HIERARCHICAL ARCHITECTURE IMPLYING FUZZY AND NEURAL TECHNIQUES FOR ON-LINE GEOMETRIC SHAPE RECOGNITION

Ioan Z. MIHU*, Arpad GELLERT*, Cosmin N. SUCIU**

** "Lucian Blaga" University of Sibiu, Computer Science Department, str. Emil Cioran, nr. 4, Sibiu, ROMANIA, E-mail: ioan.z.mihu@ulbsibiu.ro, arpad.gellert@ulbsibiu.ro*

*** "S.C. Polisano S.R.L.", str. Moldoveanu, nr. 25, Sibiu, ROMANIA, E-mail: suciu_nicu@yahoo.com*

Abstract: This paper is focused on on-line geometric shape recognition (Ulgen, et al., 1999) based on fuzzy techniques and backpropagation neural algorithm. We propose a new method for geometric shape recognition that consists of a hierarchical architecture implying a fuzzy classifier of angles and a multilayer neural network for training and classification of geometric shapes. Before the effective classification an on-line feature extraction process is applied. Our method examines the geometric shape as a whole in a way similar to the human recognition process. In the recognition process we have to use information that is invariant in terms of scaling, translation and rotation. The internal angles represent the relevant information relatively to the geometric shape. The key concept is that the neural network learns the internal angles of a shape. The optimal configuration of the neural network is determined based on the criterion of the performances' maximization. The optimal fuzzy classifier is chosen based on the same criterion.

Keywords: Geometric Shape Recognition, Neural Networks, Neural Shape Classification, Fuzzy Systems, Backpropagation Algorithm, Multi-Layer Perceptron

1. INTRODUCTION

The artificial neural networks are composed of a multitude of neurons, simple processing elements that operate in parallel. A great advantage of the artificial neural networks is their capacity to learn on examples. In order to solve a problem traditionally, we have to elaborate its model, and after that we have to indicate a succession of operations that represents the solving algorithm of the problem. However there are practical problems with a high level of complexity, and for this kind of problems it is very hard or even impossible to establish an algorithm.

In the connection models we are not forced to give a solving algorithm of a problem to the neural network, we have to offer him only a multitude of consistent examples. The network extracts the information from the training samples, in this way it is able to synthesize implicitly a certain model of the problem. In other words, the neural network builds up alone an algorithm to solve a problem. The capacity of the neural network to solve complex practical problems using a multitude of samples gives them a highly large potential of applicability. Building intelligent systems that can model human behavior has captured the attention of the world for years. So, it is not surprising that a technology such as neural networks has generated great interest.

Ulgen, *et al.* (1999), in their work implemented a geometric shape classifier and they have used a neural network with binary synaptic weights (BSW). The BSW algorithm, which was implemented on a three layer network, determines the thresholds for the hidden and output layer nodes and the weights of the synaptic links between the layers, in addition to the number of hidden layer nodes in one feed-forward pass. The main concept of the algorithm can be explained as the separation of globally intermingled patterns within an n-dimensional space through the formation of hyperplanes that separate different classes of patterns at a local region in the space.

In this work we use a multilayer feedforward neural network to recognize the basic geometric shapes such as circles, rectangles or triangles. We will choose the best configuration of the neural network based on the results obtained with our test images.

2. THE MULTILAYER FEEDFORWARD NEURAL NETWORK

This section introduces the backpropagation learning algorithm addressed by the architecture presented in this paper. More detailed descriptions can be found in classic introductory books (Hertz, et al., 1991). The typical artificial neuron model represents a device with *n* inputs and a single output. The output y_i of the *i*-th neuron of the network is computed as:

$$y_i = \sigma(p_i) = \sigma\left(\sum_{j=1}^{n} W_{i,j} \cdot x_j\right) \qquad (1)$$

for $i = 1, 2, \ldots, m$;

where $W_{i,j}$ represents a coefficient or synaptic weight associated with the *j*-th input x_j and the *i*-th neuron. The weighted sum p_i is called potential. A typical node is depicted in figure 1.

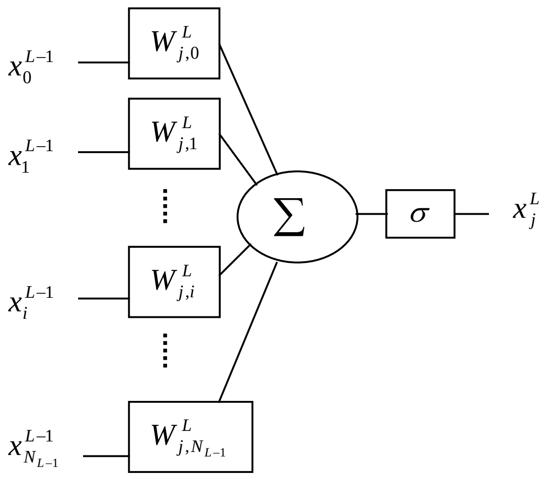

Figure 1. A Perceptron node

The nonlinear activation function σ in this case is the sigmoid, and the network is trained using the gradient descent method known as backpropagation.

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

Equation (1) can be rewritten in a matrix form as

$$\vec{y} = \sigma(\vec{p}) = \sigma(W \cdot \vec{x}) \qquad (3)$$

Usually, the activation function σ represents some saturating non-linear function. Neurons are often organized in layers, all neurons in a layer sharing the same inputs and having their outputs connected to the inputs of the next layer. The weight matrixes are then shown as $W^{[q]}$, where q is the layer number.

Neural networks usually undergo a learning process. The synaptic-weight matrixes are iteratively updated according to a learning rule. One of the simplest one is the *Hebb* rule:

$$W = W + \alpha \cdot (\vec{y} \cdot \vec{x}^{T}); \qquad (4)$$

where α is a learning factor. Though this rule is seldom used as stated, most of the commonly used learning rules are slight modifications of equation (4).

Multilayer neural networks are used for pattern classification, pattern matching, and function approximation. By adding a continuously differentiable function, such as Gaussian or sigmoid function, it is possible for the network to learn practically any nonlinear mapping to any desired degree of accuracy. There are several ways that multilayer neural networks can have their connection weights adjusted to learn mappings. The most popular technique is the backpropagation algorithm and its many variants.

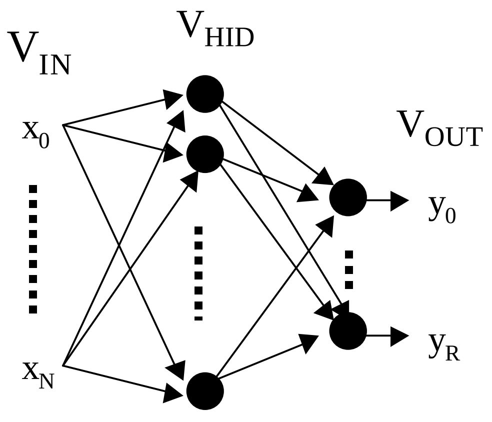

Figure 2. A multilayer perceptron with 2 active layers (one hidden layer).

Multilayer networks make it possible to implement any arbitrary function $\vec{y} = \Phi(\vec{x})$, $\vec{x}$ being the input of the first layer and $\vec{y} = \vec{y}^{[L]}$ representing the output of the last layer L. Often the activation function σ is a hyperbolic tangent. The function Φ is learned by repeated presentation of input-output pairs $\{\vec{x}, \vec{d}\}$, called prototypes. The backpropagation(BP) learning rule is a gradient-descent algorithm that updates the weights to minimize the square-error on the learning prototypes. For that purpose an error signal is computed for each layer (Zurada 1992):

$$\delta_i^{[L]} = \left(d_i - y_i^{[L]}\right) \cdot \sigma'\left(p_i^{[L]}\right) \tag{5}$$

$$\delta_i^{[q]} = \left(\sum_{k=1}^{m_{q+1}} W_{k,i}^{[q+1]} \cdot \delta_k^{[q+1]} \right) \cdot \sigma'\left(p_i^{[q]}\right) \tag{6}$$

for q = 1, 2, …, L-1;

where $\sigma'(v) = \dfrac{d\sigma(v)}{dv}$.

The equations (5) and (6) are valid for all the i neurons (i = 1, 2, …, m_q) of layer q. Once the errors have been back-propagated, the weights are updated as:

$$W^{[q]} = W^{[q]} + \alpha \cdot \vec{\delta}^{[q]} \cdot \vec{y}^{[q-1]T} \tag{7}$$

for q = 1, 2, …, L,   where $\vec{y}^{[0]} = \vec{x}$.

3. GEOMETRIC SHAPE RECOGNITION

The classical techniques based on shape partitioning into segments, followed by a syntactical analysis to match with a predefined shape, are strongly affected by noise and are weak in terms of generalization. In order to eliminate these limitations of the classical methods, our method examines the geometric shape as a whole in a way similar to the human recognition process. Human beings recognize such basic shapes regardless of the variations in size, noise on the shape border, translation, rotation, and in the case of triangles, regardless of the type of the triangle. That means that not the segments are important in the recognition process but the angles, which represent the relevant information relatively to the geometric shape. The key concept is that the neural network learns the internal angles of a shape (the angles between any two consecutive tangent vectors). As a consequence, the neural network training process will be simplified, therefore only a few training samples that represent a class of shapes (i.e. triangles, rectangles and elliptic shapes) are sufficient. Our application's aim is to recognize the basic geometric shapes (elliptic, rectangular and triangular).

3.1. Feature Extraction

The purpose of preprocessing is to create an intermediate representation of the input data and it is performed on-line (prior to the application of recognition task). The preprocessing step can be defined as a feature extraction process that is important since it prepares input data that is invariant in terms of scaling, translation and rotation. The feature extraction is performed on the captured points along the boundary of the shape.

Since the geometric shapes are hand-drawn using the mouse, the information could include noise due to the variations in capture speed of the mouse and erratic hand motion while drawing. We have to extract the features of the shape and to eliminate the noise appeared while drawing, keeping only the essential characteristics. Also the hand-drawn shape may contain interruptions that must be eliminated unifying the segments. The feature extraction process is composed of a number of steps, and the first of them is the calculation of the shape's weight center.

Calculation of the shape weight center. For the calculation of the weight center of a given geometric shape the next formulas are used:

$$x_C = \frac{\sum_{i=0}^{n-1} x_i}{n}, \qquad y_C = \frac{\sum_{i=0}^{n-1} y_i}{n}, \tag{8}$$

where x_c is the horizontal position of the shape's weight center, y_c is the vertical position of the shape's weight center, *n* is the number of captured points while drawing, x_i is the horizontal position of each captured point, and y_i is the vertical position of each captured point.

Extraction of significant points. The next step in the feature extraction process is the determination of the sample points. There are calculated *n* angularly equispaced vectors that start from the shape's weight center; *n* is the number of sample points. The intersection of these vectors with the boundary of the shape represents the sample points of that shape. The next step consists in the calculation and the tracing out of the tangent vectors to the shape in these points. In our application the tangent vectors are obtained by the union of the sample points; unifying two successive sample points a tangent vector is obtained.

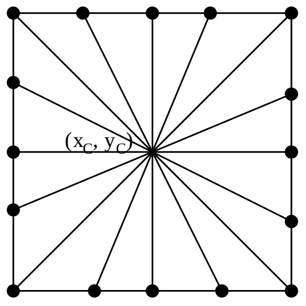

Figure 3. Extraction of sample points (the intersections of the vectors with the shape's boundary).

As we can see in figure 3, a very important parameter in the recognizing process is the number of sample points. For an efficient extraction of the relevant information necessary for the recognition process of the geometric shape, we have to use a sufficient number of sample points.

3.2. On-line Data Acquisition and Processing

When we want to create a new geometric shape to be processed by the recognition system, we have to draw it with the mouse on the application's frame. The drawing process can be interrupted anytime and resumed to complete the shape (that is the shape can be built by drawing multiple segments).

The on-line data acquisition and processing consists of the following steps:

- capturing successive points on the shape's boundary;
- extracting significant points;
- calculating the angles between consecutive segments.

Capturing successive points on the shape's boundary. The first stage in obtaining the points from the shape's boundary is very simple because the positions of the mouse during the drawing process are memorized. When the positions of the captured points are memorized, the shape's weight center is on-line calculated, too (see chapter 3.1.). The capturing process is based on the *drag and drop* event handled by the operating system. The event generation frequency is constant on a certain computer but dependent on hardware platform. On the other hand, the drawing speed is different from user to user or even from instance to instance of the same user. On the *drag and drop* event there are captured a number of points that are not neighbors in the real shape's boundary and that depends on the drawing speed (the number of points is greater if the drawing speed is low, and it is smaller in the case of a higher drawing speed). When the application was running on a modest PC computer (P200/64MB RAM), the user had to draw with a low speed for a sufficient number of points to be captured. In the case of a high drawing speed, using such a computer, the small number of captured points drove sometimes to major differences between the shapes obtained by the capturing process and the real shapes.

Thus, if on the application's frame are painted only the captured points obtained on the *drag and drop* event, a discontinuous copy of the real shape results. Visually the problem was solved tracing segments between any two consecutive captured points. Only the visual interface of the application was solved in this way. The second problem to be solved consists of finding the intersection points between the shape's boundary and the angularly equispaced vectors starting from the shape's weight center (see chapter 3.2.). There is the possibility that some of the angularly equispaced vectors don't intersect with any of the captured points. In this case the extraction of significant points cannot be made and also the geometric shape cannot be approximated.

To solve this problem we used an algorithm, which calculates and stores all the points between any two consecutive captured points obtained on the *drag and drop* event (figure 4).

There are two cases:

- If the angle between the segment and the horizontal axes is greater than 45 degrees, the straight line's equation is the following:

for i=1 to $|y_2 - y_1|$

$$y = y + i \tag{9}$$

$$x = x_1 + |y - y_1| \cdot \frac{|x_2 - x_1|}{|y_2 - y_1|} \tag{10}$$

- If the angle between the segment and the horizontal axes is less or equal to 45 degrees, the straight line's equation becomes:

for i=1 to $|x_2 - x_1|$

$$x = x + i \tag{11}$$

$$y = y_1 + |x - x_1| \cdot \frac{|y_2 - y_1|}{|x_2 - x_1|} \tag{12}$$

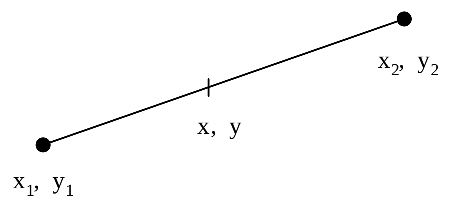

Figure 4. Calculating of the intermediate points between two consecutive captured points.

After all the points between any two consecutive captured points are calculated, they are stored in a list. This list contains all the points of the geometric shape's boundary in their drawing order.

Extracting significant point. As it was described in chapter 3.1, there are calculated n angularly equispaced vectors starting from the shape's weight center (n being the number of significant points – a parameter which will be chosen in order to maximize the application's performance). The angular distance between any two consecutive vectors is:

$$dist = \frac{360}{n} \tag{13}$$

The intersection of these vectors with the shape's boundary represents the significant points of that shape (for n vectors there will be n significant points). By tracing line segments between any two significant points an approximation of the geometric shape is obtained (figure 5).

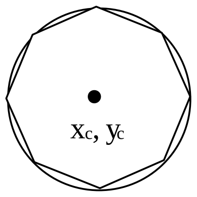

Figure 5. Approximation of a circle using 8 significant points.

Calculating the angles between consecutive segments. The angle between two consecutive segments is determined by calculating the angles between each segment and the horizontal axes (figure 6). The angles between segments belong to the [0, 360) interval, it is so large because of the erratic hand motion during the drawing process. So, the angle is calculated depending on the quadrant (0-90, 90-270,270-360) it belongs to.

In figure 6 for example, the angles between the two consecutive segments and the horizontal axes belong to the first quadrant:

$$\alpha_1,\, \alpha_2 \in [0,90) \Rightarrow \alpha = 180 - |\alpha_1 - \alpha_2| \quad (14)$$

where α_1, α_2 are the angles between the segments and the horizontal axes, and α is the angle between the two segments (one of the interior angles of the geometric shape).

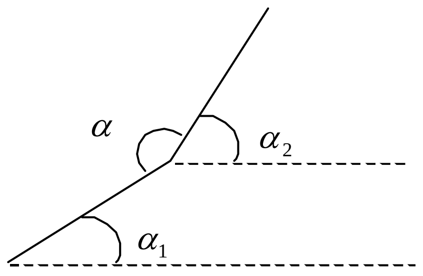

Figure 6. Calculation of the angle between two consecutive segments, with $\alpha_1,\, \alpha_2 \in [0,90)$.

3.3. Fuzzy Classification

To generate the input data for the neural network, after the feature extraction process follows the adaptation of the obtained information. The internal angles of a geometric shape offer the relevant information necessary to the classification process. The angles between the consecutive tangent vectors are calculated and we obtain n angles, which will be classified into four categories (fuzzy). Each angle will receive a membership value depending on the category to which it belongs, as it follows:

- 2 for the angles less than 75 degrees;
- 3 for angles between 75 and 135 degrees;
- 1 for angles between 135 and 150 degrees;
- 0 for the angles greater than 150 degrees;

The membership values must be given in such a way that, after the addition of the membership values according to the n angles, to obtain different sums for each class of geometric shape (i. e. triangle, rectangle, circle). We have considered that the important angles are the angles less than 150 degrees.

In the case of a rectangle or a triangle, along the sides we will have angles near to 180 degrees; because these angles are not significant, they receive 0 as membership value, in other words these angles will not contribute to the sum. Since the number of angles less than 150 degrees offers the relevant information necessary to the recognition process of the basic geometric shapes, only these angles, through their consistent membership values, will contribute to this sum, which will be a value from the interval [0, 3n]. Using the sum of the angles' membership values the dimensions of the shape don't matter (there is no difference between a little triangle and a big one), and not even the dimensions of the sides (there is no difference between a square and a rectangle), only the internal angles matter.

3.4. Neural Recognition of the Shape

The neural network's architecture and the learning algorithm used were presented in section 2. The input vector for the neural network will be obtained after the serial coding of the sum of the membership values according to the internal angles of a given geometric shape. In this way, the sum's value determines the number of bits on "1" in the serial code, and the rest of bits are "0".

The neural network is statically trained before its effective use. That means that the network will be trained using a set of prototypes (a number of representative learning shapes). Before starting the training process the weights are randomly initialized. During the training process, if the shape is correctly classified, only a backward step is made. If the shape is incorrectly classified, the backward step will be repeated until the classification becomes correct and one more time after that.

For its effective use, the neural network is initialized with the weights generated by the static training process. During the effective run-time classification process only the forward step is performed.

The dimension of the neural network's input vector must be calculated taking into consideration the most disadvantageous case that appears when all the angles takes part of the category 3 (angles between 75 and 135 degrees). In this case the calculated sum will have the maximum value (3n), and therefore we need 3n neurons in the input layer of the neural network. Consequently the neural network's input vectors are sequences of 3n binary values.

Since the neural network must recognize three categories of shapes (rectangles, triangles and circles), in the output layer we will have three neurons, one for each category. The neuron with the highest output value will win, specifying the category in which the shape takes part. Since the neural network used in this work has three layers, the dimension of the hidden layer represents a parameter and it's value will be established based on the criterion of the performances' maximization. We will vary in the next chapter the number of neurons from this layer; we want to obtain in this way the best configuration of the neural network.

4. EXPERIMENTAL RESULTS

The neural network was statically trained with 10 learning shapes for each shape category. After the learning process the recognition system was evaluated using 30 test shapes for each category. In this chapter a number of architectural parameters are varied and the obtained results are presented.

As we specified in chapter 3.1, in the feature extraction process we have to use a number of sample points as great as possible and in this way we can extract efficiently the relevant information necessary for the recognition process of the geometric shape. But if we use too many sample points there is a risk of appearance of the noise in the extracted information. Usually the noise appears because of the undesirable hand movements while drawing with the mouse. Therefore, the number of sample points represents another parameter that must be chosen based on the criterion of the performances' maximization. We synthesize in the table 1 the influence of this parameter on the performances of the geometric shape recognition system:

Table 1. The influence of the number of sample points on the shape classification.

Shape [%]	16	32	48	64
Circles	63,33	100	96,66	96,66
Triangles	56,66	60	60	80
Rectangles	86,66	70	63,33	26,66
All shapes	68,88	76,66	73,33	67,77

We can see that the best number of sample points is 32.

The internal angles of the shapes are classified into four categories in the fuzzyfication stage. We showed at 3.3 that the little angles are the most important in the classification process. In chapter 3.3 we have also presented a variant of fuzzyfication. The method of according the membership values represents the third parameter that influences the performances of the recognition system. We studied four methods of fuzzyfication and, depending on the results, we chose the best solution in order to maximize the application performance. In the same time we have decreased the training time of the neural network from 10000 iterations to 1000. In table 2 are presented the obtained results.

Table 2. Different methods of fuzzyfication of the internal angles

Shape [%]	I	II	III	IV
Circles	100	96,66	100	100
Triangles	23,33	0	76,66	73,33
Rectangles	96,66	100	90	100
All shapes	73,33	65,55	88,88	91,11

The four fuzzyfication methods used in this work have accorded membership values to the internal angles as follows:

Method I:
- 2 for the angles less than 75 degrees;
- 3 for angles between 75 and 135 degrees;
- 1 for angles between 135 and 150 degrees;
- 0 for the angles greater than 150 degrees.

Method II:
- 3 for angles less than 90 degrees;
- 2 for angles between 90 and 120 degrees;
- 1 for angles between 120 and 150 degrees;
- 0 for the angles greater than 150 degrees.

Method III:
- 4 for angles less then 90 degrees;
- 2 for angles between 90 and 120 degrees;
- 1 for angles between 120 and 150 degrees;
- 0 for the angles greater than 150 degrees.

Method IV:
- 5 for angles less then 90 degrees;
- 2 for angles between 90 and 120 degrees;
- 1 for angles between 120 and 150 degrees;
- 0 for the angles greater than 150 degrees.

The obtained results show that the fourth method of fuzzyfication represents the best solution.

We have continued our work by studying the influence of the neural network's architecture on the performances of the recognition system. We varied the number of neurons in the hidden layer (the first parameter) and we evaluated the recognizing rate for three different dimensions of the hidden layer: 5, 10 and 20 neurons. The obtained results are presented in table 3.

Table 3. The influence of the number of neurons from the hidden layer on the efficiency of the recognition system.

Shape [%]	5	10	20
Circles	96.66	100	100
Triangles	23.33	73.33	73.33
Rectangles	100	100	100
All shapes	73.33	91.11	91.11

We can observe that the best solution is to use ten neurons in the hidden layer. If we increase the number of neurons in this layer over ten, the efficiency of the neural network doesn't change, but the training time grows up exponentially.

5. CONCLUSION

In this work we presented a method of recognizing the basic geometric shapes. Both training and recognition process are made by extracting the features from the training (test) samples, and by classifying the internal angles of the shape. The information obtained after the fuzzyfication process is used as inputs for the multilayer feedforward neural network. The network learns the three classes of geometric shapes by their internal angles; the values of the internal angles are invariant in terms of scaling, translation and rotation.

We studied the influence of the number of significant points used in the feature extraction process on the efficiency of the recognition system. A small number of significant points (less than 32) is not sufficient for a correct recognition of the geometric shapes. In this case the information obtained in the feature extraction process is not sufficiently consistent to assure the desired performances of the recognition system. On the other hand if we use too many significant points (over 32), there is a risk of appearance of the noise in the extracted information. The noise, which usually appears because of the undesirable hand movements while drawing with the mouse, overlaps the relevant information and in this way it degrades the performances of the recognition system. The evaluations drove us to the conclusion that the optimal number of significant points is 32.

After that, we studied different fuzzyfication methods of the internal angles. The effected tests drove us to the conclusion that the little angles (less than 90 degrees) are very important, because the number of these angles offers the relevant information necessary to the recognition system.

Finally we studied the influence of the neurons' number from the hidden layer on the efficiency of the neural network used in the recognition process. The tests show that the best solution for the number of neurons in this layer is ten. In our opinion the results show acceptable classification accuracy of 91.11% obtained on our test images. This means that 91.11% of the hand-drawn shapes (test images) were correctly classified.

An unsolved problem appeared in the feature extraction process. If the obtained significant points avoid the corners of the shape, a relevant internal angle is lost and it is replaced by other two successive angles.

As you can see in figure 7, one corner of the triangle was replaced by two other corners and in this way, because the shape has four significant internal angles, it is possible that the recognition system will classify it as rectangle. In the same way, in figure 8, each significant angle is replaced with other two greater angles and the recognition system could classify the rectangle as being a circle.

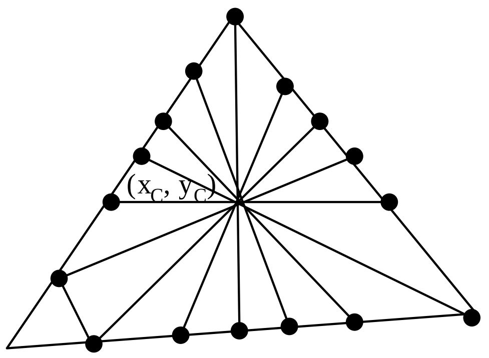

Figure 7. Extraction of sample points. One corner of the triangle is missed.

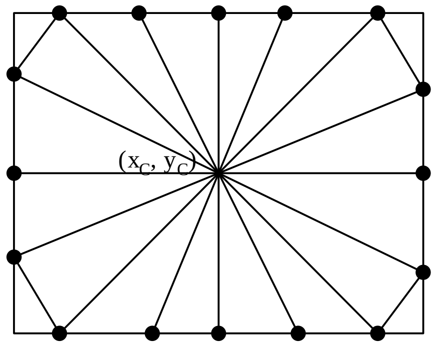

Figure 8. Extraction of sample points. All the corners of the shape are missed.

In other words a part of the relevant information is lost and in addition other information (noise) appears, which can lead to wrong classification of the shapes. To eliminate these deficiencies, the sample points must include also the corners of the shape (if the shape has corners). This is one of the development directions of the recognition system presented in this work. Another development direction consists on increasing the variety of geometric shapes, which the system can recognize.

REFERENCES

Comon P., Voz J. L. and Verleysen M. (1994) *Estimation of performance bounds in supervised classification*, In M. Verleysen, editor, ESANN: European Symposium on Artificial Neural Networks, Bruxelles, pages. 37 - 42.

Davis L. S. (1977), *Understanding shape: angles and sides*, IEEE Trans. on Computers, vol. C-26, pages 125-132.

Fukunuga K. (1986) *Statistical pattern classification*, Handbook of Pattern Recognition and Image Proc., San Diego, CA: Academic Press, pages 3-32.

Funahashi K. I. (1989) *On the approximate realization of continuous mappings by neural networks*, Neural Networks, Vol. 2, pages 183-192.

Guyon I. (1990) *Neural networks and applications*, Internal Report AT & T Bell Labs.

Hertz J., Krogh A., and Palmer R. (1991) *Introduction to the Theory of Neural Computation*, Santa Fe Institute Studies in Sciences of Complexity, Addison-Wesley, Redwood City, California.

Hornik F. (1989) *Multilayer feedforward networks are universal approximators*, Neural Networks, Vol. 2, pages 359-363.

Khotanzad A., Lu J. (1990) *Classification of invariant image representations using a neural network*, IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 38, pages 214-222.

Kosko B. (1992) *Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*, Prentice Hall.

Mihu Z. I., Gellert A. and Suciu C. N. (2003) *Geometric shape recognition using fuzzy and neural techniques*, In Proceedings of the 11th International Scientific Symposium SINTES 11, pages 354 – 358, Craiova.

Montas J. (1987) *Methodologies in pattern recognition and image analysis - a brief survey*, Pattern Recognition, vol. 20, pages 1-6.

Perantonis S. J., Lisboa P. J. G. (1992) *Translation, rotation, scale invariant pattern recognition by higher-order neural networks and moment classifiers*, IEEE Trans. on Neural Networks, vol. 3, pages 243-251

Schalkoff R.J. (1997) *Artificial Neural Networks*, McGraw-Hill.

Ulgen F., Akamatsu N. and Fukumi M. (1999) *On-line shape recognition with incremental training using a neural network with binary synaptic weights*, Industrial Applications of NNs, CRC Press, pages 159-192;

Zurada J.M. (1992) *Introduction to Artificial Neural Systems*, West Publishing Company, St. Paul.