F.S.A.I., Vol. 11, No. 1-3, pp. 61-71, 2005

# Improving the Recognition System Architecture in Order to Increase the Set of Recognized Geometric Shapes

**Ioan Z. Mihu, Arpad Gellert, Horia V. Caprita**

"Lucian Blaga" University of Sibiu, Computer Science Department, str. Emil Cioran, nr. 4, Sibiu, 550025, ROMANIA, E-mail: ioan.z.mihu@ulbsibiu.ro, arpad.gellert@ulbsibiu.ro, horia.caprita@ulbsibiu.ro

**Abstract**. We examine the effect of shape set extension, over the architectural parameters of a hierarchical system dedicated to recognize the hand-drawn geometric shapes. Such a recognition system is helpful in drawing packages and automated sketch entry in handheld computers. Our method examines the geometric shape as a whole in a way similar to the human recognition process, using information that is invariant in terms of scaling, translation and rotation. The recognition system presented in [5] is able to recognize three hand-drawn shapes: circle, triangle and rectangle. In this paper we propose an improving strategy of the hierarchical architecture presented in [5], in order to increase the number of geometric shapes to recognize. The recognition system utilizes on the first level a fuzzy function for filtering the angular differences along the shape boundaries, and on the second level a multilayer feed-forward neural network for classification. The architectural improvements proposed in this paper consist of both, adapting the fuzzification process and the neural network architecture in order to increase the number of shapes to recognize. Extending the number of categories leads unfortunately to the degradation of the recognition accuracy. The architectural improvements must compensate this degradation and maintain the recognition accuracy at a satisfactory level. The study presented in this paper analyses all these emphasized aspects when one more category is added: ellipse.

**Keywords:** geometric shape recognition, neural networks, neural shape classification, fuzzy systems

## 1 Introduction

The capacity of the neural networks to solve complex problems learning by examples, 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., in their work [7], 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 our previous work [5], we used a multilayer feedforward neural network to recognize the basic geometric shapes such as circles, rectangles or triangles. We chose the best configuration of the neural network based on the results obtained with our test images.

In this paper we propose to increase the number of geometric shapes to be recognized by the two-level hierarchical architecture, presented in our previous work and to adapt the architecture of the recognition system for the new goal. We extend the shape set from three to four by adding a new category (ellipse) and we study the implications that this set extension bring about the recognition system architecture.

## 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 [3]. 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:

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

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. The nonlinear activation function $\sigma$ in this case is the sigmoid,

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

and the network is trained using the gradient descent method known as backpropagation. Equation (1) can be rewritten in a matrix form as

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

Usually, the activation function $\sigma$ 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:

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

where $\alpha$ 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.

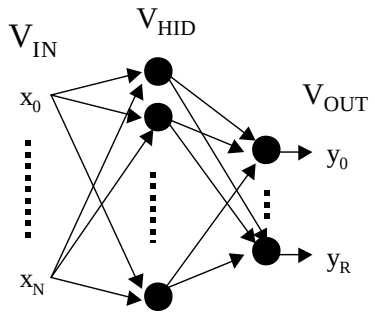



Fig. 1. A multilayer perceptron with 2 active layers (one hidden layer).

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 $\sigma$ is a hyperbolic tangent. The function $\Phi$ 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 [8]:

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

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

for $q = 1, 2, \ldots, L\text{-}1$, where $\sigma'(v) = \dfrac{d\sigma(v)}{dv}$.

The equations (5) and (6) are valid for all the i neurons ($i = 1, 2, \ldots, m_q$) 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 $q = 1, 2, \ldots, L,$ where $\vec{y}^{[0]} = \vec{x}$.

## 3 Adapting The Recognition System's Architecture

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. 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. As a consequence, the neural network training process will be simplified, therefore only a few training samples that represent a class of shapes are sufficient. Our application's aim is to recognize the basic geometric shapes (elliptic, rectangular and triangular shapes).

### 3.1 On-line Feature Extraction Process

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 on-line feature extraction consists of the following steps:

- capturing successive points on the shape's boundary;
- calculating the shape's weight center;
- extracting significant points;
- detecting the corners of the shape;

- computing the angles between consecutive segments.

*Capturing successive points on the shape's boundary*. The point 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. 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. 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.

*Calculating of the shape's 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, xi is the horizontal position of each captured point, and $y_i$ is the vertical position of each captured point.

*Extracting significant points.* The next step in the feature extraction process is the determination of the significant points. There are calculated n angularly equispaced vectors that start from the shape's weight center; n is the number of sample points. The angular distance between any two consecutive vectors is:

$$dist = \frac{360}{n} \tag{9}$$

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 2).

As it can be seen in figure 2, an 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.

*Detecting the shape's corners.* The significant points often avoid the shape's corners, and in this situation, a relevant internal angle is replaced by other two

successive angles. In figure 3 one corner of the triangle is replaced by two other corners and in this way, because the shape has four significant internal angles, the recognition system misclassifies it as rectangle. In other words a part of the relevant information is lost and in addition other information (noise) appears, which often leads to wrong classification of the shapes. We implemented an algorithm, which eliminates these deficiencies, and improves the classification accuracy.

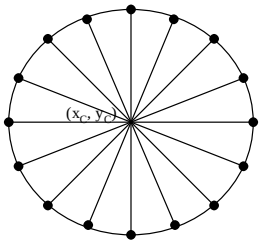



Fig. 2. Extraction of sample points (the intersections of the vectors with the shape's boundary).

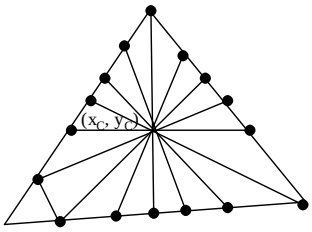



Fig. 3. Extraction of significant points. One corner of the triangle was avoided.

The corner-detection algorithm calculates the distances from the points on the real shape to the straight segment defined by two consecutive significant points (figure 4). The distances are calculated only for the points which are situated between those two significant points on the shape's boundary. When all distances are less than a certain threshold ($T$), we consider that the shape's approximation is correct. Our experiments showed that a threshold value of 2 leads to best experimental results. If one of the distances exceeds the threshold value, the maximum distance is calculated and one of the two significant points is replaced with the point situated at the maximum distance from the line segment. This is illustrated in figure 4, where the significant point ($x_2$, $y_2$) will be replaced with the real corner of the shape ($x$, $y$).

The distance between the current point ($x$, $y$) from the shape and the straight segment (($x_1$, $y_1$), ($x_2$, $y_2$)), is calculated using the following equation:

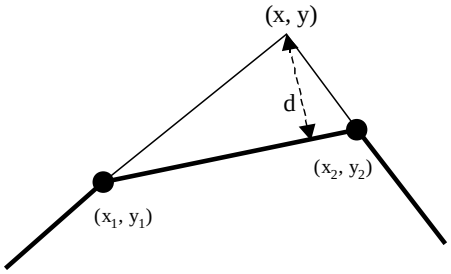
$$d = \frac{ax + by + c}{\sqrt{a^2 + b^2}}, \tag{10}$$

where $a = y_2 - y_1$, $b = x_1 - x_2$, and $c = x_2 y_1 - x_1 y_2$.

*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 5). The angle is calculated depending on the quadrant it belongs to. In figure 5 for example, the angles between the two consecutive segments and the horizontal axes belong to the first quadrant:

$$\alpha_1, \alpha_2 \in [0,90) \Rightarrow \alpha = 180 - |\alpha_1 - \alpha_2| \tag{11}$$

where $\alpha_1$, $\alpha_2$ are the angles between the segments and the horizontal axes, and $\alpha$ is the angle between the two segments.



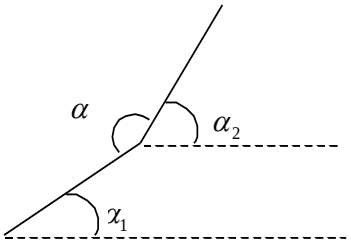




Fig. 4. Detection of the shape's corners Fig. 5. Calculation of the angle between two consecutive segments, with $\alpha_1, \alpha_2 \in [0,90)$.

### 3.2 Adapting the Fuzzy Classification Process

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 receives 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 110 degrees;
- 1 for angles between 110 and 140 degrees;
- 0 for the angles greater than 140 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. We have determined that the important angles are the angles less than 140 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 140 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, 3*n*]. 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.

Adding new categories to be recognized, the fuzzification process must be adjusted. That is, new categories require more fuzzy classes, in order to assure a

powerful discrimination between all shape categories. Increasing the number of categories to recognize, the recognition rate normally lowers. The adjustment of the fuzzification process must be done to compensate the performance degradation of the recognition system. In this study we analyzed the impact over the recognition system performance produced by adding only one new shape category. This is the reason for that the fuzzification process required only minor adjustments. Anyway, the higher number of categories to be recognized, the bigger adjustments in the fuzzification process must be done.

### 3.3 Adapting the Neural Network's Architecture

The neural network's architecture and the learning algorithm used were presented in section II. 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.

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 input vector must be calculated taking into consideration the most disadvantageous case that appears when all the angles takes part of the category with membership value of 3 (angles between 75 and 110 degrees). In this case the calculated sum will have the maximum value (3n), and therefore we need 3*n* neurons in the input layer of the neural network. Consequently the neural network's input vectors are sequences of 3n binary values.

In the output layer of the neural network we allocate one neuron for each category. The neuron with the highest output value will win, specifying the category in which the shape takes part. Increasing the number of shape categories, for each new category we have to add a new neuron in the output layer. In the present stage of our work only one new category was introduced, the ellipse, and as consequence, the total number of shape categories (circle, ellipse, triangle and rectangle), and respectively the number of output layer neurons became four. Since the neural network used in this work has three layers, the dimension of the hidden layer represents a parameter; its value will be established based on the criterion of the performance maximization. For a significantly higher number of categories, we consider that more attention must be paid to optimize the neural network architecture. The improvement might be performed by increasing the number of

neurons in the hidden layer and/or the number of hidden layers, for a more powerful codification of the categories through higher number of parameters.

## 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. As we specified in section III, 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 a parameter that must be chosen based on the criterion of the performances' maximization. In our previous works [5], we determined through experiments that the optimal number of significant points is 32.

The internal angles of the shapes are classified into four categories in the fuzzification stage. We showed in section III, that the little angles are the most important in the classification process. We have also presented an optimal variant of fuzzification, which led to the best results in our previous works, too.

In previous works [5], we also studied the influence of the neural network's architecture on the performances of the recognition system. The results show that the optimal solution is to use only one hidden layer with ten neurons. The evaluation results obtained for three shape categories are presented in table 1.

In this paper we continue our study, analyzing the impact over the recognition system's performance, produced by extending the set of prototypes to be recognized. For the beginning, we added only one new prototype, the ellipse. As we expected, increasing the number of shape categories, the recognition rate normally lowers. To compensate the performance degradation of the recognition system, it was necessary to adapt the system architecture to the new task, more complex than the previous one. The experimental results obtained for four shape categories, are presented in table 2.

*Table1.*

The classification accuracy of the recognition system for three shape *categories*

| Shape category | Classification accuracy [%] |
|---|---|
| Circles | 96.66 |
| Triangles | 100 |
| Rectangles | 96.66 |
| All shapes | 97.77 |

*Table 2*

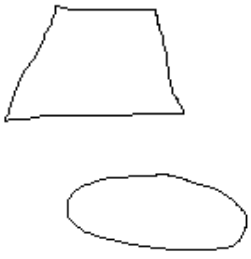
The classification accuracy of the recognition system for four shape *categories*

| Shape category | Classification accuracy [%] |
|---|---|
| Circles | 100 |
| Triangles | 100 |
| Rectangles | 96.66 |
| Ellipses | 80 |
| All shapes | 94.16 |

## 5 Conclusions

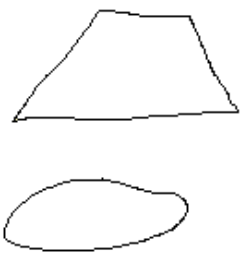
In this paper, we examine the effect of shape set extension, over the architectural parameters of a hierarchical system dedicated to recognize the hand-drawn geometric shapes. The recognition system is organized on two levels; the first one is represented by a fuzzification process and the second one, consists of a feedforward neural network. 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 fuzzification process is used as inputs for the multilayer feedforward neural network. The network learns the geometric shape classes by their internal angles. The values of the internal angles are invariant in terms of scaling, translation and rotation.

We analyzed the impact of adding a new category to the prototype set, over the recognition system performance. The new prototype is the ellipse, which is very close to an already existing one, the circle. As we expected, the extension of the prototype set produced a degradation of the recognition system performances. In order to compensate the recognition accuracy degradation, it was necessary to adjust the recognition system architecture. For this first extension step the major adjustments were made in the neural network architecture. For three shape categories we obtained in [5], a global recognition accuracy of 97.77%. Adding one new category (ellipse), which is very close to an already existing one (the circle), the global recognition rate lows to 94.16%.



a.) Distorted trapezium b.) Distorted ellipse

Fig. 6. Recognized shapes.



a) Distorted trapezium b.)Distorted ellipse

Fig. 7. Unrecognized shapes.

For a significantly higher number of categories, we consider that the neural network will require major architectural adjustments, including a higher number of hidden layer neurons, or even a higher number of hidden layers, in order to obtain a more powerful codification of the shapes' characteristics in the extended prototype set. In that case, also the fuzzification process must be significantly adjusted by adding new fuzzy classes to properly cover the extended set of prototypes.

Our system works properly and recognizes even the distorted shapes, neglectfully drawn by the user. Figure 6 presents neglectfully drawn shapes recognized by our system and figure 7 presents strongly distorted shapes, which can't be recognized.

One of the future developments of the recognition system will consist in architectural adjustments, in order to improve the recognition system performances. Our intention is to add new relevant information in the input vector of the neural network in order to obtain a better separation of the prototypes in this new extended input space. This better separation will permit also a significant extension of the shape set without an unacceptable performance degradation. This architectural improvements leads, however, to an increased volume of computations for the recognition process. That is, the architectural improvements must be done preserving the on-line characteristic of the application.

## References


[1] L. S. DAVIS: *Understanding shape: angles and sides*, IEEE Trans. on Computers, vol. **C-26**, pp. 125-132, 1977.

[2] I. GUYON: *Neural networks and applications*, Internal Report AT & T Bell Labs, 1990.

[3] J. HERTZ, A. KROGH, R. PALMER: *Introduction to the Theory of Neural Computation, Santa Fe Institute Studies in Sciences of Complexity*, Addison-Wesley, Redwood City, California, 1991.

[4] F. HORNIK: *Multilayer feedforward networks are universal approximators*, Neural Networks, vol.**2**, pp. 359-363, 1989.

[5] I. Z. MIHU, A. GELLERT, C. N. SUCIU: *Geometric shape recognition using fuzzy and neural techniques*, In Proceedings of the 11th International Scientific Symposium SINTES 11, pp. 354 – 358, Craiova, 2003.

[6] R. J. SCHALKOFF: *Artificial Neural Networks*, McGraw-Hill, 1997.

[7] F. ULGEN, N. AKAMATSU AND M. FUKUMI: *On-line shape recognition with incremental training using a neural network with binary synaptic weights*, Industrial Applications of NNs, CRC Press, pp. 159-192, 1999.

[8] J. M. ZURADA: *Introduction to Artificial Neural Systems*, West Publishing Company, St. Paul, 1992.