


FIG. 1. Convolution-kernel scheme of image processing in the CNN structure.

where U , X , Y are the $M \times N$ input, state, and output matrices, respectively, while J is the bias (it is usually uniform, $J = J$). Here the boundary condition is U , X , $Y = 0$ if $(i, j) \notin [1; M] \times [1; N]$, and (i, j) indicates the position of a cell. There is a nonlinear function between the state and the output, $Y_{i,j} = f(X_{i,j})$, where $f(\cdot)$ is usually a sigmoid. The symbol $*$ denotes convolution. A and B represent the feedback and feedforward connections. The members of shift-invariant local processing units A , B , and J together form the CNN template [1, 2]. Typically, transients settle within a few μs in a VLSI CNN [4, 5].

Figure 1 shows the architecture of a CNN, which is based on convolutions performed by A and B .

The CNN Universal Machine (CNN-UM [4]) is a high-level parallel machine. It contains several analog template memories (“analog programs”) and image memories to save and compare images. It can execute a series of simple CNN templates and arithmetical and logical operations. Using the CNN-UM, we can perform multi-template experiments, and we can apply simple functions to the different outputs.

Texture segmentation methods can be grouped into four main classes [6]:

- (1) Statistical methods
- (2) Model-based methods
- (3) Structural methods
- (4) Methods using spatial-frequency information.

In the following, we give a short overview of methods which are closely related to CNNs. This will lay the foundation for our arguments that the system can be used for texture classification and segmentation.

1. Statistical methods are based on stochastic measures computed over some neighborhood of each pixel [7]. Among others, autocorrelation and operator-based methods fall into this class. By using an asymmetric feedback template, a CNN can perform pattern shifting and some forms of autocorrelations. Operator-based methods, such as convolution with Laws’ matrices [8], are easily implemented on a CNN, and—in addition—through

its feedback a CNN can ameliorate the drawbacks of other techniques which deal only with high spatial-frequency textures.

2. Model-based methods: A class of reaction–diffusion and anisotropic diffusion equations [9] fit the CNN structure if we allow two cooperating cell layers [3]. The Markov random field (MRF) model can be used for the segmentation of textured images [10], and we recently succeeded in implementing a special MRF model in the CNN architecture [11].

3. Structural methods generally assume that textures consist of definite repeating texture elements (e.g., periodic textures) [12]. Although this assumption is not valid for many textures, this method is interesting because the classification of such textures can be performed by deconvolutions which can be implemented on a CNN [3].

4. In Section 2.1 we prove that using linear output functions, $f(\cdot)$, a CNN template can represent spatial filters. The effect of such filters can be measured by the change in the frequency spectrum of the image or (by Parseval’s theorem [13]) by the change of the power spectrum, i.e., the “energy” of the image. The latter is the average of the squared intensity values (effective value of the output signal). When used in texture recognition this measure is called texture energy (TE) [8].

We propose the following procedures for CNN-based texture segmentation:

1. Histogram normalization of the textured image (if needed)
2. Filtering of the image with one or more suitable templates which are designed through the use of genetic learning
3. Classification of either the gray-scale average or the TE of the filtered output.

We note that the recognition of texture images which have identical mean brightness cannot be done by purely linear transformations. Thus, if we use CNN in its linear domain in step (2), then in step (3) we have to use a nonlinear measure for estimation, e.g., TE. If, however, we utilize nonlinear effects when filtering, and the network settles outside the linear slope of its output characteristics (we get mostly black and white output), then the ratio of black pixels (ROB) or the gray-scale average can be a measure for the classification in step (3). As can be seen, we have nonlinear transformation in both cases, just at different levels of the computation. TE (or higher order measures) result in finer resolution in a classification or segmentation step, but only if the filtered image (output of step (2)) is gray-scale. However, recent analog VLSI chips [27] perform well with half-toned (HT) binary outputs, which results in reliable classifications. In this case, filtering and half-toning effects can be combined in a single template, as we demonstrate in Section 2.2. In Section 4 we give examples of both the HT and the TE cases.

2.1. CNN as a Convolution/Deconvolution Filter in the Linear Domain

The CNN contains convolutions in both its feedforward and feedback operators. They can represent spatial filters when

the state-variable of each cell remains inside the linear domain.

In real-life environments the shift-invariant part of image degradations can be caused by feedforward convolutions (e.g., blur) and feedback convolutions (e.g., reflection spread). The general steady state model of a relatively complex natural image filter can be described as

$$\mathbf{W} = \mathbf{H} * \mathbf{V} + \mathbf{K} * \mathbf{W} + \mathbf{L}, \quad (2)$$

where \mathbf{V} , \mathbf{W} are the $M \times N$ input and output images, \mathbf{H} , \mathbf{K} are the feedforward and feedback convolution kernels, and \mathbf{L} is the bias. Since Eq. (2) is the canonical form of any linear shift-invariant filter, it covers imaging effects such as smoothing, reflections in a lens system, first-order optical errors, and volume distortion of microscopic targets.

Using the symbolic convolution-based formulation

$$\mathbf{K}^- * = 1 - \mathbf{K} *,$$

where \mathbf{K}^- is a convolution matrix which is the negative of \mathbf{K} with 1.0 added to the central element of $-\mathbf{K}$. \mathbf{V} and \mathbf{W} can be given via a convolution/deconvolution process, i.e.,

$$\mathbf{K}^- * \mathbf{W} = \mathbf{H} * \mathbf{V} + \mathbf{L}. \quad (3)$$

If the output of this linear filter is applied to the input of the CNN system in Fig. 1 ($\mathbf{W} = \mathbf{U}$), there is a template (\mathbf{A} , \mathbf{B}) which transforms the input into the original unfiltered image in the least squared error sense, i.e., $\mathbf{Y}(t = \infty) = \hat{\mathbf{V}}$. This can be proved using the well-known gradient iteration method [14].

If $\mathbf{R} = \mathbf{K}^- * \mathbf{U} - \mathbf{H} * \mathbf{Y} - \mathbf{L} = \mathbf{K}^- * \mathbf{W} - \mathbf{H} * \hat{\mathbf{V}} - \mathbf{L}$ is the residual error-matrix of the estimation, and $\|\mathbf{R}\|$ is its square norm, it can be shown that the gradient of this squared residual depending on the estimated elements of the \mathbf{Y} matrix is

$$\mathbf{G}_Y = \frac{d\|\mathbf{R}\|}{dY} = 2\mathbf{H}^c * (\mathbf{H} * \mathbf{Y} - \mathbf{K}^- * \mathbf{U} + \mathbf{L}). \quad (4)$$

Here the \mathbf{H}^c convolution-matrix comes from a 180° rotation of the convolution-matrix \mathbf{H} .

Remaining in the linear domain of the CNN, where $\mathbf{Y} \equiv \mathbf{X}$, this gradient can be used in an infinite iteration process in calculating \mathbf{X} (IIR convolution filter). This time-dependent approximation process is given by the next state-equation,

$$\frac{d\mathbf{X}}{dt} = -\eta \mathbf{G}_Y = -\eta \mathbf{H}^c * \mathbf{H} * \mathbf{X}(t) + \eta \mathbf{H}^c * \mathbf{K}^- * \mathbf{U} - \eta \mathbf{H}^c * \mathbf{L}, \quad (5)$$

where η is the step size. It follows from the stability theory of linear algebra [15] that $\mathbf{H}^c * \mathbf{H}$ in the feedback is always stable if the system block-matrix given by the convolution-matrix \mathbf{H} is nonsingular.

Putting Eq. (1) and Eq. (5) together, there is a one-to-one correspondence between the parameters:

$$\begin{aligned} \mathbf{A} * &= 1 - \eta \mathbf{H}^c * \mathbf{H} *, & \mathbf{B} * &= \eta \mathbf{H}^c * -\eta \mathbf{H}^c * \mathbf{K} *, \\ \mathbf{J} &= -\eta \mathbf{H}^c * \mathbf{L}. \end{aligned} \quad (6)$$

There is another solution for this inverse template:

$$\mathbf{A} * = 1 - \eta \mathbf{H} *, \quad \mathbf{B} * = \eta - \eta \mathbf{K} *, \quad \mathbf{J} = -\eta \mathbf{L}. \quad (7)$$

This is a good approximation of Eq. (6) for weak blurring effects; however, its linear stability is doubtful.

From Eqs. (6) and (7) it can be seen that the CNN can perform any transformation requiring simultaneous convolution/deconvolution. Filtering in the Fourier domain can be performed by designing the zeros (convolution) and the poles (deconvolution). The same task can be performed by the CNN templates. Since local filtering in the Fourier domain is an effective method for texture and edge detection [6], the present CNN-domain linear filtering method (emphasizing the analog feedback) can execute a similar task. Moreover, it does not need transformation and division by small values (as Fourier deconvolution does [16]), and it is parallel and very fast.

The effect of the above filters can be measured by the change of the spectral power in a given image-segment, which from Parseval's theorem equals the change in the average of the square of the output on the image segment. Measuring the power of the outgoing pixel value (simple effective value), this texture energy gives a good measure of the texture quality. However, we still have to address the problem, namely, several types of CNN VLSI chips provide only binary output [5]. In the following section it will be shown that by slightly modifying the central element in the kernel representing an arbitrary linear filter, we get a nonlinear filter which provides half-toned output. In this case the outgoing ratio of black to white pixels gives the measure of similarity.

2.2. Some Nonlinear Effects

Let us consider a convolution/deconvolution pair (\mathbf{C} , \mathbf{D}) in the CNN, for which the CNN is stable and remains in the linear domain:

$$\mathbf{A} * = 1 - \mathbf{D} *, \quad \mathbf{B} * = \mathbf{C} *, \quad \mathbf{J} = 0.$$

If we change the template by adding a small $\epsilon > 0$ to the self-feedback ($\mathbf{A} * = 1 + \epsilon - \mathbf{D} *$), the steady state is broken, and the system begins to behave as if it had only one nonzero feedback element, the self-feedback with the value of ϵ . This causes the output values (elements of \mathbf{Y}) to saturate in the nonlinear domain.

If \mathbf{A} satisfies the criteria of [1, Theorem 5] with self-feedback > 1.0 and symmetry of \mathbf{A} , the output is always binary and stable. However, there are several other cases when the CNN output is binary and (nearly) stable. If the filter convolutions have

smoothing effects, then the binary output is a half-toned version of the input. This is examined in detail in [17, 18].

In the case of an asymmetrical opposite-sign template, pattern shifting [19] can be combined with filtering and the result is stable in the nonlinear domain of the sigmoid. However, owing to the shifted image-content, there may be some cross-correlation effects between input and output.

This complex behavior of a single CNN template, combining filtering, half-toning, and pattern-shifting, is the crucial feature of our CNN texture analysis system. However, the parameters in the template are difficult to design since heuristics are difficult to apply. Furthermore, the processing of the combined functions is not easy to handle by analytical methods. It is important to note that the nonlinear behavior of the CNN can result in interesting effects that are difficult to foresee.

3. LEARNING CNN TEMPLATES WITH GENETIC ALGORITHMS FOR MULTIPATTERN TASKS

Owing to their novelty, applicable learning algorithms for the CNN structure do not have a broad literature. The interesting results of [20] initiated the use of genetic algorithms (GA) for CNN optimization, and we show here how GAs perform on this task.

There are three main reasons for using GAs for our purposes:

- In previous applications [21], the GA performs well if there are relatively few free parameters (here, template element values) and a large number of dependent variables (here, the output pixel values).
- There are no methods which allow for an analytical description of textures nor the incorporation of their recognition into the CNN structure.
- The complexity of the transient trajectories of a CNN makes it hard to trace how a slight change in the template values will influence the output. Thus, we cannot say explicitly how to modify template elements according to the difference of the desired and the computed output. However, the probabilistic operators used in GAs can overcome this difficulty.

Each template element is coded using a simple 4–8 bit binary code in the $[-5; 5]$ range that corresponds to the expected VLSI technological limits. Template design methods usually consider some specific template structure [e.g., 17, 19, 22]. Here we make no restrictions on templates generated during optimization. Stability conditions are stated for some classes of the CNN templates [19, 23], but using them here would unreasonably restrict the parameter space. In our case absolute stability (stable convergence) is not necessary. We need only a well-defined time-interval during which the output does not change, or changes very slightly, so that the output can be read after a predefined transient time. However, we have found in our experiments that our templates are usually stable in the long-term sense as well.

We described in Section 2 that, after normalizing the textured images, we classify them according to the gray-scale average

or texture energy of the output of a CNN transform. Thus, the GA fitness function should be formed in such a way that the gray-scale average or texture energy of the output generated by an optimum template will differentiate different textures but has only small variation inside a single texture. In the fitness function the variance of local average gray-levels $\hat{g}_{i,j}$ is related to the gray-level differences between the images of different texture-classes ($\hat{G}_k - \hat{G}_{k-1}$). Here \hat{G}_k indicates the average gray-scale of the k th texture in the output, numbered in ascending order of gray-level. We have found the following form to be useful:

$$\text{cost}(A, B, J) = \text{maximum}_k \left(\frac{1}{Q_k} \sum_{i,j|k} \text{Error}_k(i, j) \right) \quad (8)$$

$\text{Error}_k(i, j)$

$$= \begin{cases} 0, & \text{if } (\hat{G}_{k-1} + \hat{G}_k)/2 < \hat{g}_{i,j|k} < (\hat{G}_k + \hat{G}_{k+1})/2 \\ 1, & \text{else.} \end{cases} \quad (9)$$

Here Q_k is the area of the k th texture. Creating the next generation, $\text{fitness} = 1 - \text{cost}(A, B, J)$ is used as the measure of survival probability. To avoid results which do not differentiate the different textures, fitness is corrected by the minimum contrast; e.g.,

$$\text{fitness}(A, B, J) = (1 - \text{cost}(A, B, J)) \text{minimum}_k (\hat{G}_k - \hat{G}_{k-1}). \quad (10)$$

It should be noted that we use the same thresholding-based discrimination function (Eq. (9)) during both testing and training. In the case of up to 8 texture classes (see Section 4.2) the $\hat{g}_{i,j|k}$ values of the different textures have separable statistics. We have found that the perfect classification using the Bayesian decision [14] results in a small difference of the estimation error. The perfect Bayesian decision may require forming a table of additional statistical information, which can be done (if necessary) by using additional digital hardware connected to the CNN structure, as is shown for a recognition task in [24].

We have incorporated techniques to improve the speed of our algorithm. These are as follows:

- The first population of chromosomes is initialized partly from a template library. In this library we store templates found for other texture recognition tasks by our algorithm and standard CNN templates which realize typical image transformations, e.g., edge detection. This initialization makes it more likely that we will quickly arrive at suitable templates, but admittedly may bias the search.
- A single CNN template contains 19 elements when using 3×3 convolution kernels. The number of elements increases with kernel size quadratically. As mentioned earlier, GA performance decreases when searching in higher-dimension parameter spaces. For this reason, only a few (3–10) of the template elements are iterated at a time. The parameters are reselected after every 10 iterations according to their significance estimated from

previous runs. We have found that using more than about 8–10 parameters simultaneously in the optimization process leads to lack of convergence. While our restriction to searches over a limited number of elements may be suboptimal, we have empirically found that our results are quite satisfactory.

- If we apply a fixed, relatively high mutation rate (5–10%), the search process has a greater parameter space, but may not converge; on the other hand, application of a fixed, low mutation rate (0.2–1.0%) may cause low convergence speed and suboptimal results. During our experiments, we have found that periodically interchanging (say, with periods of 10 generations) a higher and a lower mutation rate results in faster convergence than a fixed, medium (2–3%) mutation rate.

- In every generation, we create three extra templates besides those created by standard GA operations. These are the inverse template, average template, and time-interpolated template. The inverse template is the linear inverse (see Eq. (7)) of the worst template in the current generation. The average template is the sum of all the templates in a population weighted by the normalized fitness for each template. Parameter values in the best template are multiplied by a convergence speed factor (see Eqs. (6) and (7)) to create the time-interpolated template. The speed factor is computed from the gradient of the cost function at the time we stop the network simulation. Although the considerations that led us to include these templates are based on linear effects, they appear to be very useful. The inverse template usually has low fitness, but has a similar effect to the mutation operator, by creating a new template far from its parent in the parameter space. The average template and the time-interpolated template are combinations of the better templates, and they quite often have superior fitness in their population.

Training speed is highly dependent on the task and the initial population. If the initial population contains templates performing similar tasks, a good solution can be found after not more than 10 generations (with a population of 20 to 50 templates). We have found that for more difficult recognition tasks our algorithm typically generates an acceptable solution within 100–200 generations. For cases in which there are no usable initial templates, the convergence time also depends on the parameter accuracy (number of bits) [18]. We observe that a satisfactory convergence speed can be achieved using a parameter accuracy of about 4–5 bits [18].

Testing a 128×128 image, 100 generations with population sizes of 100 were generated and fully tested within 4 h using a 100 MHz Pentium PC and a CNN simulator board [25] containing 4 TMS320C25 DSPs.

As described in Section 5, using a CNN VLSI chip for the same task, it takes only a few minutes to obtain satisfactory templates.

4. EXPERIMENTS

Since the CNN structures implemented in VLSI chips [5, 26, 27] process only a limited number of pixels (up to 32×32), more

extensive experiments can be done on a CNN digital hardware simulator system [25]. In these simulations we show the effects and limits of the method itself, without consideration of the VLSI consequences. CNN chip results are described in Section 5.

In our demonstrations four or eight different textures are trained together in the learning process. They are normalized to have the same average gray-level and similar equalized gray-level histograms to ensure the generality of the experimental test. Histogram equalization can be done by a 2-layer (2 memories/cell) CNN model [28]. Some of the CNN VLSI chips contain an automatic gain control [5, 27], resulting in normalized input images.

We used the natural Brodatz textures [29] in our experiments. (The serial numbers of textures are noted in brackets after the texture names.) Scanning resolution is between 100 and 25 dpi, depending on the periodicity of the given texture, since the detection-area of the CNN array (smoothing window at the output) should contain several periods, and the scanning resolution of the images should be sufficient for discrimination [30]. In our examples the template size is 3×3 or 5×5 , and the (quasi)period of the textures is between 5 and 30 pixels.

The main steps of texture *segmentation* are as follows:

1. Local filtering of input textures to get uniform average gray-levels and contrast.
2. CNN texture-filtering processing of the image using one or more templates.
3. Squaring the outgoing pixel values (in the case of texture energy output only).
4. Gray-scale averaging by smoothing convolutions or heat-diffusion [9].
5. Gray-scale thresholding as a final decision by a nonlinear output function.

Each process can be executed by a one-layer CNN-UM [4]. When the lighting conditions are stable or the CNN chip has a normalized optical input, then step 1 can be omitted. When the task is the *classification* of the texture class of an image, the two last steps are not necessary and the outgoing pixel values can be evaluated as the ratio of black pixels (ROB for black/white output, computed by wired summation on the chip) or as the effective value of the outgoing pixel voltage (squared output, TE). These measurements can be implemented in VLSI by simple wiring changes in the hardware architecture.

Figure 2 shows the above process when four Brodatz textures (Fig. 2a) are segmented using a 3×3 filter-template and squared output (Fig. 2b), smoothing and final thresholding (Fig. 2c).

Testing the classification, after the CNN process (and squaring the analog output in the case of TE) every output pixel is replaced by the smoothed average of the surrounding $W \times W$ detection window. The error of misclassification is computed from this smoothed image. The measure of classification error is the ratio of pixels with false classification.

Training and testing image sets are disjoint. Textures are usually represented by a $\approx 128 \times 128$ image in training and by

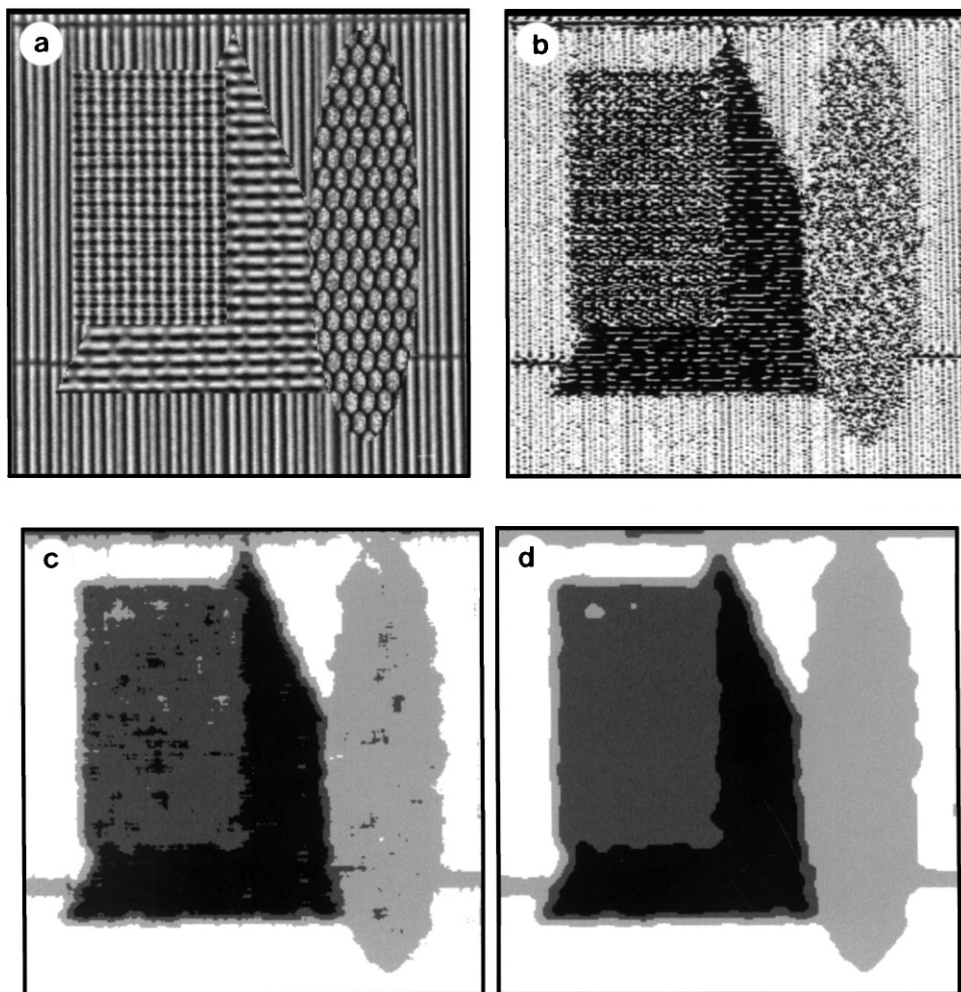


FIG. 2. Segmentation of four Brodatz textures with a single 3×3 CNN template. (a) Test image (256×256) contains netting (34), canvas (21), straws (49 and 53). Textures are equalized so that they have similar flat histograms. (b) Squared output after the CNN transformation. (c) Recolored final result after smoothing and thresholding. (d) Markov random field segmentation of (c) using a CNN MRF model [11].

$\approx 256 \times 256$ in test sets. When the width of the detection window is $W = 21$, there are 36 independent detection windows in training and 144 in the test, but many of the overlapping positions (about 10^4) are also considered. The detection window scans the output as the optical input array of a CNN chip can scan the surface at different positions. Using only one detection template, the segmentation error (border offset) can be estimated as the radius of the appropriate detection window, $W/2$.

Using the output of these $W \times W$ averaging windows for detection statistics is similar to the recognition from undersampled patterns [31]. We could get better classification results if, for a given pixel position, the gray-level histogram of the scanning decision window were considered instead of one average level. However, this makes the process more time-consuming, and calculation of statistics using a CNN chip in the learning process is not easily accomplished.

It is important to note that in our simulations the training and the test of classification or segmentation are executed on a mixture of different textures (multiple textures/one image), and not on separate texture images (one texture/one image). This makes our method independent of the size of the image and robust against the effect of interclass pattern stain.

4.1. Classification of Four Textures

We have found that a 3×3 CNN template may contain enough information to separate four different textures.

Figure 3 shows the classification test of herringbone weave (17), straw clothes (52 and 53), and lizard skin (36). The CNN transforms the input into a half-toned output. This texture set is not a trivial task because of the changing period and contrast, and we found that successful linear inverse templates were very important in the GA learning process.

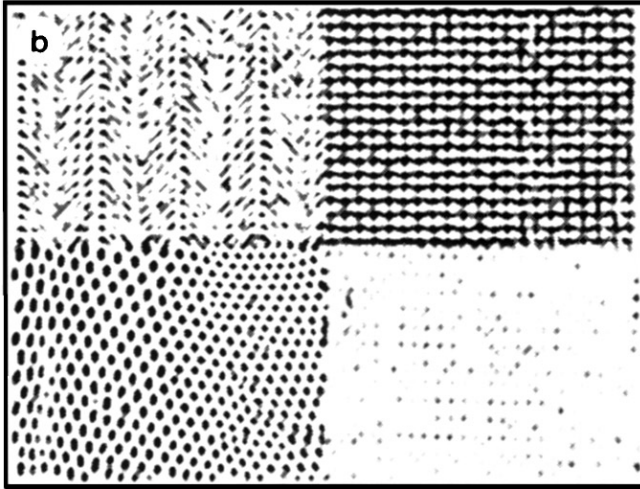
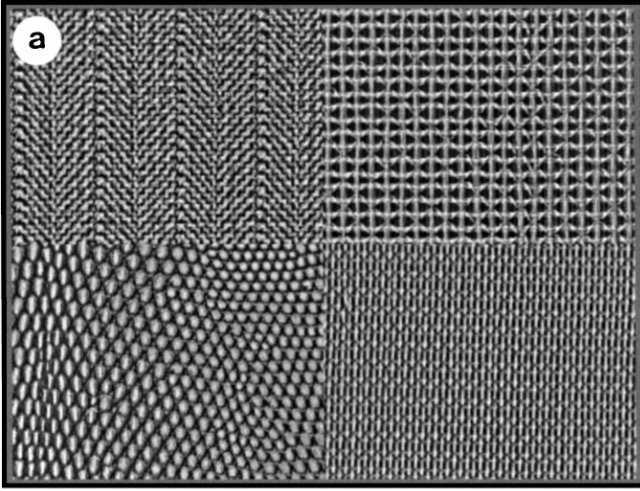


FIG. 3. CNN transformation of four Brodatz textures with a 3×3 template resulting in a halftoned-like output. (a) Input image (300×400) with the equalized herring (17), straw cloths (52 and 53), and lizard skin (36) textures. (b) Output of the CNN transformation. This output can be directly used for classification by scanning with a small detection-window.

The 3×3 template for Fig. 3 is

$$\mathbf{A} = \begin{pmatrix} 2.70 & 0.39 & 4.18 \\ 0.43 & -4.84 & 0.00 \\ 4.26 & -3.55 & 4.14 \end{pmatrix}, \quad J = -2.23,$$

$$\mathbf{B} = \begin{pmatrix} -5.00 & -2.81 & -2.85 \\ -1.21 & 3.71 & -3.40 \\ -5.00 & -3.83 & -5.00 \end{pmatrix}.$$

Considering texture energy (TE) at the output, the J offset value is usually zero. For the texture set in Fig. 2, the template

is as follows:

$$\mathbf{A} = \begin{pmatrix} -1.60 & -1.87 & 2.62 \\ 2.62 & -3.59 & 2.23 \\ -1.33 & -2.07 & 2.54 \end{pmatrix}, \quad J = 0.00,$$

$$\mathbf{B} = \begin{pmatrix} 1.17 & 1.76 & -4.96 \\ -2.46 & 3.83 & 1.17 \\ 3.55 & -1.41 & 0.86 \end{pmatrix}.$$

Table 1 contains the summarized numerical results of the above experiments, as well as several others. The table gives the classification error within the texture.

4.2. Classification of Eight Textures

When the number of possible texture classes is higher, larger texture-segments are needed for adequate classification. Since the CNN results in a halftone-like output with different average gray levels for each texture, this restricts the space of the filter parameters. Using the squared output (TE) for classification, this problem can be avoided, and we can achieve reliable classification results with good regional consistency for most of the classes.

Figure 4 shows the 200×400 training image containing eight textures. Using a 5×5 template and TE gives good segmentation. Although the misclassification error for the training set is lower than 1.0% using 25×25 detection-windows, misclassification error becomes 5.4% using the same detection-window size, and it is 0.3% with 51×51 detection-windows on the 512×1024 test image. Figure 5 shows the histograms of the detection windows scanning the output of the different textures. Contrast (divergence of the \hat{G}_k values) among the different output areas is 93% (100% means the possible maximum equal differences between the consecutive \hat{G}_k values). To achieve lower misclassification error with a smaller detection window for a relatively large number of textures we would need to use more templates, or texture-specific templates.

4.3. Segmentation of 1 of 16 Textures

It is much easier to train the parameters when only one texture is extracted from many others, with a goal of obtaining a satisfactory segmentation border. Usually, this can be done with 3×3 templates. Training for single-texture segmentation is similar to that for multitexture classification except that the extracted texture should become more black-colored than the remaining textures.

In the following cases training was done on 4 textures, while the test was done on a set of 16 randomly chosen textures. The members of the training set are chosen to enhance the characteristic features of the examined texture during training, which results in a template which is associated with the given texture class.

TABLE 1
Test Results for the Classification of Four Textures by CNN Depending on the Template Size, Output Type
(Half Toning/TextureEnergy), and Size of the Detection Window

Texture set (Brodatz No. [29])	Fig.	Template size	Output HT/TE	Classif. error	Window size	Aver. gray-level in %			
						\hat{G}_1	\hat{G}_2	\hat{G}_3	\hat{G}_4
Raffia (84), AluWire (14), French canvas (21), Straw cloth (53)	7	3×3 3×3 5×5	HT TE HT	1.4% 0.8% 0.6%	25 21 25	66 29 10	78 52 21	88 67 32	98 83 46
Herringbone (17), Cloth (52), LizardSkin (36), Cloth (53)	3	3×3 3×3	HT HT	0.0% 4.0%	31 21	2 2	17 17	28 28	44 44
StrawCloth (52), Canvas (77), Matting (55), Rattan (65)		3×3	TE	1.1%	35	37	53	65	77
Netting (34), Canvas (21), Straw (49), Straw (53)	2	3×3	TE	0.7%	21	20	45	64	85
Pellets (66), AluWire (6), AluWire (14), Herringbone (16)		3×3	TE	2.7%	35	53	58	61	82

Figure 6a shows 16 textures, and the half-toned results for the aluminum wire and straw cloth textures can be found in Figs. 6b and 6c (half-toned results are scaled by 2). The ROB results for detecting the 4 different textures are summarized in Table 2.

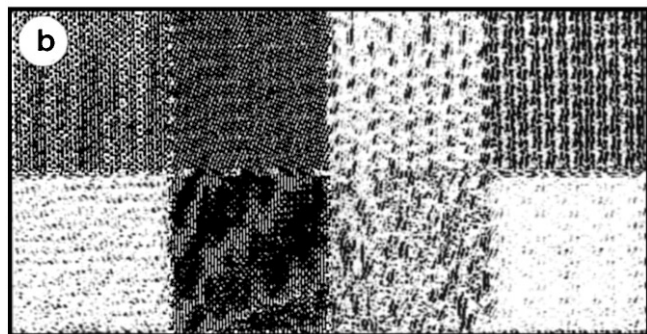
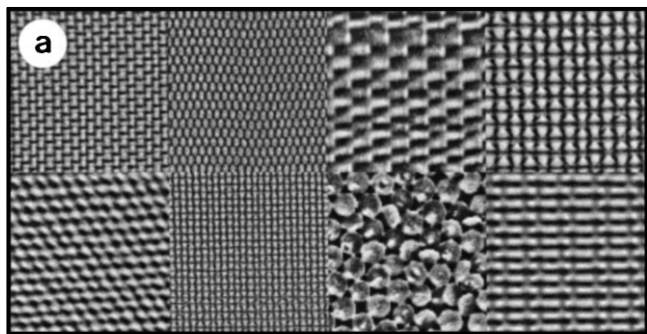


FIG. 4. Classification of eight Brodatz textures with a single 5×5 CNN template. (a) Input sample image (200×400) with the equalized aluminum wire (6), netting (34), straw matting (55), straw cloth (52 at 200 dpi), canvas (77), straw cloth (52 at 50 dpi), pellets (67), and straw cloth (53) textures. (b) Squared image after the CNN transformation.

4.4. Fine Multitemplate Segmentation

In the previous classification examples we have found that there may be a misclassified stripe between two textures when we want to segment several (e.g., four) textures. This results from the fact that between two average gray-levels of two neighboring textures there may be one or more other feature gray-levels of other textures. Figure 2c shows this result. We have chosen a small detection window ($W = 11$) to minimize the false stripe, but this results in a moderate misclassification error inside the texture.

We have developed a Markov random field image segmentation method for the CNN architecture, using simulated annealing and a modified Metropolis algorithm [11]. This process requires eight memories/cell and some very simple hardware functions. Figure 2d shows the segmentation result corresponding to that of Fig. 2c. This result is much better, but the MRF cannot remove the misclassified stripe of Fig. 2c. Moreover, the MRF

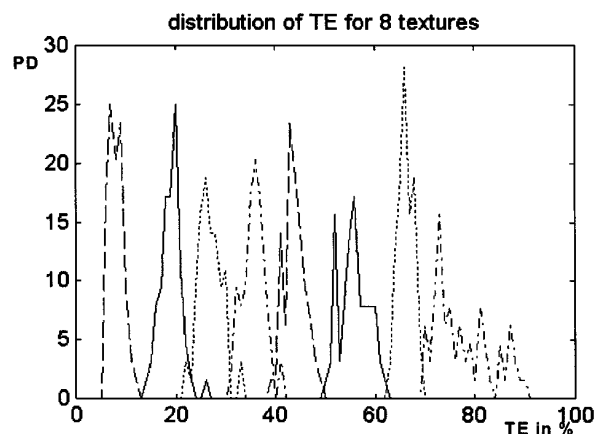


FIG. 5. Texture Energy (TE) histograms of textures from Fig. 4 after CNN transformation. TE is computed over a 21×21 detection window.

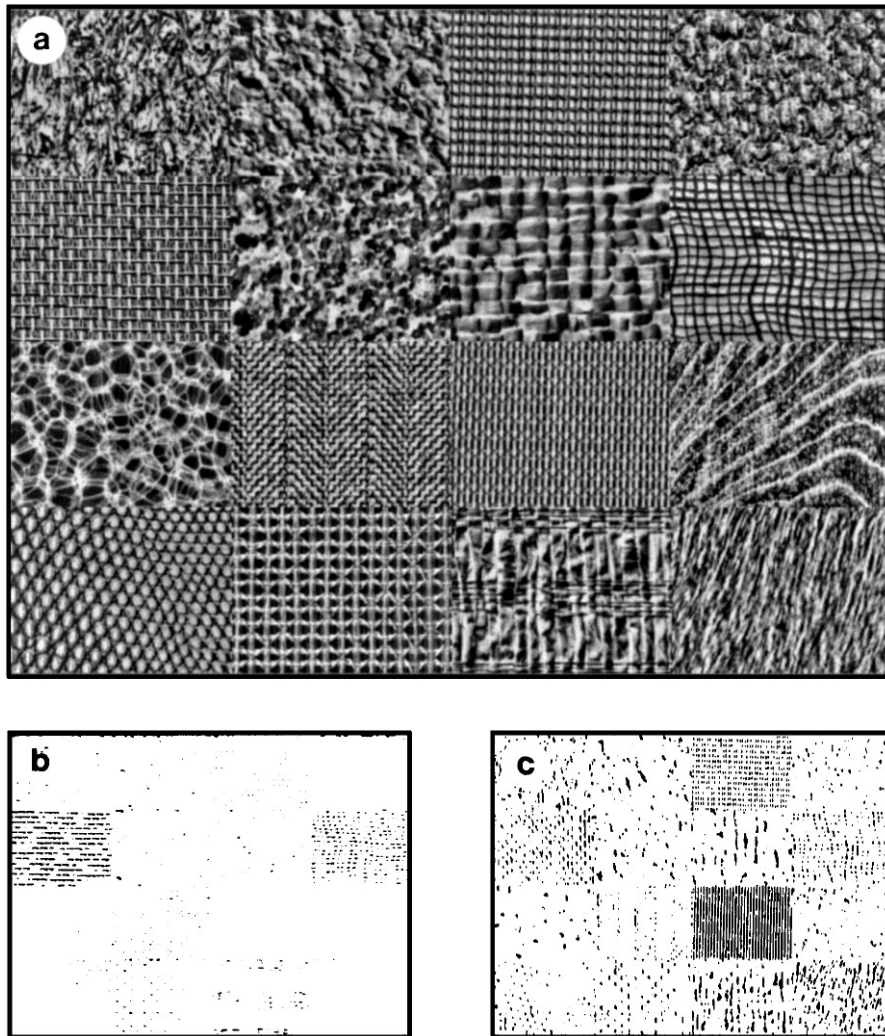


FIG. 6. Classification of a single texture out of 16 textures using pattern-sensitive CNN templates in the output image one texture will have higher average grey-level than the others. (a) Input test image (440 × 600) with the equalized grass (9), cork (4), French canvas (21), pigskin (92), aluminum wire (14), sand (29), raffia (84), burlap (104), bubbles (111), herring (17), straw cloth (53), wood (69), lizard skin (36), straw cloth (52), straw cloth (80), and fur (93) textures. (b) Output image when detecting aluminum wire. (c) Output image when detecting straw cloth.

CNN structure is a more complex architecture and requires considerably more processing.

We can avoid this problem with our method if we use more templates for the same texture set. The structure of the CNN-UM

[4, 5], containing template memories and cell memories, supports this solution. We run more templates for the same image containing multitexture patterns, and using the Bayesian decision (or a good estimate of it) the classification of every pixel is

TABLE 2
Experimental Results for the Classification of One Texture out of 16 by CNN: Using a Specific Template, the Named Texture is Enhanced by the Highest ROB Value in the Output Image

AluWire (14)				Straw cloth (53)				French canvas (21)				Herringbone (17)			
4	2	5	1	55	36	85	20	65	116	259	95	139	126	122	129
153	7	14	49	61	27	60	51	21	109	81	158	140	106	126	104
1	6	0	1	19	17	369	22	101	40	1	62	99	272	89	117
1	11	13	0	22	33	93	98	17	155	69	6	131	121	85	79

Note. Segment positions refer to Fig. 6a, white = 0, fully black = 1000.

done in the statistically optimum sense. The most important is that the order of average gray-levels (\hat{G}_k) of the texture outputs should be different when using different templates.

The multitemplate process is as follows:

- First, each detection template is run on the training image, followed by a smoothing (running the heat-diffusion process), and histograms are calculated for every texture area. In the case of 4 textures and 5 detection templates we have 20 histograms.

- In the test process, the different templates followed by the diffusion run consecutively on the test image, saving the resulting smoothed output image each time.

- Finally, a Bayesian decision is made considering the different outputs and the histograms. This process can be embedded in a hardware system containing CNN VLSI chips and some logical/arithmetic units, as for character recognition in [24]. Since the histogram functions can be replaced by simple thresholding effects (as in Eq. (9)) with a marginal increase in the misclassification error, the above process could be embedded in an enhanced version of a CNN-UM VLSI chip.

Figure 7 shows the segmentation of raffia (84), aluminum wire (14), French canvas (21), and straw cloth (53). The segmentation

is based on the half-toned output of different 3×3 templates. Five detection templates are used in this demonstration: four of these are texture-specific for one of the four textures and one is a four-texture classifier. The last yields good results (see Row 1 in Table 1) using a 25×25 detection area in the classification, but this window is relatively large for segmentation. Figure 7b shows the output of this template. The texture-specific templates are similar to those given for Fig. 6 and Table 2. Figure 7c shows the results of segmentation. False stripes are removed and the misclassification error inside the texture areas is zero. However, using only three templates results in moderate misclassification error.

In the above example five different templates must be trained, and this process requires more computation. However, we now show that acceptable segmentation results can be achieved without specific training. Considering the eight-texture classification test (Figs. 4 and 5), the trained template is used to segment four textures: aluminum wire (6), pellets (66), and straw cloths (52, 53). Figure 8a shows the input image of four textures and Fig. 8b shows the squared result using this template. As we have shown previously, this template is not well suited for fine segmentation; it is primarily sensitive to the main features of eight textures. We

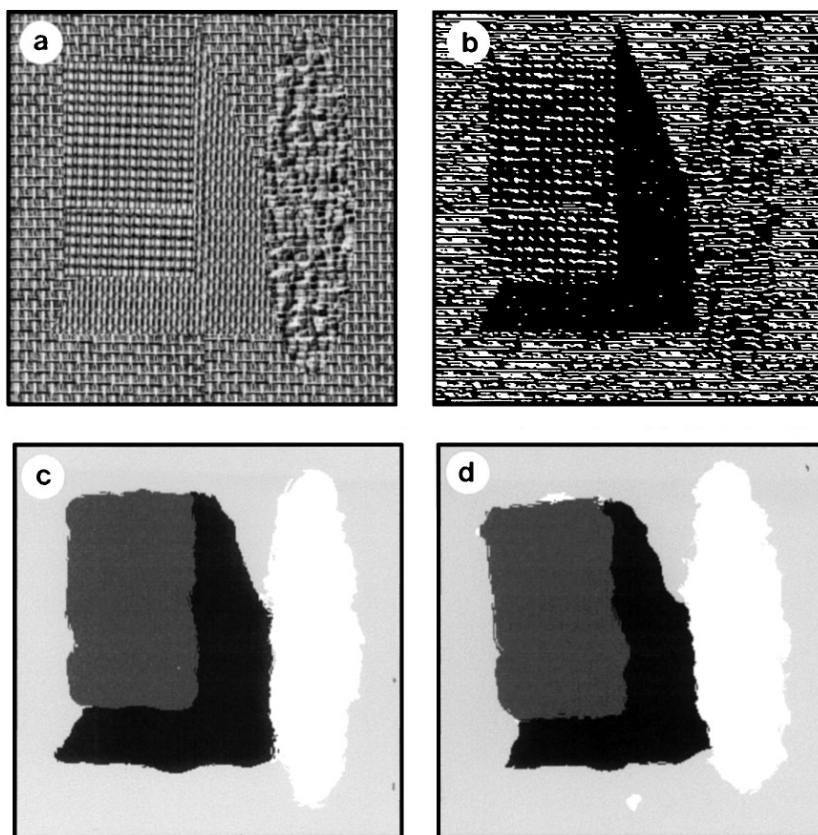


FIG. 7. Segmentation of four Brodatz textures using five different 3×3 CNN templates. (a) Test image (256×256) contains the equalized raffia (84), aluminum wire (14), French canvas (21), and straw cloth (53) textures. (b) Half-toned-like output using a single template. (c) Segmentation using Bayesian decision based on the gray-level histograms of texture outputs for each template. (d) Segmentation result when the input image containing the textures is rotated by a 5° angle.

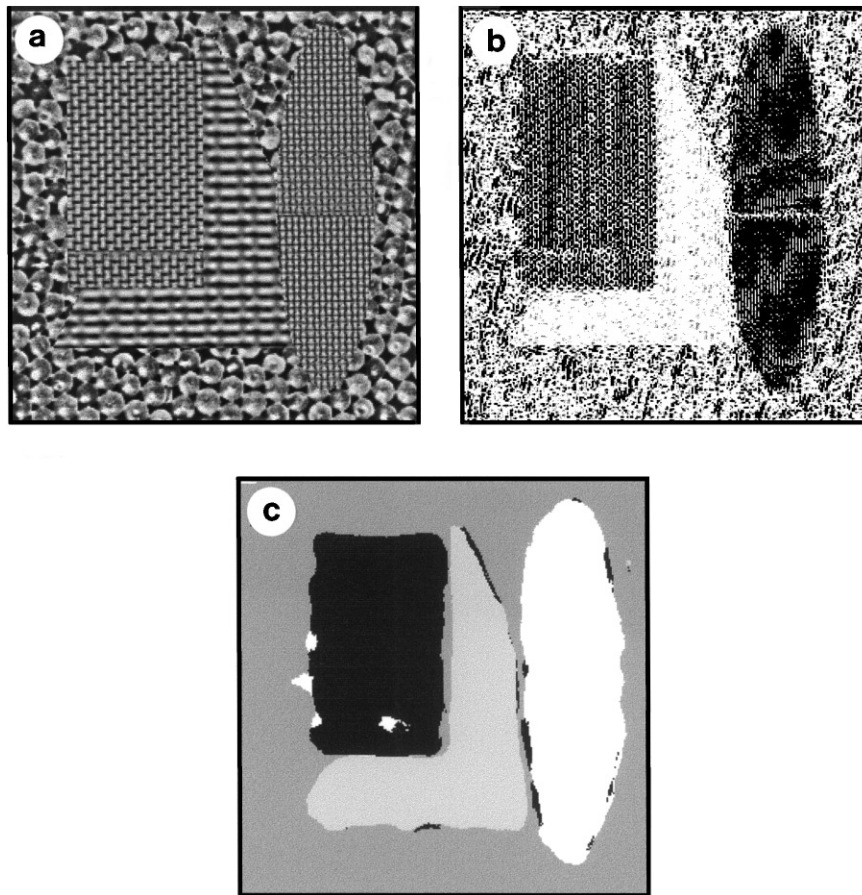


FIG. 8. Segmentation of four Brodatz textures with the template used in the example of Fig. 4 and four other templates not dedicated to this set of textures. (a) Test image (256×256) contains the equalized aluminum wire (6), pellets (66), straw cloth (53), and straw cloth (52) textures. (b) Squared output using a single CNN template. (c) Segmentation using Bayesian decision based on the TE histograms of texture outputs for each template.

have chosen other templates which are sensitive to other texture sets. These templates have poor classification results for this set, but using them together with the one usable template, segmentation is quite good, as Fig. 8c shows.

Using the above multitemplate method for the classification test of eight textures, the misclassification error can be lowered significantly.

4.5. Inhomogeneity, Orientation, and Scale Independence

In the previous examples the orientation and periodicity of the textures change slightly (e.g., see Fig. 3, lizard skin). Our results show that this system is insensitive to small amounts of scaling and rotation. This conclusion is also supported by the chip experiments (see Section 5), when the illuminated patterns are slightly rotated and lifted above the chip window in the measurement. However, the trained CNN can not only classify the textures appropriately, but also detect inhomogeneities inside the textures. Figure 7d shows that texture-sensitive templates are invariant to a small rotation (5° in the example) of the image.

The CNN can be trained to classify a group of textures into one class and another group of different textures into another

class, etc. Different rotations and scaled versions of the same texture can form such a texture group. First, simple scaling- and 90° -rotation-independent texture detection is tested, using four groups of textures. Each group contains a given texture at 1 : 1 and 1 : 2 scaling with original and 90° rotated images. The four textures are raffia (84), aluminum wire (14), French canvas (21), and lizard skin (36). The detector template detects the rotated and scaled versions together, so this template is rotation- and scaling-invariant in this sense. Using a 35×35 detection window, the classification error is 2.7% on the half-toned output image.

In the next experiment, different rotations are checked for the previous texture set. From every texture a quadruple image is generated. Each quadruple contains the same texture at 0° , 25° , 50° , and 75° rotations. We used a 5×5 template and squared output. Classification error is 12% using a 35×35 detection window and is 0.0% using 55×55 detection windows scanning the test image. The detector template detects the rotated versions together, so this template is a rotation-invariant classifier. Average gray-levels in % for the four quadruples in the output of the test image are (63, 65, 66, 66), (55, 56, 57, 59), (71, 71, 73, 76), (77, 78, 79, 80).

TABLE 3
Experimental Results for the Classification of Rotated Textures by CNN

Texture type	Template size	Contrast to others	Classif. error	Window size	ROB in %			
					AluW	Canvas	Lizard	Raffia
AluWire (14)	5 × 5	7%	1.0%	41	71–85	59–63	48–64	61–64
Canvas (21)	3 × 3	4%	0.5%	37	73–78	85–90	70–79	72–80
LizardSkin (36)	5 × 5	18%	3.6%	31	39–64	39–66	84–90	40–46
Raffia (84)	5 × 5	9%	6.8%	31	10–25	3–11	6–19	34–40

In this experiment a relatively large detection-window and a large template (5 × 5) are required in order to obtain good classification results. When only one type of textures is detected from the others, the classification of rotated textures can also be performed on the basis of half-toned outputs. In this case one texture class is classified from the set of four classes by different templates for each texture-class. In Table 3 we present the results of the rotation-independent texture classification.

4.6. Robustness and Noise Sensitivity of the CNN Texture Detector

In [18] it has been shown that different tasks, such as 2D or 3D deconvolution [22] and texture segmentation, can be solved in a VLSI CNN environment without significant loss of efficiency and accuracy when the low precision (about 6–8 bits) and random variability of the VLSI parameters are considered. CNN turns out to be very robust against template noise, image noise, imperfect estimation of templates, and parameter accuracy.

Parameters of a template are tuned using GA learning at different parameter accuracies (represented by bitlength of parameter). These optimized parameters depend on the precision of the architecture. It was found that about 6–8 bits of precision was sufficient for a complicated multilayer deconvolution, and only 4 bits of precision was sufficient for texture classification in the presence of noise and parameter variances in the case of binary (half-toned) output. Tolerance sensitivity of template parameters is considered for VLSI implementation. Figure 9a shows the robustness–accuracy plot for the textures of Figs. 3 and 7 when the parameters are represented for different bitlength accuracies. Figure 9b shows the effect of parameter fluctuations for different parameter accuracies. Figures 9a and 9b together demonstrate the robustness of the CNN texture detector versus VLSI parameter inaccuracies. It was found that the misclassification error changes only slightly in the case of image noise (Fig. 9c). These results show that the CNN process for texture detection is very robust, although the process is dynamic and the VLSI parameters may be inaccurate. This can be understood by considering the effect of the feedback [18]: In each case of local filtering the CNN executes an error propagation process (as in half-toning [17]), which optimizes the output depending on the template characteristics and the input. This process does not accumulate the computation error, since the feedback controls

the direction of the process. This means that noise and structural uncertainty do not affect the results when the CNN process itself is stable. In texture analysis, the template executes a kind of convolution/deconvolution in addition to moving effects. However, this moving (and the correlation) depends rather on the signs of the template elements than on the exact template values.

4.7. Unsupervised Training

In the previous examples of supervised learning the first question is whether the task is solvable in the given system. This can be answered using unsupervised learning, in which we search for a template which results in the most diverse output values (gray-levels of smoothed output image). In this case, the fitness should be computed in a way which maximizes the entropy (≤ 1.0) of the histogram of gray-levels in the whole smoothed output image. Therefore, we are looking for a CNN template which extracts the most diverse information from the input image with uniform mid-gray when it is smoothed. From the resulting output image of the template with the maximum entropy we can find the distinguishable and indistinguishable patterns in the given case. We achieved results similar to those previously reported for the supervised training (as in Fig. 4). However, this unsupervised training results in the detection of common features of the different textures rather than the textures themselves, but many of the textures can be well segmented.

5. EXPERIMENTS WITH AN ANALOG 22 × 20 VLSI CNN CHIP

The previous experiments and other robustness results [18] show that even a simple, noisy, inaccurate CNN chip with a very limited number of cells may achieve good performance in texture classification.

We tested one of the latest types of CNN–UM chips. This chip [5, 27] has a simplified structure, since its input and output are binary. It has four analog memories per cell. Logical functions and analog 3 × 3 templates can be run on the chip. Template parameters are transmitted to the chip via eight-bit DA converters. The chip has optical inputs with 1 sensor/1 pixel scanning. This optical input array is adaptive to achieve the ratio of black pixels (ROB) of around 50% in the scanned input.

In this experiment we used different Brodatz textures printed onto a transparent sheet. This transparency is mounted onto the

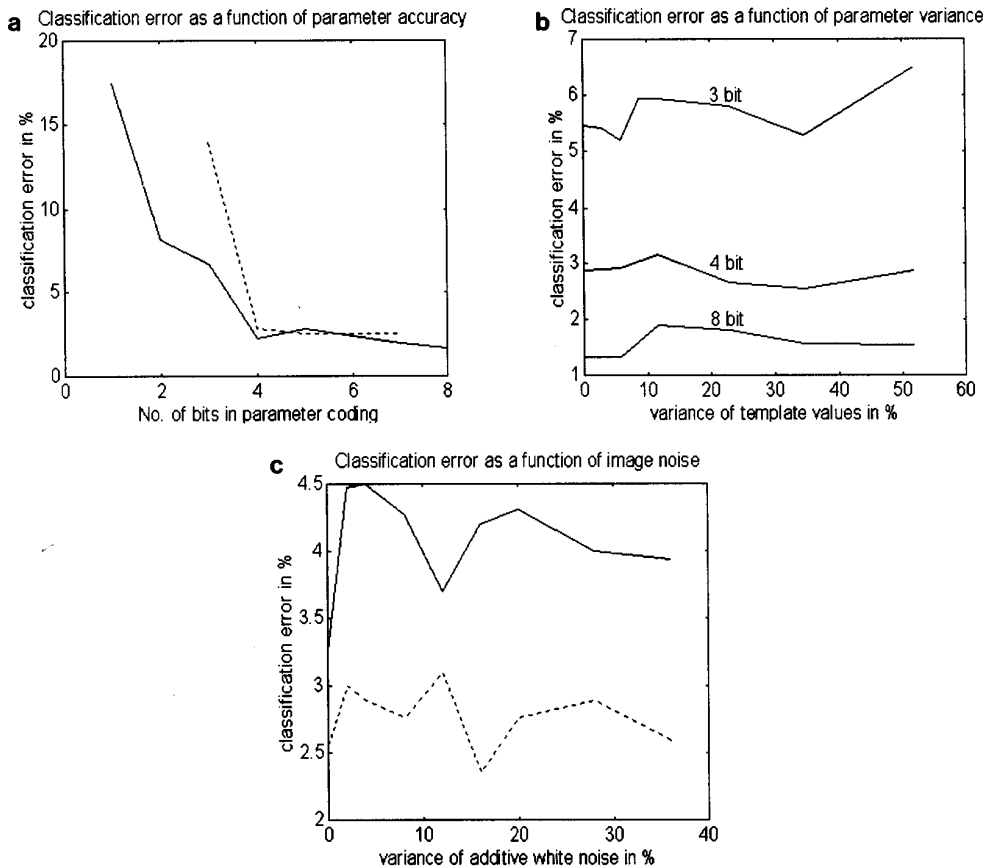


FIG. 9. Results of robustness tests. (a) Texture misclassification error as a function of parameter accuracy. CNN templates were trained for the textures of Fig. 3 (dashed line) and Fig. 7 (solid line), using different bitlengths in parameter coding. It can be seen that there is an optimal selection for bitlength ensuring good convergence. (b) Texture misclassification error as a function of template value fluctuation. The test was performed for three different bitlengths using the textures in Fig. 7. (c) Texture misclassification error as a function of image noise. The templates trained for textures of Fig. 3 (solid line) and Fig. 7 (dashed line) were tested adding white noise of different variances to the textured image.

window of the chip and is illuminated with a simple table lamp. The image samples are captured at different positions, but at a good resolution (0.2 mm/pixel). This configuration ensures random sampling at changing illumination and noise. It is important to note that in these chip experiments the illuminated patterns are slightly rotated and lifted during the measurement. The measurements show that this method has at least modest robustness against scaling, blurring and rotation.

Since the sigmoid function of the state and the binary input/output differ from the conventional CNN model [1, 2, 4] and the chip array is not uniform, we implemented our GA using the chip itself in a real-time learning process. Training and test images are real-time scanned by the on-chip sensors. The training period for 3–5 textures is only a few minutes.

The ROB is about 0.49 for the input images, with some small variation which is independent of the texture class.

Figure 10 shows four examples of the original, the scanned-thresholded input, and the output images using a simple template. The texture set consists of French canvas (21), straw cloths (52 and 53), and straw matting (55). As a result, the ROB at

the output is significantly different for the four texture classes. The distributions of ROB for the different classes are plotted in Fig. 11a. Each curve is measured over 4000 test samples. As shown, the histograms of the different texture classes can be well separated. Here the histogram is the measured probability density (PD) function of the output ROB values for a given texture. Measuring the contrast by the average difference of the neighboring histogram peaks of the different texture classes, this contrast is better than 5% for four textures, while the in-class consistency (the curve width around a peak) is good.

These measurements take only a few seconds. Further, only a small portion of the processing time is devoted to the chip-process, with a greater amount of time necessary for the computer interface and the evaluation. When a smart-sensor architecture contains the CNN chip and a few on-board hardware components, this processing time can be reduced to be less than 10 μ s/test.

Figure 11b shows a robustness experiment for the textures canvas and matting. These textures are scaled (lifting up) and rotated in random positions. The histogram changes slightly,

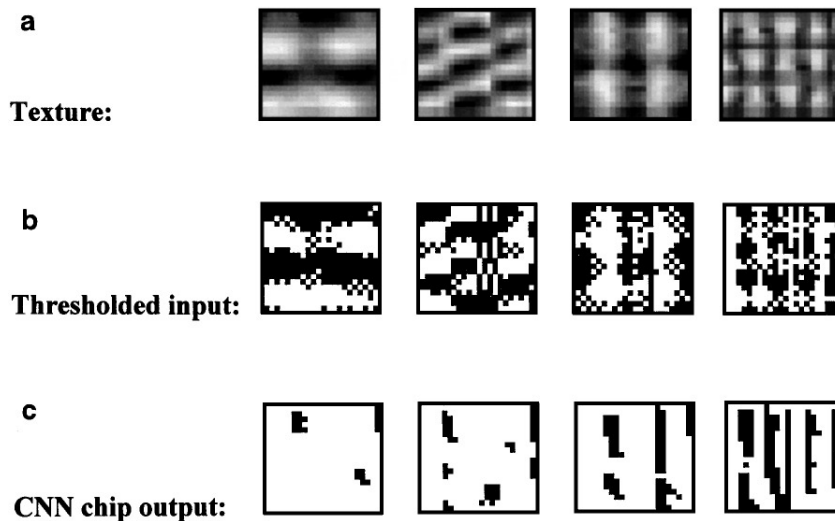


FIG. 10. Classification of 4 Brodatz textures by a programmable VLSI CNN chip of size 22×20 pixels with on-chip optical input. (a) Sample images on the chip-window which are scanned by the on-chip photosensors. The texture set consists of French canvas (21), straw cloths (52 and 53), and straw matting (55). (b) Adaptively thresholded binary input after scanning in. (c) Binary output image after CNN transformation with different ROB for the different textures using a single 3×3 template.

but the results of the imperfect texture images remain separable. Using another chip of the same type, the histograms are nearly identical, meaning that the VLSI parameters are independent of the identity of a chip.

In these experiments the Bayes misclassification error is about 4–5% for four textures, 2–4% for three textures, and 0–4% for two textures. We have also tested other texture sets with similar results.

In another experiment, we trained the chip to detect different rotations of textures containing nearly straight sections. The template was trained for one texture, but it can detect any other if the periods are in similar range.

In the above experiments only one template is used. Exploiting the template and image storing capabilities of the chip, more precise recognition can be achieved in a *multitemplate process*. Using three templates to classify the textures of Fig. 10, the misclassification error is only 0.15%. Rotating and tilting the image-sheet result in a small increasing of the error (0.7%). We tested a different texture set using four different templates. Misclassification error is 0.8% and it is only 1.2% in the case of texture-rotation and sheet-tilt in a wide range.

6. CONCLUSIONS

We have demonstrated a CNN-based method for texture classification and fine segmentation. Our method is robust against noise and template variance which can arise from VLSI inaccuracies. Classification works well using simple and small VLSI CNN chips as well. Segmentation failed only if the features changed significantly within a single texture. The CNN can also detect the inhomogeneity (e.g., error) or orientation within a given texture.

The CNN can be trained to perform orientation and scaling independent detection as well. However, if we want to classify many textures with different orientations and scaling using only one CNN template, the window-size for classification has to be larger. If we use more templates (e.g., different templates to extract different textures), segmentation will be more accurate.

The CNN has superior detection performance when the different texture classes have similar periodicity and they differ only in small structural features.

The templates are determined by GA learning. Developing a template library for a class of problems, it is easy and relatively fast to generate the most appropriate template for a given task. The results presented in Tables 1, 2, and 3 and in Figs. 6, 7, and 8 demonstrate that the performance of this CNN-based texture detection is (at least) similar to that of the other methods (e.g.,

TABLE 4
Comparison of Conventional Convolution-Based Classification and the CNN Method: Test Results for Classification of Eight Textures by 5×5 Kernels and TE using Simple Feedforward Convolutions (Conv) and CNN Templates

Texture set	Window size	Misclassification % Conv \leftrightarrow CNN	Contrast % Conv \leftrightarrow CNN
Set 1 (Fig. 4)	11	36 \leftrightarrow 23	66 \leftrightarrow 93
	15	25 \leftrightarrow 16	66 \leftrightarrow 93
	25	8.5 \leftrightarrow 5.4	67 \leftrightarrow 96
	41	1.0 \leftrightarrow 0.5	67 \leftrightarrow 94
Set 2	15	31 \leftrightarrow 16	58 \leftrightarrow 92
	25	15 \leftrightarrow 5.6	58 \leftrightarrow 92
	41	6.6 \leftrightarrow 3.4	58 \leftrightarrow 92

Note: Kernels have been optimized for each size of detection-windows.

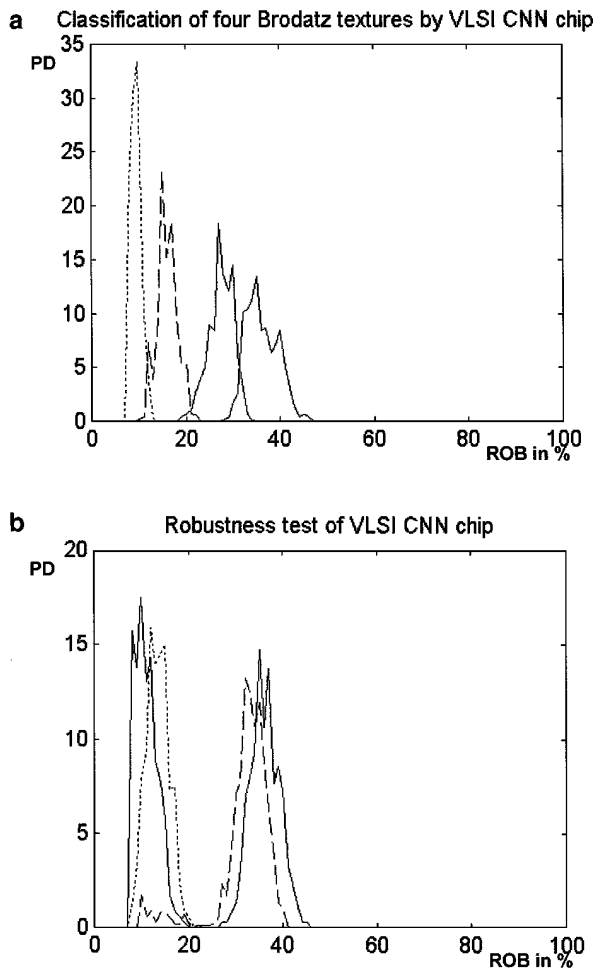


FIG. 11. Classification statistics of the chip measurements, testing the ROB values for four different texture-classes. (a) Histograms (PD) generated by moving the textures over the scanning window of the chip sampling at 4000 different locations for each texture and measuring the ROB of the output. (b) Robustness test of the template used in the above experiment for two of the textures. The transparency holding the textures was lifted to a small extent above the chip surface, rotated around the vertical axis and tilted max. 10° round the horizontal axes. Histograms of textures in the right positions (solid lines) are nearly identical to the results of unsharp and rotated positions (dashed lines).

[32–36]), at a superior speed. However, while these methods use relatively large detection windows (128×128 in [34], 64×64 in [7, 32], or 32×32 in [35]), our method makes it possible to use smaller windows, and it results in better segmentation properties. This is due to a deconvolution effect: the filtered output image of a quasiperiodic texture is also a fine structured image.

To be more specific, we compared our method to the simple feedforward convolution-based methods (as in [8, 32, 35]). We tested them in our environment, using the same GA optimization method to determine the convolution kernels. The results of this test are listed in Table 4 for two texture sets with eight textures in each. These results show that the CNN structure, containing both feedforward and feedback convolutions, can achieve better contrast and smaller misclassification error (the factor is

about 1.5–2 for both parameters) than the simple convolution methods.

Since the CNN can result in stable half-toned output in the case of texture classification, this system is very stable in the presence of noise and inaccuracies arising from VLSI analog chips. Moreover, as our chip-tests show, using a CNN VLSI chip yields dramatic improvements in processing speed. The whole texture-classifier system can be incorporated in a small smart sensor containing only minimal hardware elements and the CNN chip.

ACKNOWLEDGMENTS

Special thanks are due to Professor T. Roska and T. Kozek for helpful discussions and to P. Földesi for the assistance in chip tests. We thank Professor A. Rodriguez-Vazquez and his laboratory at the University of Seville for the CNN chips which were given to our laboratory for testing. The help of the anonymous reviewers is gratefully acknowledged. This work was supported by the Hungarian Research Fund (OTKA T-017248 and T-019062).

REFERENCES

1. L. O. Chua and L. Yang, Cellular neural networks: Theory, *IEEE Trans. Circuits Systems* **35**, 1988, 1257–1272.
2. L. O. Chua and L. Yang, Cellular neural networks: Application, *IEEE Trans. Circuits Systems* **35**, 1988, 1273–1290.
3. T. Roska and T. Szirányi, Classes of analogic CNN algorithms and their practical use in complex image processing tasks, in *Proceedings, IEEE Nonlinear Signal and Image Processing*, Vol. 2, pp. 767–770, Halkidiki (Greece), June 1995.
4. T. Roska and L. O. Chua, The CNN Universal Machine: An analogic array computer, *IEEE Trans. Circuits Systems II* **40**, 1993, 163–173.
5. S. Espejo, R. Carmona, R. Dominguez-Castro, and A. Rodriguez-Vazquez, A CNN universal chip in CMOS technology, *Int. J. Circuit Theory Appl.* **24**, 1996, 93–110.
6. T. R. Reed and J. M. H. Buf, A review of recent texture segmentation and feature extraction techniques, *CVGIP: Image Understanding* **57**, 1993, 359–372.
7. R. M. Haralick, K. Shanmugam, and I. Dinstein, Textural features for image classification, *IEEE Trans. Systems Man Cybernet.* **3**, 1973, 610–621.
8. K. I. Laws, Textured Image Segmentation, Technical Report USCPI Report 940, Department of Electronic Engineering, Image Processing Institute, University of Southern California, Los Angeles, 1980.
9. P. Perona and J. Malik, Scale-space and edge detection using anisotropic diffusion, *IEEE Trans. Pattern Anal. Mach. Intelligence* **12**, 1990, 629–639.
10. R. Chelappa and S. Chatterjee, Classification of textures using Gaussian Markov random fields, *IEEE Trans. Acoust. Speech Signal Process.* **33**, 1985, 959–963.
11. T. Szirányi and J. Zerubia, Markov random field image segmentation using cellular neural network, *IEEE Trans. Circuits Systems I* **44**, 1997, 86–89.
12. S. N. Jayaramamurthy, Texture discrimination using digital deconvolution filters, in *Proceedings, 5th International Conference on Pattern Recognition, Miami Beach, Florida*, pp. 1184–1186, 1980.
13. G. Zelniker and F. J. Taylor, *Advanced Digital Signal Processing*, Dekker, New York, 1994.
14. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall International, London, 1989.
15. P. Rózsa, *Lineáris Algebra és Alkalmazásai*, Műszaki Könyvkiadó Publ., Budapest, 1976. [In Hungarian]

16. J. Ens and P. Lawrence, An investigation of methods for determining depth from focus, *IEEE Trans. Pattern Anal. Mach. Intelligence* **15**, 1993, 97–108.
17. K. Crouse, T. Roska, and L. O. Chua, Image halftoning with cellular neural networks, *IEEE Trans. Circuits Systems* **40**, 1993, 267–283.
18. T. Szirányi, Robustness of cellular neural networks in image deblurring and texture segmentation, *Int. J. Circuit Theory Appl.* **24**, 1996, 381–396.
19. L. O. Chua and T. Roska, Stability of class of nonreciprocal cellular neural networks, *IEEE Trans. Circuits Systems* **37**, 1990, 1520–1527.
20. T. Kozek, T. Roska, and L. O. Chua, Genetic algorithm for CNN template learning, *IEEE Trans. Circuits Systems* **40**, 1993, 392–402.
21. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison–Wesley, New York, 1993.
22. J. P. Miller, T. Roska, T. Szirányi, K. R. Crouse, L. Nemes and L. O. Chua, Deblurring of images by cellular neural networks with applications to microscopy, in *Proceedings, 3rd IEEE Workshop on CNN and their Applications, Rome*, pp. 237–242, 1994.
23. L. O. Chua and C. W. Wu, On the universe of stable cellular neural networks, *Int. J. Circuit Theory Appl.* **20**, 1992, 497–517.
24. T. Szirányi and J. Csicsvári, High speed character recognition using a dual cellular neural network architecture (CNND), *IEEE Trans. Circuits Systems II* **40**, 1993, 223–231.
25. T. Roska, G. Bártfay, P. Szolgay, T. Szirányi, A. Radványi, T. Kozek, Zs. Ugray, and Á. Zarándy, A digital multiprocessor hardware accelerator board for cellular neural networks: CNN-HAC, *Int. J. Circuit Theory Appl.* **20**, 1992, 589–599.
26. P. Kinget and M. S. J. Steyaert, A programmable analogue cellular neural network CMOS chip for high speed image processing, *IEEE J. Solid-State Circuits* **30**, 1995, 235–243.
27. R. Dominguez-Castro, S. M. Espejo, A. Rodriguez-Vazquez, R. Carmona, P. Földesy, A. Zarándy, P. Szolgay, T. Szirányi, and T. Roska, A 0.8 μm CMOS 2-D programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage, *IEEE J. Solid-State Circuits* **32**, 1997, 1013–1026.
28. M. Csapodi and T. Roska, Adaptive histogram equalization with cellular neural networks, in *Proceedings, 4th IEEE Conference on Cellular Neural Networks and Applications, CNNA-96, Seville*, pp. 81–86, 1996.
29. P. Brodatz, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, 1966.
30. T. Szirányi and Á. Böröczky, Overall picture degradation error for scanned images and the efficiency of character recognition, *Opt. Enging.* **30**, 1991, 1878–1885.
31. T. Szirányi, Subpixel pattern recognition by image histograms, *Pattern Recognition* **27**, 1994, 1079–1092.
32. J. You and H. A. Cohen, Classification and segmentation of rotated and scaled textured images using texture tuned masks, *Pattern Recognition* **26**, 1993, 245–258.
33. J. M. H. Buf, M. Kardan, and M. Spann, Texture feature performance for image segmentation, *Pattern Recognition* **23**, 1990, 291–309.
34. M. K. Tsatsanis and G. B. Giannakis, Object and texture classification using higher order statistics, *IEEE Trans. Pattern Anal. Mach. Intelligence* **14**, 1992, 733–750.
35. M. Pietikainen, A. Rosenfeld, and L. S. Davis, Experiments with texture classification using averages of local pattern matches, *IEEE Trans. Systems Man Cybernet.* **13**, 1983, 421–426.
36. H. H. Nguyen and P. Cohen, Gibbs random fields, fuzzy clustering, and the unsupervised segmentation of textured images, *CVGIP: Graph. Models Image Process.* **55**, 1993, 1–19.