SOFM Neural Network Texture Segmentation

Erik Cuevas[1,2], Daniel Zaldivar[1,2], Marco A. Pérez-Cisneros[2] and Rául Rojas[1]
[1]Freie Universität Berlin, Takustr. 9
14195 Berlin, Germany
{cuevas, zaldivar, rojas}@inf.fu-berlin.de
http://www.inf.fu-berlin.de
[2]Universidad de Guadalajara, CUCEI, Blvd. Marcelino Garcia Barragan 1421,
44430 Guadalajara, Jalisco, Mexico
marcopc@cucei.udg.mx

(Paper received on August 11, 2006, accepted on September 26, 2006)

Abstract. Texture segmentation is a difficult problem, just as it is apparent from camouflage pictures. A textured region can contain elements of several sizes, each of which can itself be textured. This work presents an algorithm to segment textures. The algorithm is based on self-organizing feature maps which are used to generate a histogram that characterizes the texture. Through classification, the histograms are compared to each other using signal cross-correlation that operates on a defined window according to the texture complexity. The algorithm was tested on benchmark images. The experiments, their results and relevance are presented in the results and future work section.

1. Introduction

Texture is generally recognized as being fundamental to perception. The taxonomy of problems encountered within the context of texture analysis could be that of classification, description, and segmentation. Recognition of texture patterns has applications in radiography and aerial and satellite photography, among others. There is no concise definition or characterization of a texture available in practice. Texture has been described in a variety of ways. Intuitively, texture descriptors provide measures of properties such as smoothness, coarseness, and regularity. One way to describe texture is to consider it as being composed of elements of texture primitives. Texture can also be defined as the mutual relationship among intensity values of neighboring pixels repeated over an area larger than the size of the relationship.

Texture segmentation is the problem of breaking an image into components within which the texture is constant. Texture segmentation involves both representing a texture, and determining the basis on which segment boundaries are to be determined.

Many texture feature extraction and recognition algorithms are available in practice ([1], [2], [3], [4], [5], [6], [7], [8], [9]). Conventional texture recognition algorithms can be grouped into three classes: statistical, structural, and spectral. Statistical approaches yield characterizations of textures as smooth, coarse, grainy, and so forth. Statistical algorithms are based on the relationship between intensity values of pixels; measures include entropy, contrast, and correlation based on the gray level co-occurrence matrix. Structural algorithms are based on image primitives, which they regard as a formational element. Structural algorithms generate, and describe rules for

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 259-268

generating, repeating patterns. The notion of a primitive is central to texture analysis. A *texel* is (loosely) a visual primitive with certain invariant properties. Texels occur repeatedly in different positions, deformations, or orientations inside a given area. Texture primitives may be pixels or aggregates of pixels. One way of describing rules that govern texture is through a grammar. Structural approaches generate patterns by applying the rules of a grammar to a small number of symbols. Spectral techniques are based on properties of the Fourier spectrum and are used primarily to detect global periodicity in the image by identifying high-energy narrow peaks in the spectrum. Both statistical and structural measures lack neurophysiological support [10].

Many neural network models have been suggested for texture recognition ([11], [12]). A generic model for segmenting images by using texture requires the identification of those features that both define texture and allow discrimination between different textures. A class of 2-D filters based on Gabor functions for the texture segmentation was proposed [15], in this approach is shown analytically that applying a properly configured band-pass filter to a textured image produces distinct output discontinuities at texture boundaries. Rao and Vemuri proposed a neural network architecture for texture segmentation and labeling. Their model consists of two major components: the feature extraction network and the texture discrimination network. The feature extraction network is a multilayer hierarchical network governed by Grossberg's boundary counter (BC) system [13]. The texture discrimination network is based on the adaptive learning algorithm devised by Kohonen [14]. Neural network models based on FT-domain feature extraction can also be used for texture feature extraction.

This work this organized in the following way, in the section 2 we analyze the main characteristics of the SOM neural networks and define the used nomenclature, in the section 3 the proposed algorithm is described, finally in the section 4 the results are shown and it is analyzed future work.

2. SOFM neural networks

Self-organizing feature maps (SOFM) [14] learn to classify input vectors according to how they are grouped in the input space. They differ from another networks in that neighboring neurons learn to recognize neighboring sections of the input space. Thus, competitive layers learn both the distributions and topology of the input vectors that they are trained on.

The architecture for a Self-organizing map network is shown in Fig. 1. The |Ndist| box in the figure accepts the input vector **p** and the input weight matrix **IW** and produces a vector having *S* elements. The elements are the negative of the distances between the input vector **p** and the vector **IW**. The net value **n** of the Self-organizing layer is computed by finding the negative distance between input vector **p** and the weight vector **IW**.

The competitive transfer function C accepts a net value **n** and returns neurons outputs of 0 for all neurons except for the winner, the neuron associated with the most positive element of the input **n**. Thus, the winner's output is 1.

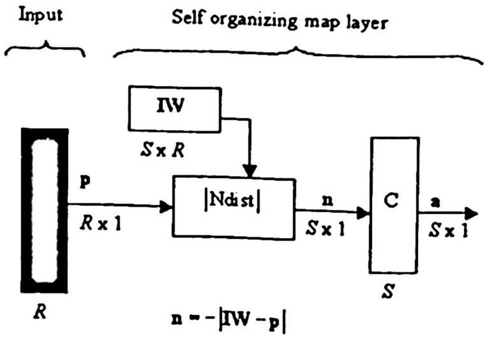


Fig. 1. SOM neural network architecture.

The neurons in the layer of an SOM are arranged originally in physical positions according to a topology function, the more used are square grid, hexagonal and random topology. Distances between neurons are calculated from their positions with a distance function. The Link distance is the most common.

A self-organizing map network identifies a winning neuron i^{th} using the same procedure as employed by a competitive layer, updating not only the winning neuron, but all neurons within a certain neighborhood $N_{i^*}(d)$ of the winning neuron using the Kohonen rule. Specifically, we adjust all such neurons $i \in N_{i^*}(d)$ as follows:

$$ {}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) \text{ or} \tag{1}$$

$$ {}_i\mathbf{w}(q) = (1-\alpha)\,{}_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q) \tag{2}$$

Here α is the learning rate and $N_{i^*}(d)$ contains the indices for all of the neurons that lie within a radius d of the i^{th} winning neuron. Thus, when a vector **p** is presented, the weights of the winning neuron and its closest neighbors move toward **p**. Consequently, after many presentations, neighboring neurons will have learned vectors similar to each other. The winning neuron's weights are altered proportional to the learning rate. The weights of neurons in its neighborhood are altered proportional to half the learning rate. In this work, the learning rate and the neighborhood distance (used to determine which neurons are in the winning neuron's neighborhood) are not altered during training.

Here the *neighborhood* $N_i(d)$ contains the indices for all of the neurons that lie within a radius *d* of the winning neuron i^{th}. To illustrate the concept of neighborhoods, consider Fig. 2.

$$N_i(d) = \{j, d_{ij} \leq d\} \tag{3}$$

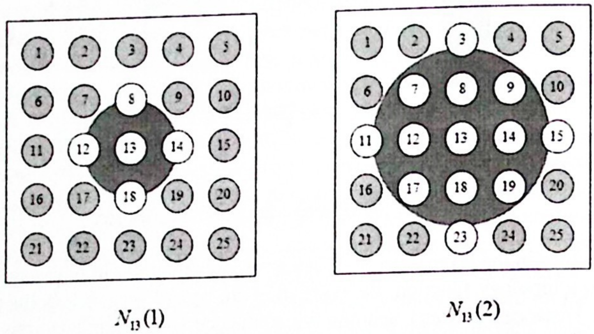


Fig. 2. At left is shown a one-dimensional neighborhood of radius *d*=1 around neuron 13, at right is shown a neighborhood of radius *d*=2.

These neighborhoods could be written as: $N_{13}(1) = \{8,12,13,14,18\}$ and $N_{13}(2) = \{3,7,8,9,11,12,13,14,15,17,18,19,23\}$.

3. Segmentation algorithm

To carry out the segmentation, it is defined a window of p1 x p2 pixels, with the condition that the window contains in "representative form" the structure that the texture defines. With this window the neural net SOM is trained. The neural net architecture has 3 input that represent the *x*, *y* coordinates and the pixel intensity while the outputs number depend from the complexity of the structure to segment, although in several experiments were showed to give a good result between 25 and 36 (located in a square arrangement of *h* x *k*) neurons. Fig. 3 shows the architecture of the neural net used.

Although the performance of the network is not sensitive to the exact shape of the neighborhoods with the objective of observing the output layer deformation and its covering with the data space of the training window the square grid was chosen, using as distance approach the link distance. The figure 4b shows the deformation presented at the end of the training considering the window shown in Fig. 4(a).

Once trained the net, the net has the capacity to order the input space, that is to say it will learn how to react to repetitive structures found in the window, and considering the training and operation form of these nets, structures that are similar will make to react the neighboring neurons. This is if for an input pattern the neuron 3 reacts as the winner neuron, if a pattern is presented with similar characteristic will react as winner neuron one of the neighboring neurons of 3.

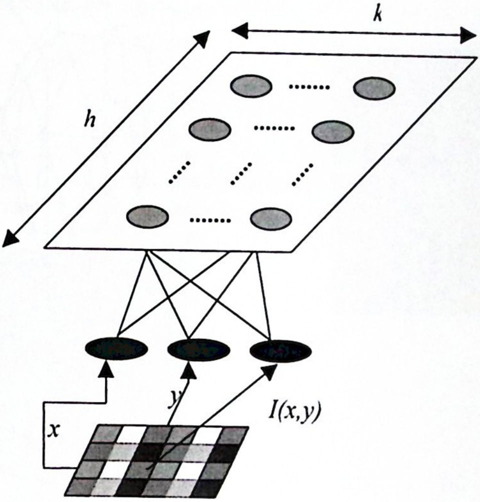
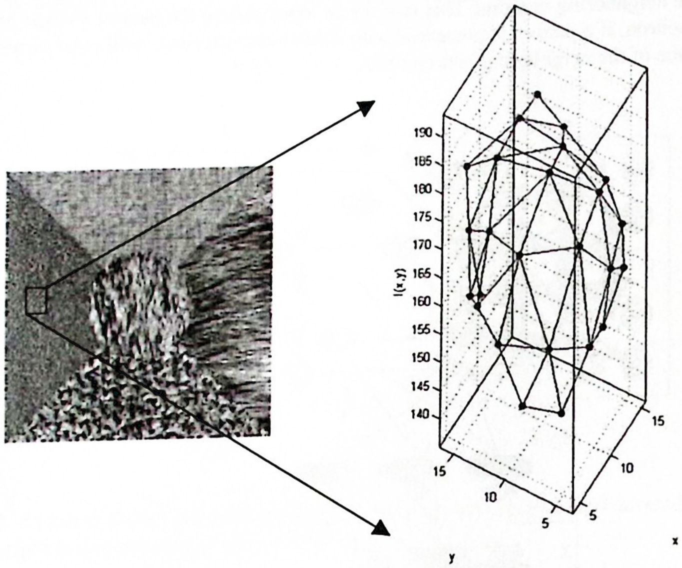


Fig. 3. The neural net architecture has 3 input that represent the x, y coordinates and the pixel intensity.

To characterize the texture using the trained neural network we use the histogram obtained by the net when it is presenting as input the pixels contained in the window. The histogram will contain the information of how many times each neuron was shot as a consequence of have been processed the window pixels. The histogram allows to facilitate the characterization process converting the problem of several dimensions to an one-dimensional problem. Fig. 5(b) shows the obtained histogram considering as training window the figure shown in 5(a).

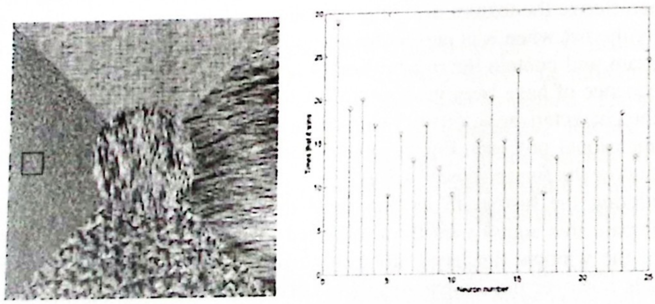
Considering the histogram as the signal that characterizes the texture we can simply compare this signal with those obtained for the histograms generated by the versions of the window displaced along the image. Many forms exist of comparing these signals however the simple way is to accomplish the cross-correlation of both signals.

In statistics, the term **cross-correlation** is sometimes used to refer to the covariance between two random vectors X and Y. In signal processing, the **cross-correlation** (or sometimes "cross-covariance") is a measure of similarity of two signals, commonly used to find features in an unknown signal by comparing it to a known one. It is a function of the relative time between the signals, is sometimes called the *sliding dot product*, and has applications in pattern recognition and cryptanalysis.



(a) (b)

Fig. 4. Figure 4(b) shows the deformation presented at the end of the training considering the window shown in Fig. 4(a).



(a) (b)

Fig. 5. Figure 5(b) shows the obtained histogram considering as training window the figure shown in 5(a).

For discrete functions f_i and g_i the cross-correlation is defined as

$$(f * g)_i \equiv \sum_j f_j^* g_{i+j} \tag{4}$$

where the sum is over the appropriate values of the integer j and an asterisk indicates the complex conjugate. For continuous functions $f(x)$ and $g(x)$ the cross-correlation is defined as

$$(f * g)(x) \equiv \int f^*(t) g(x+t) dt \tag{5}$$

where the integral is over the appropriate values of t. The cross-correlation is similar in nature to the convolution of two functions.

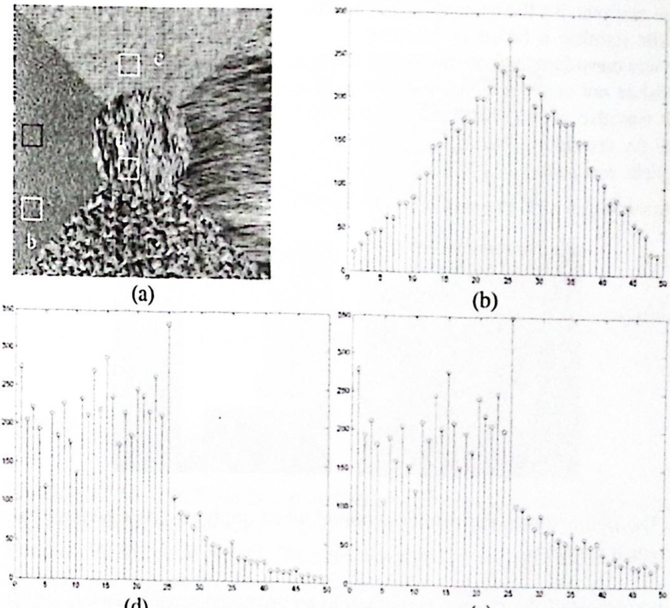


Fig. 6. In the figure 6b is shown the cross-correlation of windows that correspond to the same texture presenting a high correlation among both signals, which does not happen to the signals of the windows histogram of other textures.

In Fig. 6 the histogram obtained in Fig. 5 that characterizes to a texture is compared by cross-correlation with different windows placed in different textures. In Fig. 6(b) it is shown the cross-correlation of windows that correspond to the same texture presenting a high correlation among both signals, which does not happen to the signals of the windows histogram of other textures.

The segmentation algorithm of the textures is divided in two parts as usually happen in applications of neural networks, they are training and operation. During the training for each texture a neural network SOM is trained. The net architecture is similar to the one shown in Fig. 3. Considering the same training window it is obtained the histogram which will be the signal that characterizes the texture. In the operation the window is displaced by the whole image accomplishing the cross-correlation among the signals that characterizes the texture and the new histogram obtained as a result of applying as input to the net the displaced window.

4. Results and future Work

The algorithm was implemented using the neural architecture SOM proposed, using 25 output neurons, for the training and operation was used a window of 20 x 20 pixels. During the training a factor of learning of 0.2 was used and a neighborhood of 5 neurons was considered, what means that the modifications to the network weights are accomplished not only on the winning neuron, but in those that are in a neighborhood of 5. It was also used in the training a square grid and as distance approach the link distance. As discrimination approach the mean of theobtained cross-correlation of both signals was used being this $\mu_{correlation} = 210$. The training, the correlation and the image processing was implemented in MATLAB.

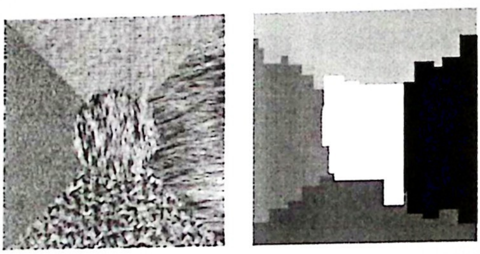


Fig. 7. The figure shows the results obtained when applying the algorithm based on SOFM neural networks.

The implemented algorithm was applied to an image commonly used as benchmark for the comparison of segmentation methods, the result is shown in Fig. 7. The result is good in all homogeneous regions of the textures except in the edges where the algorithm does not allow to discriminate correctly. The Algorithm was proven with other images presented a good performance however the problem in the edge remains constant. The worst case is presented in the edge of several textures types (more of 2) where the window can be classified indistinctly in anyone of them. With the objective of to enlarge the experimental work and to carry out a performance comparison of the

SOFM algorithm, a segmentation texture algorithm is used based on gabor functions such as it is described in [15], both algorithms were applied to Fig. 8 (*Top*), having as results Fig. 8 (*left*), for the algorithm based on gabor functions and Fig. 8 (*right*) for the SOFM algorithm.

The future work can be concentrated on two directions. First the problem of trying with the texture edges using an adaptive method that allows to change the window size and the second is to find a different discrimination method that considers other characteristic (different to cross-correlation) making in this way perhaps a robuster classification.

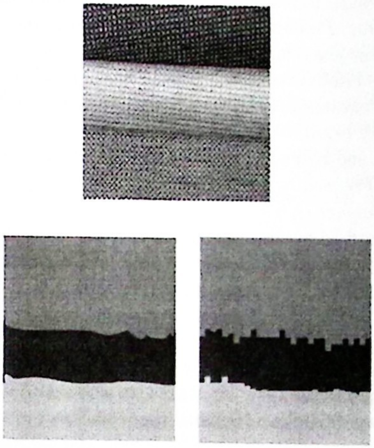


Fig. 8. (*Top*) Original image, (*left*) segmented image using gabor functions and (*right*) segmented image using the SOFM algorithm

References

[1] Haralick, R. M., 1967. Statistical and structural approaches to texture. *Proceedings of the IEEE*, pp. 786-804.

[2] Haralick R. M. et al., 1973. Texture features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 610-620.

[3] Weszka, J. C. et al., 1976. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 269-285.

[4] Ehrich, R. W. and Foith, J. P., 1978. A view of texture topology and texture description. *Computer Graphics and Image Processing*, vol. 8, pp. 174-202.

[5] O'Toole, R. K. and Stark, H., 1980. Comparative study of optical vs. all digital techniques in textural pattern recognition. *Applied Optics*, vol. 19, pp. 2496-2506.

[6] Rosenfeld, A. and Kak, A., 1982. *Digital Image Processing*, vol. I. Orlando, FL: Academic Press.
[7] Ballard, D. H. and Brown, C. M., 1982. *Computer Vision*. Englewood Cliffs, NJ: Prentice Hall.
[8] Coggins, J. M. and Jain, A. K., 1985. A spatial filtering approach to texture analysis. *Pattern Recognition Letters*, vol. 3, pp. 195-203.
[9] Daugman, J. G., 1989. Network for image analysis: Motion and texture. *Proceedings of the International Joint Conference on Neural Networks*, Washington DC, vol. I, pp. 189-193.
[10] Rao, N. and Vemuri, V., 1989. A neural network architecture for texture segmentation and labeling. *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, vol. I, pp. 127-133.
[11] Clark, M. et al., 1987. Texture segmentation using a class of narrow-band filters. *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, April, pp. 14.6.1-14.6.4.
[12] Kulkarni, A. D. and Byars, P., 1991a. Artificial neural network models for image understanding. *Proceedings of the SPIE Conference on Image Processing Algorithms andTechniques II*, San Jose, CA, vol. 1452, pp. 512-522.
[13] Grossberg, S. and Mingolla, E., 1988. A neural network architecture for preattentive vision: Multiple scale segmentation and regularization. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, pp. 177-184.
[14] Kohonen, T., 1988. *Self-Organization and Associative Memory*. Berlin: Springer-Verlag.
[15] D. Duna, Higgins W. E. and Wakeley J., Texture Segmentation using 2-D Gabor Elementary Functions, *IEEE, Tansactions on pattern análisis and machine intelligence*, February 1994 (Vol. 16, No. 2) pp. 130-149.