n Recognition, Vol. 25, No. 1, pp. 25–44, 1992
ed in Great Britain

0031-3203/92 $5.00 + .00
Pergamon Press plc

OBJECT RECOGNITION USING INVARIANT OBJECT BOUNDARY REPRESENTATIONS AND NEURAL NETWORK MODELS

GEORGE N. BEBIS and GEORGE M. PAPADOURAKIS
Department of Computer Science, University of Crete, P.O. Box 1470, Iraklion, Crete, Greece;
Institute of Computer Science, FORTH–Hellas, P.O. Box 1385, Iraklion, Crete, Greece

(*Received* 26 *December* 1990; *in revised form* 15 *April* 1991; *received for publication* 1 *May* 1991)

Abstract—Object recognition is an essential part of any high-level computer vision system. In this paper, several approaches for classifying two-dimensional objects which are based on the use of both invariant boundary transformations and artificial neural networks (ANNs) were implemented and compared. Specifically, the centroidal profile, the cumulative angular and the curvature representations were used. Two different ANN learning approaches were considered. The first involved supervised learning while the second involved unsupervised. In particular, the multilayer ANN trained with the predict back-propagation rule and the Kohonen ANN were utilized. Implementation issues, simulation results and comparisons show the strengths and weakness of each approach, especially when noisy and distorted objects were used for recognition.

Object recognition Centroidal profile Cumulative angular function
Curvature function Artificial neural networks

1. INTRODUCTION

Computer vision systems are currently being introduced in various environments in order to provide machines with the capability to "see", the intelligence to "understand" the surrounding environment and the ability to identify various objects.[1] Most of the current object recognition systems are model-based systems,[2] in which recognition involves matching the input image with a set of predefined models of objects. In such a system the known objects are precompiled, creating a model database, and this database is used to recognize objects in an image scene. Under these circumstances object recognition entails the assignment of an unknown object to one of several classes of objects based on a finite set of object features. However, certain applications require objects to be recognized regardless of their location, size or orientation and these objects could also be overlapped or partially occluded. In addition, problems concerning distortion and noise must be resolved since segmentation errors always exist due to the position of lights, surface material, quantization, reflection and shadows.

In many industrial applications the objects to be recognized can be described by their object boundaries. A careful examination of the boundary of an object reveals that similar objects have similar boundaries. For a large number of object images, use of the object boundary to provide information is attractive because of the reduction in quantity of information, compared to that of the original 2D object images. The process of boundary extraction usually starts by applying an edge detector to the original gray scale image and then applying a thresholding technique. There are many techniques available to describe objects based on their boundaries. Among them are the chain codes[3] and the polygonal approximations.[4] Once the boundary has been established, it is often desirable to express it in terms of 1D functions using boundary transformations. Such transformations include the centroidal profile functions,[5] the slope density function,[6] the cumulative angular function[7] and the curvature function based on the parametric equations of the boundary.[8]

Present object recognition methods can be categorized as either global or local in nature. Global methods are based on global features of the boundary or of an equivalent representation. Such techniques are the Fourier Descriptors,[7] the Moments[9] and methods based on Autoregressive Models.[10] Local methods use local features such as critical points[5] or holes and corners. They perform extremely well in the presence of noise, distortion or partial occlusion since such effects on an isolated region of the contour alter only the local features associated with that region, leaving all the other local features unaffected. However, the choice of representative local features is not trivial and the recognition process based on local features is more computationally intensive and time consuming. On the other hand, global methods have the disadvantage that a small distortion in a

section of a boundary of an object will result in changes to all global features.

After feature extraction has been established, a classification scheme must be used for the recognition stage. Although several robust techniques have been developed, an improvement can be made in the computation time and the degree of noise that a classifier can tolerate. In this paper, the classification of 2D objects is attempted using artificial neural networks (ANNs). The approach is essentially model based, since it is assumed that the objects to be recognized are known in advance, forming a set of models. Most of the existing ANN classifiers require a 2D object to be presented in a fixed position, orientation and size. Thus, a preprocessing stage is required to normalize it. In this study, we present and compare three different 2D object representations which seem to be quite effective for the preprocessing stage. These representations are based on the use of invariant boundary transformations. Specifically, the centroidal profile,[5] the cumulative angular [7] and the curvature[8] boundary transformations are used. Objects to be recognized can be corrupted with noise and distortion. We restrict our study in cases where the objects can be described by their boundaries which are closed and they do not cross themselves. ANNs can perform different tasks, one of which is in the context of a supervised classifier. Utilizing all the information supplied by the invariant contour transformations, the feature extraction and the actual classification is accomplished using ANNs. Over the past few years, a veritable explosion of interest in ANN models and their applications has occurred.[11–13] ANNs possess a number of properties which make them particularly suited to complex classification problems.[14–16] Unlike the traditional classifiers, ANN models are able to examine numerous competing hypotheses simultaneously through the use of massive interconnections among many simple processing elements. In addition, ANNs perform extremely well under noise and distortion conditions.

In a model-based object recognition system, object models are represented into a model database, using a proper knowledge representation scheme. Generally, increasing the number of object models in the model database has the effect that the computational complexity and the time requirements of the system are greatly increased. However, implementation of a model-based object recognition scheme using ANNs seems to be very attractive. First of all, ANNs provide their own way to represent the knowledge that they store.[17] In addition, the complexity and the computational burden increases very slowly as the number of data models increases. Although ANNs' performance is excellent, many researchers still criticize ANNs because they require a lot of training time before they become able to perform a specific task. However, in our case, the recognition phase is of the most importance and it must be implemented as quickly and accurately as possible. Obviously the training phase can be performed off-line.

Two different learning ANN approaches are compared in this work. The first approach utilizes supervised learning while the second approach utilizes unsupervised. Specifically, a multilayer ANN[18] trained with the "predict back-propagation" learning rule[19] and the Kohonen ANN[20] are considered. The paper has been organized as follows: in Sections 2–4 we describe the boundary transformations that are utilized, and in Section 5 we present a brief overview of the multilayer ANNs and the predict back-propagation rule. Section 6 contains a short description of the Kohonen ANN. In Section 7 the experiments, the simulation results and certain comparisons among the different approaches are presented. Section 8 provides a comparison among certain traditional methods and the best ANN approaches. Finally, Section 9 follows with conclusions and remarks.

2. CENTROIDAL PROFILE REPRESENTATION

The centroidal profile is a 1D representation of the boundary of an object. It is characterized by an ordered sequence that represents the distance from the digitized boundary of the object to its centroid as a function of distance along the boundary. A simple object it shown in Fig. 1(a) and its corresponding centroidal profile is illustrated in Fig. 1(b). The length of the sequence is determined by the number of points along the boundary. Freeman[5] describes a method for obtaining the centroidal profile from a chain-coded boundary. Our technique is based on the Cartesian coordinates of the boundary. The centroid (X_g, Y_g) is estimated using the following formulae:

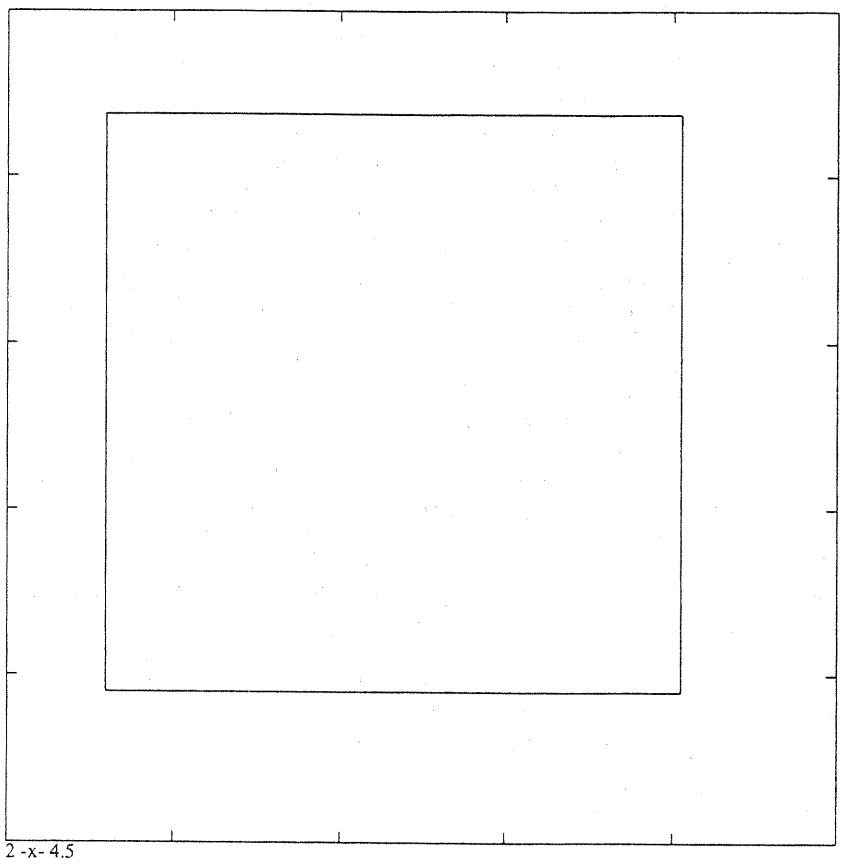
$$X_g = \frac{\sum_x \sum_y f(x,y)x}{\sum_x \sum_y f(x,y)} \qquad Y_g = \frac{\sum_x \sum_y f(x,y)y}{\sum_x \sum_y f(x,y)},$$

where $f(x, y)$ is set to 1 for object points and set to 0 elsewhere. Since the centroid's coordinates are determined by the ratio of first-order moments to enclosed area, they are relatively insensitive to noise and therefore the centroid tends to be a stable point of reference for the object.

Next, defining N as the number of points along the boundary, the distances $d(i)$, $i = 1, 2, \ldots, N$, from the centroid to the boundary points $\{x(i), y(i), i = 1, 2, \ldots, N\}$, are computed starting from an arbitrary position of the boundary and tracking the boundary in a counterclockwise direction. Using as distance measure the Euclidean norm we have:

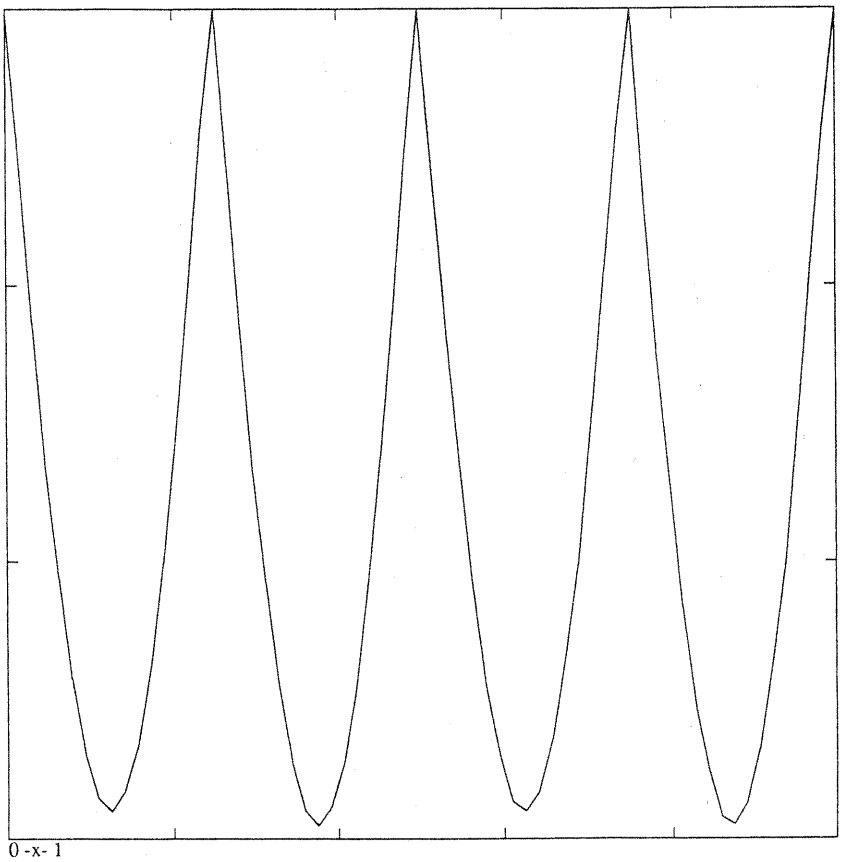
$$d(i) = \sqrt{(x_i - X_g)^2 + (y_i - Y_g)^2}, \quad i = 1, 2, \ldots, N.$$

The centroidal profile is similar to other boundary representations such as Peli's shape signature[21]



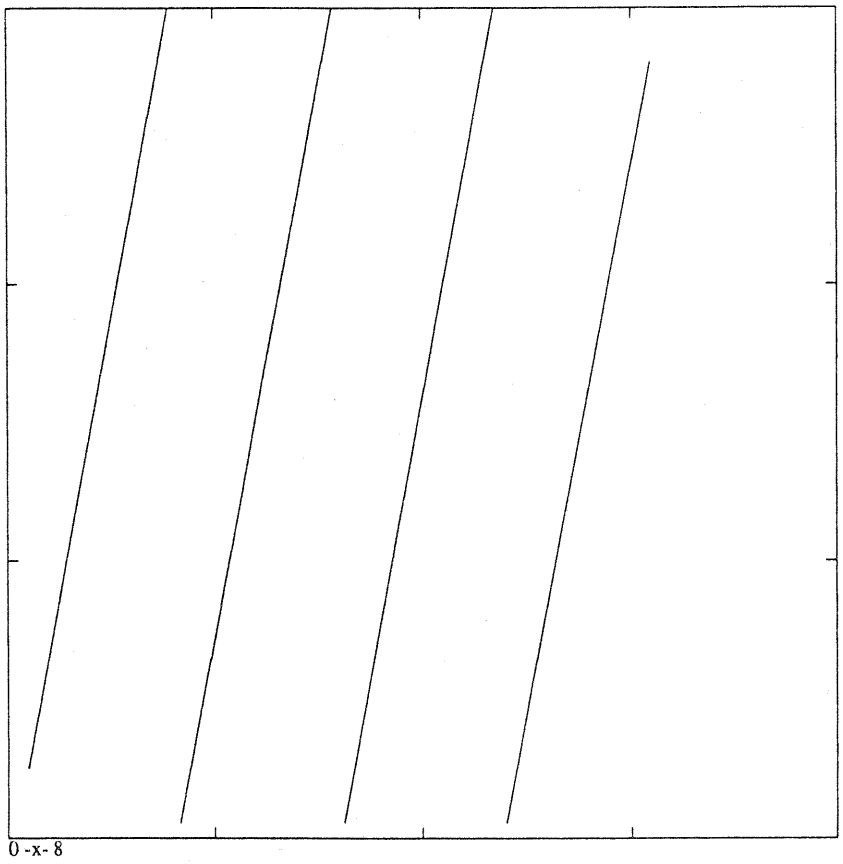
(a)

Fig. 1. A simple object and its boundary representations: (a) a square; (b) centroidal profile representation; (c) cumulative angular representation; (d) curvature representation.



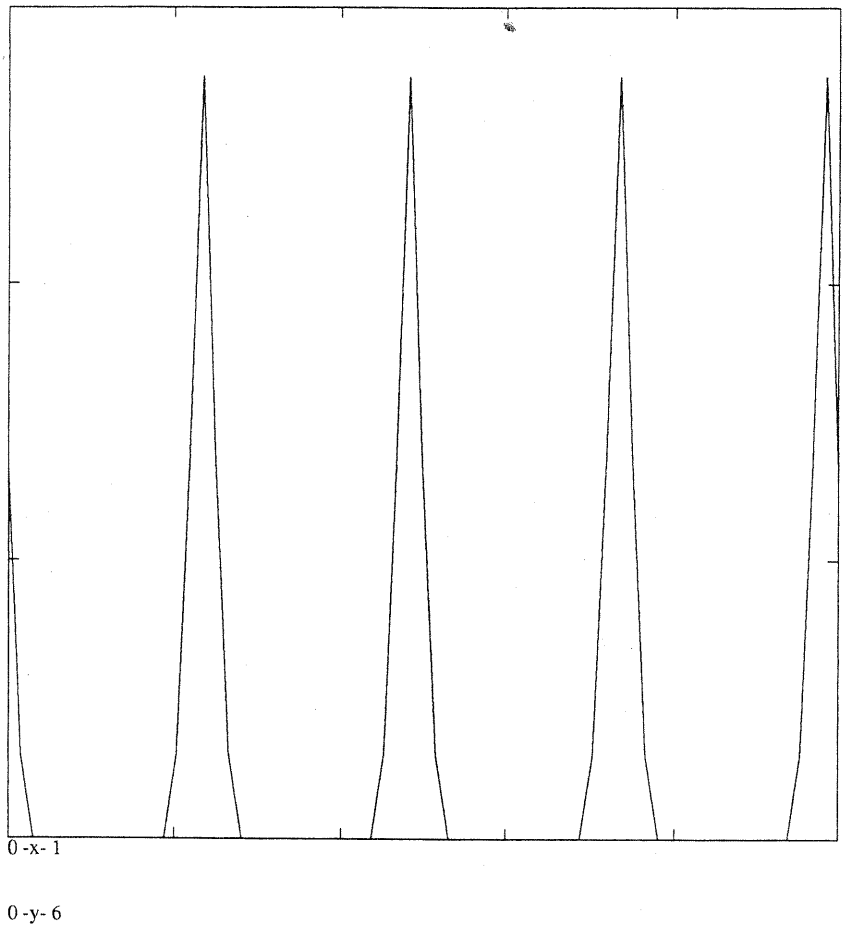
(b)

Fig. 1. (*Continued.*)



(c)

Fig. 1. (*Continued.*)



(d)

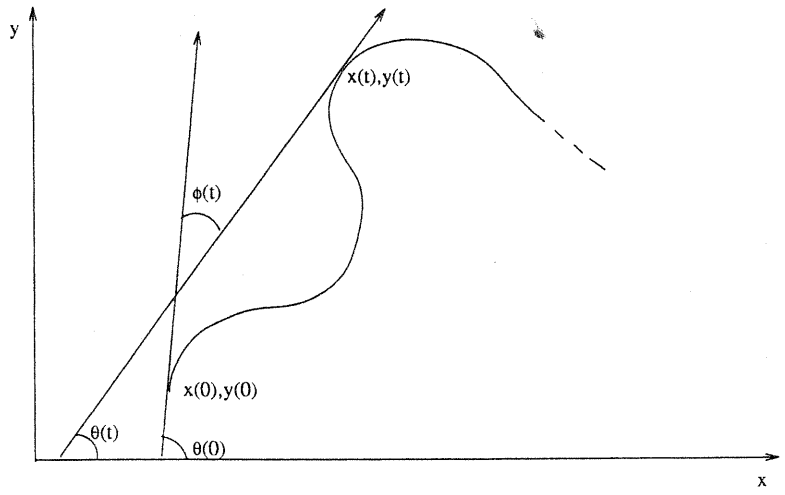
Fig. 1. (*Continued.*)

that instead of taking sample points at equal distances on the boundary, the sampled points are taken at equal angular steps. The angular step size should satisfy the sampling theorem for the contour waveform. The representation obtained by sampling the contour at equal angles from the centroid can be used for the reconstruction of the original boundary. Furthermore, it has a fixed duration and this is a very convenient representation for techniques that require a fixed number of observations. However, the shape variation profile has some serious disadvantages. Equi-angular sampling of the boundary does not lead to uniform spacing among the selected points along the boundary. In addition, problems may occur if the boundary includes highly convex and concave regions. This has the impact that the resulting representation can be a multivalued function. There have been proposed[22] some solutions to the above problems but they require additional processing cost and further approximations. The centroidal profile function overcomes the above disadvantages. The boundary can be sampled at equal distances and the resulting representation is always a single valued function.

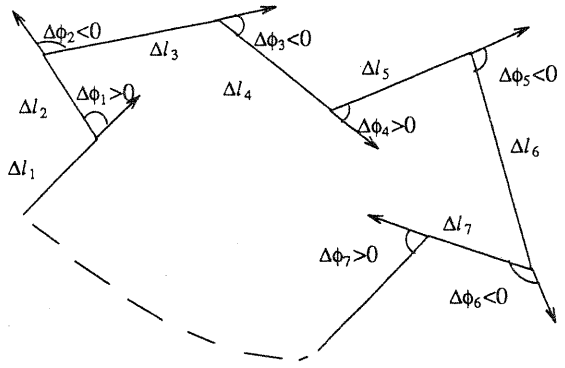
A boundary sampled at equal distances provides a centroidal profile that can be used for the reconstruction of the original boundary. Specifically, if we know the starting boundary point, the normalization scale factor and the distance between boundary points, the centroidal profile generation process is reversible. This suggests that two objects can be compared by comparing their centroidal profiles. In other words, the similarity between two objects can be determined by the degree of similarity between their centroidal profiles.

3. CUMULATIVE ANGULAR REPRESENTATION

The cumulative angular representation is a 1D description of an object's boundary which has been proposed by Zahn and Roskies,[7] as an alternative way of defining the Fourier descriptors of the object's boundary. The boundary of an object can be described by the parametric equations $\{x(t), y(t)\}$,



(a)



(b)

Fig. 2. Changes in angular direction: (a) the $\theta(t)$ and $\phi(t)$ functions; (b) description of simple closed polygon in terms of Δl_i and $\Delta\phi_i$.

where t is a linear function of arc length with $0 \leq t \leq L$, and L denotes the total length of the boundary. Representing the angular direction of the boundary at point t by the function $\theta(t)$, then the cumulative angular function $\phi(t)$ can be defined as the net amount of angular bend between starting point $t = 0$ and point t. Assuming that a boundary does not cross itself, the cumulative angular function can be defined as:

$$\phi(t) = \theta(t) - \theta(0).$$

Figure 2(a) illustrates graphically both $\theta(t)$ and $\phi(t)$. The cumulative angular representation of the object shown in Fig. 1(a) is presented in Fig. 1(c).

An alternative way to compute the cumulative angular representation utilizes changes in angular direction $\Delta\phi_i$s of its successive vertices. Figure 2(b) illustrates the vertex bends $\Delta\phi_i$s of a simple polygon contour. Assuming that the length of vertex i is Δl_i and the change in its angular direction is $\Delta\phi_i$, then:

$$\phi(t) = \sum_{i=1}^{k} \Delta\phi_i \text{ for } \sum_{i=1}^{k} \Delta l_i \leq t \leq \sum_{i=1}^{k+1} \Delta l_i.$$

Zahn and Roskies have shown that the boundary of an object can be reconstructed by its cumulative angular function $\phi(t)$. It is obvious that object boundaries having different cumulative angular representations are distinct. This suggests that this representation can be utilized for object classification.

4. CURVATURE REPRESENTATION

Since Attneave's famous observation[23] that information on the shape of a curve is concentrated at dominant points having high curvature, the curvature function of a curve plays an important role in image

analysis. There have been many approaches for a precise definition of discrete curvature. The curvature k of a planar curve at a point P on the curve is defined as the instantaneous rate of change of the slope angle ψ of the tangent at point P with respect to arc length s:

$$k(s) = \frac{d\psi(s)}{ds}.$$

The curvature inverse equals the radius of a circle (the osculating circle) whose curvature agrees with that of the contour at point P. Assuming that the curve C is expressed by the parametric equations $\{x(t), y(t)\}$, where t is a linear function of the path length ranging over the closed interval $[0, 1]$, then the curvature k can be computed as follows:

$$k = \frac{y''}{(1 + (y')^2)^{3/2}}, \tag{4.1}$$

where y' and y'' are defined as:

$$y' = \frac{dy}{dx}, \quad y'' = \frac{d^2y}{dx^2}. \tag{4.2}$$

Furthermore, if we define:

$$\dot{x} = \frac{dx}{dt}, \quad \ddot{x} = \frac{d^2x}{dt^2}, \quad \dot{y} = \frac{dy}{dt}, \quad \ddot{y} = \frac{d^2y}{dt^2}, \tag{4.3}$$

then we can express y' and y'' as:

$$y' = \frac{\dot{y}}{\dot{x}}, \quad y'' = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^3}. \tag{4.4}$$

Substituting Equation (4.4) into (4.1) the resulting expression for the curvature k is given by:

$$k = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}. \tag{4.5}$$

The curvature function of the object shown in Fig. 1(a) is illustrated in Fig. 1(d).

Discrete curvature computation depends on a neighborhood which is called region of support, whose size must be chosen. Discrete curvature may be defined by simply replacing the derivatives in Equation (4.1) by first differences. However, this leads to the problem that small changes in slope are impossible, since successive slope angles on the digital curve can differ only by a multiple of 45°. Furthermore, curvature depends on higher order derivatives making it sensitive to noise. In order to select the region of support and to provide adequate noise insensitivity, a smoother version of the curvature is utilized, which is obtained by convolving the parametric functions $x(t)$ and $y(t)$ of the curve with a 1D Gaussian[8] kernel $g(t, \sigma)$:

$$g(t, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-t^2/2\sigma^2}.$$

The convolution of the $x(t)$ with the Gaussian kernel is defined as:

$$X(t, \sigma) = x(t)^* g(t, \sigma) = \int_{-\infty}^{\infty} x(u) \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-u)^2/2\sigma^2} du$$

and $Y(t, \sigma)$ is defined similarly. The smooth curvature k_σ is similar to Equation (4.5) and is given by:

$$k_\sigma = \frac{\dot{X}\ddot{Y} - \dot{Y}\ddot{X}}{(\dot{X}^2 + \dot{Y}^2)^{3/2}},$$

where

$$\dot{X} = x(t)^* \frac{\partial g(t, \sigma)}{\partial t}, \quad \ddot{X} = x(t)^* \frac{\partial^2 g(t, \sigma)}{\partial t^2}$$

and $\dot{Y}$, $\ddot{Y}$ are defined in a similar manner.

It should be mentioned that, using elementary techniques of differential geometry,[24] a curve may be reconstructed from its curvature function given a starting point and tangent specifications. Thus, object contours having similar curvatures are similar.

5. MULTILAYER ANNs AND THE PREDICT BACK-PROPAGATION RULE

Artificial neural networks are specified by the topology of the network, the characteristics of the nodes (i.e. nerves) and the processing algorithm. The intelligent information properties of an ANN arise from the above specifics. The topology of a multilayer ANN is a structured hierarchical layered network as shown in Fig. 3. It consists of several distinct layers of nodes including an input layer and an output layer. Between the input and the output layer we have one or more layers of nodes which are called hidden. Hidden nodes, the nodes in the hidden layers, are used to represent domain knowledge useful for solving recognition tasks.[17] Generally, each node in one layer is interconnected with all the nodes in adjacent layers with connections, known as synapses. Each connection is associated with a weight which measures the degree of interaction between the corresponding nodes. Nodes are relatively simple

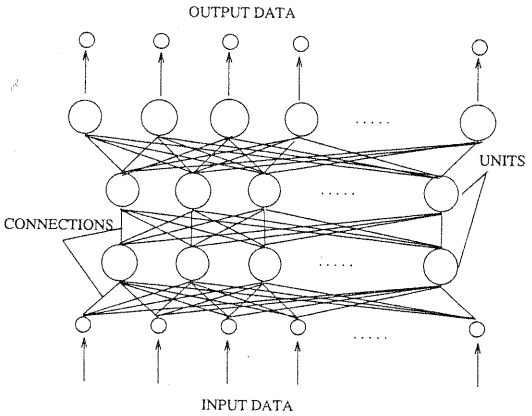


Fig. 3. A multilayer Artificial Neural Network.

processing elements and the capabilities of multilayer ANNs stem from the nonlinearities used within them.

The algorithms for multilayer ANN processing can be divided into two phases: retrieving and learning. In the retrieving phase of the algorithm, information flows from the input layer through the hidden layers to the output layer. Basically in this phase the nodes update their own activation values based on the system dynamics. In the learning phase, an adaptation of the weights corresponding to the connection nodes takes place. In this paper the predict back-propagation[19] learning algorithm is used which is an improvement of the popular back-propagation rule.[18] This algorithm performs supervised learning and in each step adjusts the connection weights in the network, minimizing the mean-square error between the target value (the desired) and the output value (the actual) of the network.

A general L-layered feed-forward artificial neural net consists of N_0 input nodes and N_L output nodes. The number of nodes in the hidden layers is N_l for $1 \le l \le L-1$. In our notation the input layer is not counted as a layer. So an L-layer feed-forward artificial neural net has $L-1$ hidden layers and the Lth layer is the output layer. During the retrieving phase, we present continuous valued input data $x_1, x_2, \ldots, x_{N_0}$ called exemplar patterns and the corresponding desired output data $t_1, t_2, \ldots, t_{N_L}$, called target patterns. Input data are propagated forward through the network, which computes the activation value for each node, until the output layer is reached. The learning phase involves a backward pass through the network during which the error signals produced at the output layer are passed to each node in the network and appropriate weight changes are made. For each weight the gradient of the output error with respect to that weight is computed. Then, the weight is changed in the direction that reduces error.

In the original back-propagation algorithm the exemplars are presented to the network continuously, in a generally random way. The predict back-propagation algorithm determines a dynamic order in which exemplar patterns are loaded to the network, depending on the "learning difficulty" of each exemplar pattern. Specifically, when an exemplar is loaded to the network, the error signals produced at the output layer yield a measure of its learning "difficulty".

Predict back-propagation algorithm

Step 1. Initially, the weights w_{ij} and the internal thresholds take some small random values.

Step 2. The total errors $\varepsilon^{(m)}$, for $m = 1, 2, \ldots, M-1$, are calculated by presenting to the network all the training exemplar patterns. If the patterns have been learned (that is, the total errors are less than a threshold), then the training stops.

Step 3. The exemplar pattern having the maximum total error is chosen and presented to the network.

Step 4. *Retrieving phase.* The output results per layer are calculated:

$$\alpha_i^{(1)}(t) = \sum_{j=1}^{N_0} w_{ij}^{(1)}(t) o_j^{(0)}(t) + \theta_i^{(1)}$$

$$o_i^{(1)} = \mathrm{f}(\alpha_i^{(1)})(t), \quad 1 \le i \le N_1$$

$$\alpha_i^{(2)}(t) = \sum_{j=1}^{N_1} w_{ij}^{(2)}(t) o_j^{(1)}(t) + \theta_i^{(2)}$$

$$o_i^{(2)} = \mathrm{f}(\alpha_i^{(2)})(t), \quad 1 \le i \le N_2$$

$$\vdots \qquad \vdots$$

$$\alpha_i^{(L)}(t) = \sum_{j=1}^{N_{L-1}} w_{ij}^{(L)}(t) o_j^{(L-1)}(t) + \theta_i^{(L)}$$

$$o_i^{(L)} = \mathrm{f}(\alpha^{(L)})(t), \quad 1 \le i \le N_L,$$

where f is a nonlinear output function, $\{o_i^{(l)}(t), l = 1, 2, \ldots, L$, is the output result of node i, of layer l at time $t\}$, $\{o_i^{(0)}(t) = x_i$, that is, the current exemplar input$\}$, $\{\alpha_i^{(l)}(t)$ is the activation value of node i in layer $l\}$, $\{\theta_i^{(l)}$ is the internal threshold of node i in layer $l\}$, and $\{w_{ij}^{(l)}(t)$ is the weight coefficient between the node j of layer $l-1$ and node i of layer $l\}$.

Step 5. *Learning phase.* The weights between two successive layers (l and $l-1$) are adjusted, starting from the output layer until the input layer is reached (that is, $l = L, L-1, \ldots, 1$), as follows:

$$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) + \eta \delta_i^{(l)}(t) o_j^{(l-1)}(t),$$

where η is the learning rate and $\delta_i^{(l)}(t)$ is the error signal, which is determined as follows:

$$\delta_i^{(L)}(t) = (t_i^m - o_i^{(L)}(t)) o_i^{(L)}(t)(1 - o_i^{(L)}(t)) \tag{5.1}$$

$$\delta_i^{(l)}(t) = o_i^{(l)}(t)(1 - o_i^{(l)}(t)) \sum_j \delta_j^{(l+1)}(t) w_{ji}^{(l+1)}(t), \tag{5.2}$$

where t_i^m is the desired output value of node i in layer L, when the m exemplar pattern is presented to the network.

Step 6. The total error for the current exemplar pattern is updated. If the current exemplar pattern has been learned (that is, its total is less than a threshold) then continue to Step 2, otherwise, continue to Step 3.

The selection of the output function f has a critical role to the performance of the algorithm. Generally,

this function is the sigmoid function

$$o_i^{(l)}(t) = f(\alpha_i^{(l)}(t)) = \frac{1}{1 + e^{-\alpha_i^{(l)}(t)}}.$$

Internal node thresholds $\theta_i^{(l)}$ should be learned just like any other weight. We simply imagine that $\theta_i^{(l)}$ is the weight from a node that always has constant-valued inputs. It is easy to show that the internal thresholds can be adjusted according to the following rule:

$$\theta_i^{(l)}(t+1) = \theta_i^{(l)}(t) + \eta\delta_i^{(l)}(t).$$

The learning "difficulty" of each exemplar pattern is computed by calculating the total errors, which depend on the output signal errors. Let $e_i^{(m)}$ denote the error associated with the ith output node, when the m exemplar pattern is presented to the network. This error is defined as follows:

$$e_i^{(m)} = t_i^{(m)} - o_i^{(L)}(t).$$

The total error $e^{(m)}$ can be calculated by choosing the max-norm of $e_i^{(m)}$ over all the output nodes, as follows:

$$e^{(m)} = \max_i |t_i^{(m)} - o_i^{(L)}(t)|,$$

for each $m = 1, 2, \ldots, M-1$, where M is the total number of training exemplar patterns. Calculating the total errors $e^{(m)}$ and finding the maximum of them has the effect that the exemplar pattern having the maximum error signals (described by Equation (5.1), which indirectly depends on Equation (5.2)) is chosen. Thus, at each iteration of the algorithm, the exemplar pattern having the maximum total error is chosen and presented to the network.

It should be noted that each time the learning phase takes place, the total errors associated with the training patterns are changed, since the weight coefficients are adjusted. In order for our algorithm to be strictly enforced, an update of the total errors $e^{(m)}$, for $m = 1, \ldots, M-1$, must be performed each time we are loading an exemplar pattern to the network. However, the computation cost of such an action is very high and it affects negatively the performance of the algorithm, although it results in a decreased number of learning steps. Thus, the total errors are not computed in each iteration of the algorithm but only when some specific assumptions are held. Specifically, each exemplar pattern is loaded to the network repeatedly, until the learning for this pattern has been accomplished (however, its learning may not have been completed). Next, all the exemplar patterns are presented to the network (only the retrieving phase takes place) and their associated errors are recalculated. The exemplar pattern having the total associated error is chosen and loaded to the network. The same process is repeated until all the exemplar patterns have been learned.

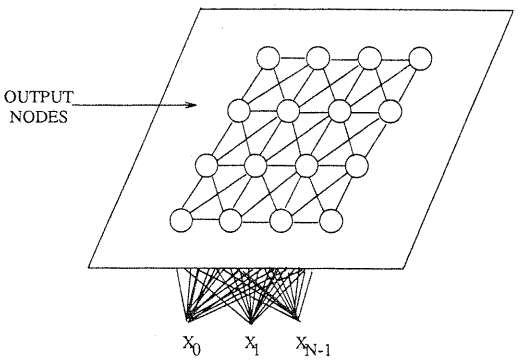


Fig. 4. The Kohonen Artificial Neural Network.

6. THE KOHONEN ANN

Kohonen[20] has proposed this ANN algorithm in order to produce self-organizing feature maps similar to those occurring in the brain. In this study, Kohonen's ANN is used as a clustering algorithm, which is similar to the K-Means traditional classifier.[25] This ANN is one-layered, and its output units form a 1D or 2D array, as shown in Fig. 4. Each input node is connected to each output node. The number of input nodes is determined by the dimension of the exemplar patterns. For a pattern classification task, the required classes are represented by the output nodes.

The training of the algorithm is unsupervised. Continuous valued input vectors are presented randomly in time without specifying the desired output. The mapping from the external input patterns to the network's activity patterns is realized by correlating the input patterns with the connection weights. After enough input patterns have been presented, weights will specify cluster centers that sample the input space. In other words, the representative exemplar pattern of each class is stored in the input to output node connection weights.

This ANN operates in two phases: the similarity matching phase and the weight adaptation phase. Initially, the weights are set to small random values and a pattern is presented to the input nodes of the network. During the similarity matching phase, the distances d_j between the input and output nodes are computed using the Euclidean distance;

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2,$$

where $x_i(t)$ is the input to node i at time t and $w_{ij}(t)$ is the weight from input node i to output node j at time t. Next, the output node j^* having the minimum distance d_{j^*} is chosen. The next step in this phase is the definition of a topological neighborhood N_{j^*} of this node j^*.

In the second phase, the weights connecting the input nodes to the selected output node j^* takes place. In addition, the weights from the input nodes

which are contained in the neighborhood N_{j^*} of node j^* are also adapted. The weight changes are based on the rule:

$$w_{ij} = w_{ij}(t) + \eta(x_i(t) - w_{ij}(t)),$$

for $j \in N_{j^*}$ and $0 \leq i \leq N-1$. The parameter η is the learning rate of the algorithm. This process is repeated until weight convergence is accomplished.

Generally, the parameters N_{j^*} and η do not remain constant over time. As Kohonen points out after extensive computer simulations, the performance of the algorithm improves if N_{j^*} and η decrease slowly with time. Thus $N_{j^*} = N_{j^*}(t)$ and $\eta = \eta(t)$. However, the way that these parameters are adjusted is determined by experience, as indicated by Kohonen.[20]

7. IMPLEMENTATION ISSUES AND SIMULATION RESULTS

Given an arbitrary boundary representation, several successive steps can normalize it so that it can be matched to a test set of boundary representations regardless of the original object's size, position or orientation. In the following subsections, the normalization steps required by each boundary representation to become invariant under translation, rotation and scaling are described.

7.1. *Centroidal profile normalization*

The centroidal profile representation consists of a circular sequence which can be easily shown to be invariant to translation and rotation, provided that the same starting contour point is used. A change in starting point results in a circular shift of the centroidal profile representation. A way to select a good starting point is to choose the contour point closer to the major principal axis and located further away from the centroid. However, noise and distortion effects can easily affect this starting point selection scheme, which results in the selection of another starting point. It should be mentioned that the ANNs in general do not recognize circular shifted patterns, although it is possible to recognize noise and distorted inputs. This suggests that a more powerful way must be devised to guarantee, as much as possible, the selection of a unique starting point. This problem is anticipated by choosing the starting point at a higher resolution. First, the boundary is convolved with a Gaussian filter of large variance σ which in our application was set to $6\sqrt{2}$. While this operation smooths the boundary it still preserves its major characteristics. Selecting the contour point closer to the major principal axis and further away from the centroid but at higher resolution and then corresponding it to the original resolution, the starting point is chosen in an optimal manner. Scale dependence can be removed by dividing all profile values by their maximum value.

7.2. *Cumulative angular normalization*

Since the cumulative angular function is defined over [0, 1], it simply contains absolute size information. Furthermore, it can be shown that $\phi(L) = -2\pi$ since all smooth closed curves with clockwise orientation have a next angular bend of -2π. Thus, $\phi(L)$ does not convey any boundary information. Zahn and Roskies[7] have proposed a normalized version of $\phi(t)$, denoted by $\phi^*(r)$, which is invariant under translation, rotation and scaling. The domain definition of $\phi^*(r)$ is normalized to the interval $[0, 2\pi]$. The normalized cumulative angular function $\phi^*(r)$ is defined as follows:

$$\phi^*(r) = \phi\left(\frac{Lr}{2\pi}\right) + r,$$

where $\phi^*(0) = \phi^*(2\pi) = 0$.

As an intuitive justification for the definition of ϕ^* it is noted that $\phi^* = 0$ for a circle which is in some sense the most shapeless closed contour. Thus, the function $\phi^*(r)$ measures the way in which a boundary differs from a circular contour. Starting point selection is accomplished using exactly the same methodology described in Section 7.1.

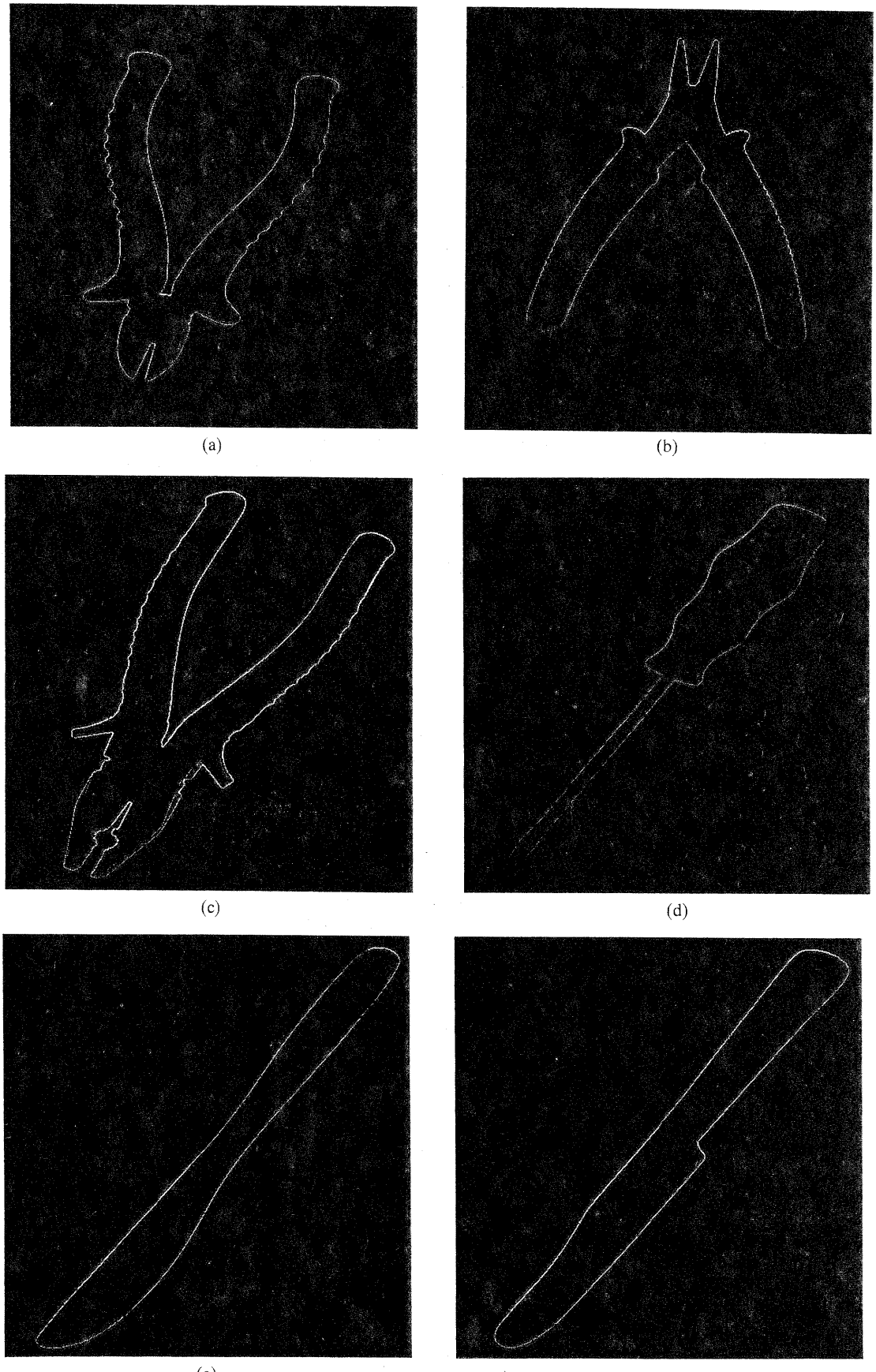
7.3. *Curvature normalization*

Curvature computation is performed by convolving the parametric equations of a curve with a Gaussian kernel which is defined over an infinite range, but it can be safely truncated at a distance of 3σ from its center. The kernel size is chosen to be 5σ, which is the size that can still provide a good approximation for the second derivative of the Gaussian function. Thus, the region of support for the computation of the curvature function is chosen to be 5σ. In our experiments, σ was set to $\sqrt{2}$. Since curvature computation is based on the use of first and second order derivatives, it is very sensitive to noise. The above value of σ preserves most of the fine detail in the object's contour and provides adequate noise suppression.

Curvature computation of a path-based parameterized object boundary yields translation and rotation invariance properties provided that the same starting boundary point is used (i.e. the point at $t = 0$). A change of starting point leads to a circular shift of the curvature function. The starting point is selected by using the methodology described in Section 7.1. Scale invariance can be accomplished by dividing all the curvature values by the maximum curvature value.[8]

7.4. *Simulations*

In order to evaluate the performance of the previously described invariant contour representations, two object sets were used. Specifically, the first set

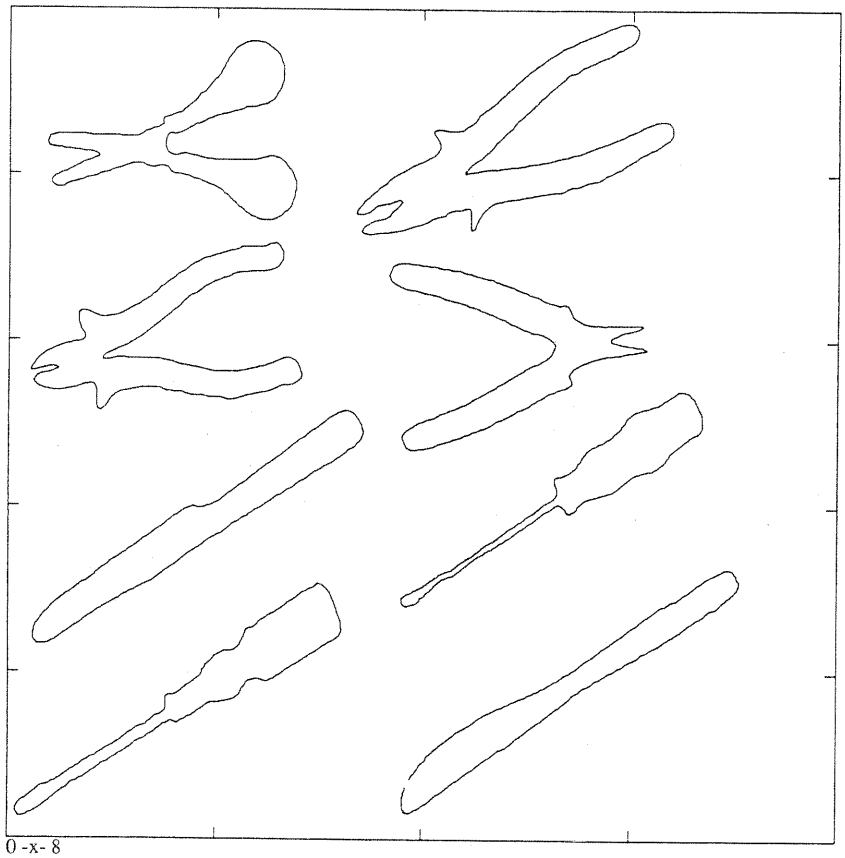


(a) (b) (c) (d) (e) (f)

Fig. 5. Raw image data.

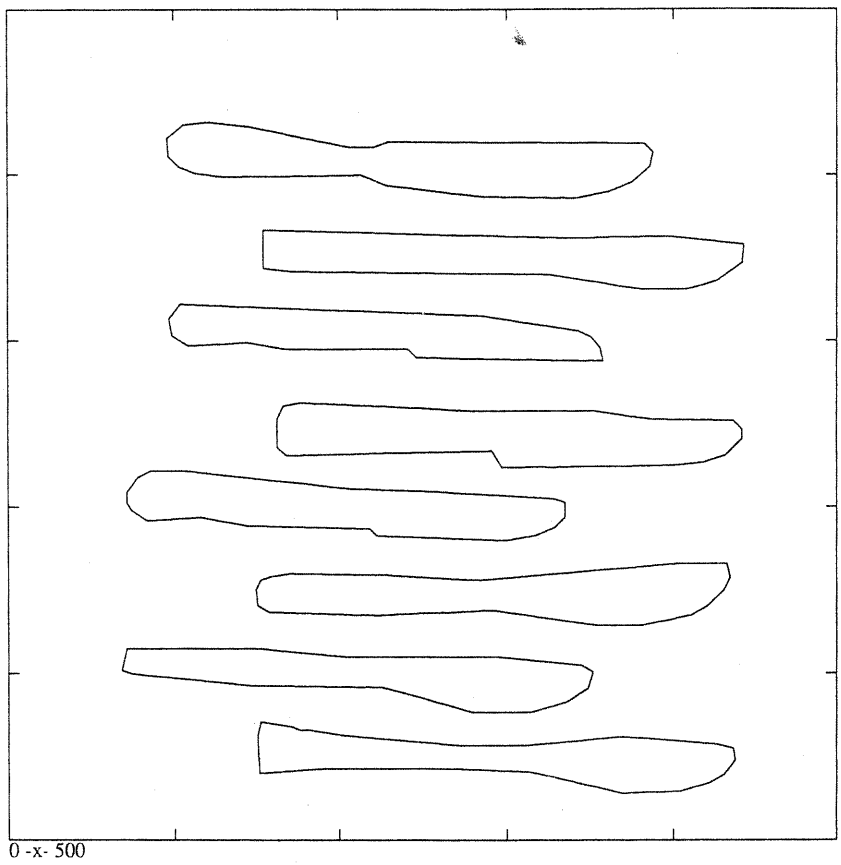
consists of the images of eight different knives, while the second set consists of the images of eight different tools. The objects were positioned on a light table and as a result high contrast images were obtained. A Laplacian edge detector separated the objects from the background and a boundary-following routine extracted their boundaries (Fig. 5). Object contours were then smoothed using a Gaussian function of $\sigma = 1.0$. The filter's masksize was equal to 5. The 16 closed contour boundaries are shown in Figs 6(a) and (b). The first set includes dissimilar objects and the discrimination task seems to be easy. On the other hand, the second set contains very similar objects, making their discrimination very difficult. All the boundaries were approximated by 256 contour pixels along the boundary. Inner boundary information was not utilized in this study. The length of each boundary representation determines the number of input nodes (256) for all the ANNs. The number of output nodes in each ANN was set to 8, indicating the number of distinct objects in each set. The correct recognition of an object is indicated by the value of one in the corresponding output node, while the remaining output nodes take the value of zero.

7.4.1. *Implementation of the multilayer ANN.* The multilayer ANN architecture used in this study consists of three layers (two of them are hidden). Without any a priori knowledge, we have set the number of nodes per hidden layer equal to 50. The value of learning rate was set to 0.2 and it remained constant. As we have mentioned earlier, at each time step of the algorithm, the exemplar pattern having the maximum associated total error is chosen to be presented to the network. The updating of the errors associated to each exemplar pattern is done in relation to the termination condition of the algorithm. Our termination condition determines that one task is considered solved when the total error associated with each exemplar pattern is less than a given threshold. Each time that the termination condition was true for some exemplar, the rest of exemplars were presented to the network, one after the other, until for some exemplar the termination condition was false. At the same time, the total error associated to each exemplar pattern is updated. This strategy decreases by a large amount the number of steps required for the updating without seriously increasing the number of learning steps. In these simulations, the threshold parameter was set to 0.2.



(a)

Fig. 6. (a) The set of different tools; (b) the set of similar knives.



(b)

Fig. 6. (*Continued.*)

7.4.2. *Implementation of the Kohonen ANN.* In this ANN we must determine simultaneously two parameters, the radius of neighborhood and the learning rate, as mentioned earlier in Section 6. Specifically, we must find the ways in which to decrease the radius of neighborhood and to change the learning rate $\eta(t)$. This task is extremely difficult and proper values for these parameters can best be determined experimentally[24] using the error and trial method. It should be mentioned that the adjustment of these parameters is more complex since they contain other subparameters, which in turn must be fine tuned. Our choices are based on the ideas given by Kohonen *et al.*[26] An initial task is to divide the whole problem into two parts. Assuming that the whole process contains t_2 steps, we must determine the number of steps t_1, in the first part. Choosing the initial radius of neighborhood, it decreases linearly to zero after t_1 steps. Then, in the remaining $t_2 - t_1$ steps the adaptation of the weights take place only for the weights connecting the input nodes with the selected output node. In our simulations the neighborhood is 1D (1×8) and the initial value of the radius of neighborhood was set to 1×2. Defining k_1 and k_2 as two constants, a simple practical choice for $\eta(t)$ is: for $0 \le t \le t_1$, $\eta(t) = k_1(1 - t/t_1)$, and for $t_1 \le t \le t_2$, $\eta(t) = k_2(1 - t/t_2)$. In our simulations, we have set $t_1 = t_2/2$, $k_1 = 0.2$ and $k_2 = 0.8$.

7.4.3. *Experiments.* All the experiments were implemented on a Sun 4/110 Workstation using C. For each object, test data sets were formed by changing the size of the object in various translational and rotational positions. In addition, these samples were corrupted with noise and distortion. Specifically, the objects were rotated over the range from 0 to 2π radians and translated in random positions within the image boundaries. The size of each object image was varied from 0.5 to 1.5 times the size of the original object. Noise and distortion effects were introduced by adding random noise to the boundary points according to the approach used by You and Jain.[27] Specifically, if the coordinates of the kth boundary point are $(x(k), y(k))$ then the coordinates of the corresponding point on the noisy boundary $(x_n(k), y_n(k))$ are given by:

$$x_n(k) = x(k) + drc\cos(\theta(k))$$

$$y_n(k) = y(k) + drc\sin(\theta(k))$$

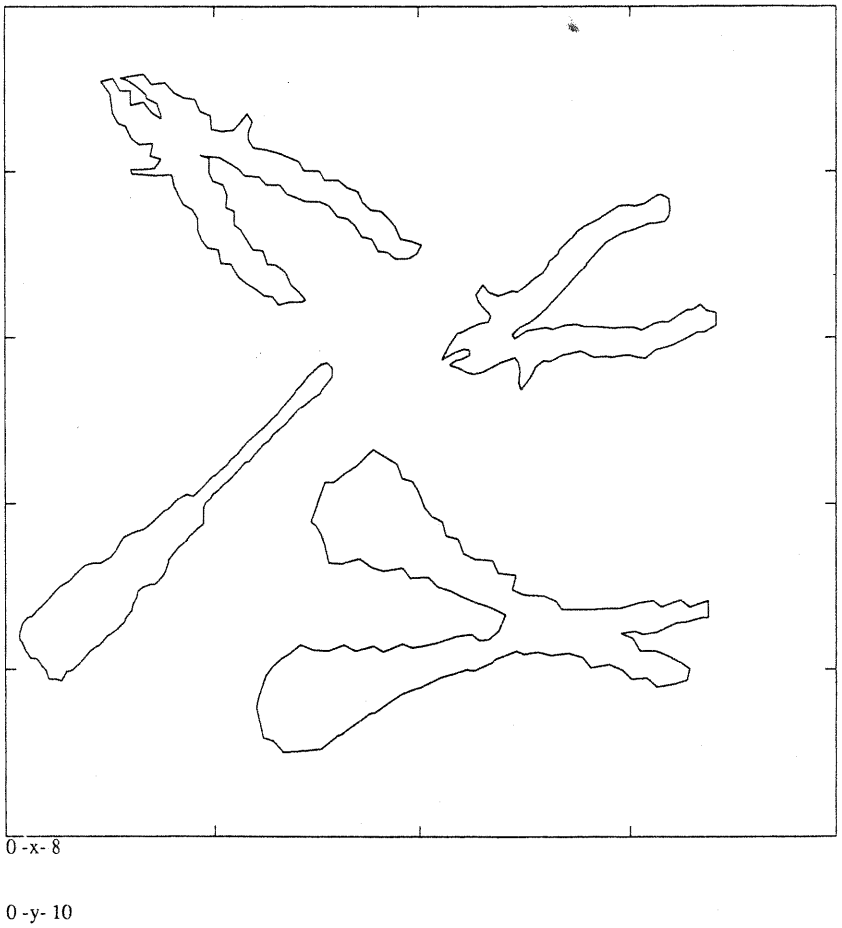


Fig. 7. Objects corrupted by noise.

where d is the distance of boundary point k to point $k+1$, r is a sample from the Gaussian distribution $N(0, 1)$, c is a parameter which controls the amount of distortion set to 0.9, and $\theta(k)$ is the angle from the x-axis to the normal direction of the boundary at point k. The points that caused crossover on the boundary were omitted. The measure of noise in an object contour is determined by the percentage of contour points corrupted with noise, and it is called noise percentage. Some noisy objects in various locations, orientations and sizes are shown in Fig. 7.

7.4.4. *Results.* Initially, each ANN was trained with the centroidal representation of each object shown in Fig. 5. After training, each object representation was assigned to a distinct output class. Next, the ANNs were trained with the cumulative angular and curvature representations of the objects belonging to the same set. The same process was then repeated for the objects shown in Fig. 6. During the training phase the most troublesome algorithm was the Kohonen ANN because it did not classify each character to a distinct class. This algorithm contains a large number of parameters which were determined after extensive experimentation. Table 1 shows the number of steps (t_2) required for weight convergence, after these parameters were established. Each step is defined as a pass of all the exemplar patterns through the network. On the other hand, during the training phase of the predict backpropagation algorithm, no such difficult parameter determination was needed. However, its convergence rate was slower in most cases. The number of steps required during the training phase for each task are presented in Table 2.

After the training phase was completed, the ability of the above ANNs to recognize noiseless objects as well as objects corrupted with noise was evaluated. The recognition phase was very simple for both ANNs and it consisted of only a feed-forward pass of the information presented to the input layer. In terms of accuracy, the simulation results are illustrated in Table 3. These results show the percentage of recognition accuracy when input objects are corrupted with a noise percentage. Our simulations indicate that recognition of pure objects from both data sets, regardless of the representation used, was perfect for both ANNs. Comparing different representations for both data sets in the presence of

Table 1. Number of steps for the different objects

	Different objects	
	Multilayer	Kohonen
Centroidal	1953	46
Cumulative	144	46
Curvature	504	20

Table 2. Number of steps for the similar objects

	Different objects	
	Multilayer	Kohonen
Centroidal	22,550	52
Cumulative	258	34
Curvature	568	36

Table 3(a). Centroidal profile representation using the multilayer ANN

Noise	Different objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100	100
30	100	100	100	100	100	100	100	100
40	100	100	100	100	100	100	100	100

Table 3(b). Centroidal profile representation using the Kohonen ANN

Noise	Different objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100	100
30	100	100	100	100	100	100	100	100
40	100	100	100	100	100	100	100	100

Table 3(c). Cumulative angular representation using the multilayer ANN

Noise	Different objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	97	87	90
20	100	100	97	93	100	93	83	83
30	100	100	97	93	100	93	83	83
40	97	93	97	83	100	87	63	80

Table 3(d). Cumulative angular representation using the Kohonen ANN

Noise	Different objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	90	100
20	100	100	100	100	100	97	87	93
30	100	100	97	97	100	93	83	87
40	100	93	93	90	100	87	67	83

Table 3(e). Curvature representation using the multilayer ANN

Noise	Different objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100	100
30	100	97	100	100	100	100	97	93
40	97	93	100	100	97	90	93	90

Table 3(f). Curvature representation using the Kohonen ANN

Noise	Different objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100	100
30	100	100	100	100	100	100	100	100
40	100	100	100	100	100	100	100	100

Table 3(g). Centroidal profile representation using the multilayer ANN

Noise	Similar objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100	100
30	100	100	100	100	100	100	100	100
40	97	100	100	100	100	100	93	97

Table 3(h). Centroidal profile representation using the Kohonen ANN

Noise	Similar objects							
(%)	1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100	100
30	100	100	100	100	100	100	100	100
40	97	100	100	100	100	100	93	97

Table 3(i). Cumulative angular representation using the multilayer ANN

Noise (%)	Similar objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	97	100	95	97	100	100	97	100
20	93	100	93	95	100	100	93	97
30	77	100	90	93	97	97	90	90
40	73	90	80	80	90	97	73	80

Table 3(j). Cumulative angular representation using the Kohonen ANN

Noise (%)	Similar objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	97	100	95	100	100	100	97	100
20	93	100	93	100	100	100	93	97
30	80	100	90	100	100	100	90	90
40	76	97	83	90	97	97	85	83

Table 3(k). Curvature representation using the multilayer ANN

Noise (%)	Similar objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	80	100
20	97	100	100	100	97	100	77	80
30	93	100	100	100	77	100	73	73
40	77	80	97	97	63	100	70	60

Table 3(l). Curvature representation using the Kohonen ANN

Noise (%)	Similar objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	93	100
20	97	100	100	100	100	100	90	100
30	93	100	100	100	100	100	87	100
40	80	100	100	100	100	100	83	87

noise corruption, the centroidal profile representation was the best for both ANNs. Specifically, the recognition of noisy objects was perfect even with 40% noise percentage, as indicated in Tables 3(a), (b), (g) and (h). Comparing the other two representations (Tables 3(c), (d), (i) and (j) vs 3(e), (f), (k) and (l)), in most cases, the curvature representation was better than the cumulative angular representation.

Evaluating the performance of the ANNs, the results were quite satisfactory. Comparing them in more detail (Tables 3(a), (c), (e), (g), (i) and (k) vs 3(b), (d), (f), (h), (j) and (l)), the Kohonen ANN in general was better than the multilayer ANN, regardless of the representation and the object set used. As it was expected, the ANN classifiers performed better, regardless of the representation used, when the objects to be recognized were dissimilar (Tables 3(a)–(f) vs 3(g)–(l)). Overall, the best object recognition approach consists of the centroidal profile representation utilizing either a Kohonen or a multilayer ANN classifier.

8. COMPARISON WITH CLASSICAL METHODS

For comparison purposes, the traditional well-known methods of Fourier descriptors[28] and invariant moments[29] utilizing the minimum distance classifier[25] were used. Simulation results were obtained in order to evaluate the performance of the Fourier descriptors and the invariant moments methods. In addition, comparisons were performed between the classical methods and the best ANN approach (centroidal profile with either a Kohonen or a multilayer ANN).

8.1. *Comparisons using the Fourier descriptors*

Fourier analysis based on contour representations has been used extensively in a number of applications.[7,28] Consider a closed contour C in the complex plane. The x–y coordinates of each point in the boundary become complex numbers $x + jy$. Tracing the boundary in a counterclockwise direction with uniform velocity, the complex function $z(t) = x(t) + jy(t)$ is obtained with parameter t. The velocity is chosen such that the time required to traverse the contour is 2π. Traversing the contour more than once yields a periodic function, which may be expanded in a convergent Fourier series. If $z(k)$ is a uniformly sampled version of $z(t)$ of dimension N, its discrete Fourier transform is given by the following equation:

$$z(k) = \sum_{n=0}^{N} a_n e^{j2\pi nk/N},$$

where a_n are the N lowest frequency coefficients defined as:

$$a_n = \frac{1}{N} \sum_{k=0}^{N} z(k) e^{-j2\pi nk/N}.$$

The Fourier descriptor (FD) of the boundary is defined as the above Fourier series.

Fourier descriptor normalization under translation can be accomplished by simply forcing a_0 to zero. Size normalization is performed by dividing all a_n by $|a_1|$. If the contour is a simple closed figure and it is traced in the counterclockwise direction, this coefficient will be the largest. The orientation and starting

point operations affect only the phases of the FD coefficients. Since there are only two allowable operations, the definition of standard position and orientation must involve the phases of at least two coefficients. Phase and starting point normalization can be accomplished by requiring that the phases of the two largest in magnitude coefficients to be zero. The coefficient with the largest magnitude value is a_1. Let the coefficient of the second largest magnitude be a_k. Normalization begins by multiplying each coefficient a_i by the form:

$$e^{[(i-k)u+(1-i)v]/(k-1)},$$

where u and v denotes the phases of a_1 and a_k, respectively. However, the requirement that a_1 and a_k should have zero phase can be satisfied by $m(k)$ different orientation/starting point combinations,[28] where $m(k) = |k-1|$. Obviously, if $k = 2$ then we have achieved a unique standard normalization. In general, a_2 will not be the second largest coefficient in magnitude, therefore additional normalization steps are required. A method[28] for solving the above ambiguities requires the determination of a third coefficient a_p, having the largest magnitude for $1 < p \leq m$ and satisfying the following restrictions: (a) $m(p) \neq m(k)$, where $m(p) = |p-1|$, (b) $m(p)$ is not a multiple of $m(k)$, (c) $m(p)$ is not a factor of $m(k)$, and (d) $m(p)$ is not a multiple of a factor of $m(k)$ less than $m(k)$. Finally, a_p replaces a_k in the normalization procedure.

In our simulations, in order to characterize each object, a subset containing the ten lowest frequency coefficients was utilized. The degree of similarity between two objects was determined by calculating the Euclidean distance between their corresponding FDs. The classification of an unknown object was performed using the minimum distance algorithm.[25] The experimental results for the two object sets are presented in Tables 4(a) and (c). Compared to the best ANN approach, the performance of the Fourier descriptors method was almost identical for the dissimilar objects set. However, the best ANN approach outperformed the FD method for the similar objects set.

Table 4(a). Fourier descriptor method

Noise (%)	Different objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100
20	100	100	100	100	100	100	100	100
30	100	100	100	100	100	100	100	100
40	100	100	100	100	100	97	97	100

Table 4(b). Invariant moment method

Noise (%)	Different objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	100	100	100	100	100	83	100	100
20	100	100	100	100	100	57	100	100
30	100	100	100	100	100	27	100	100
40	100	100	100	100	100	13	100	100

Table 4(c). Fourier descriptor method

Noise (%)	Similar objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	93	100	80	100	100	100	97	100
20	77	100	77	100	100	97	93	100
30	67	100	60	100	100	93	93	100
40	63	100	57	97	100	87	90	93

Table 4(d). Invariant moment method

Noise (%)	Similar objects 1	2	3	4	5	6	7	8
0	100	100	100	100	100	100	100	100
10	45	27	43	50	100	100	83	100
20	30	17	37	43	100	97	80	100
30	27	13	13	33	100	97	77	97
40	17	10	03	23	100	83	43	93

8.2. *Comparisons using the invariant moments*

Moments have been used as pattern features in a number of applications,[9,29] to provide invariant recognition of 2D image patterns. The regular moments m_{pq} of a digital image pattern represented by $f(x, y)$ are defined as:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y), \quad p, q = 0, 1, 2, \ldots$$

Hu[29] first introduced moments as image recognition features. Using nonlinear combinations of normalized central moments, he derived a set of seven invariant moments which has the desirable property of being invariant under image translation, scaling and rotation. Specifically, the central moments that have the property of translation invariance are given by:

$$\mu_{pq} = \sum_x \sum_y (x^p - \bar{x})(y^q - \bar{y}), \quad p, q = 0, 1, 2, \ldots,$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \bar{y} = \frac{m_{01}}{m_{00}}.$$

The following moments $\phi_1, \phi_2, \ldots, \phi_7$, are invariant under translation and rotation:

$$\phi_1 = \mu_{20} + \mu_{02}$$

$$\phi_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

$$\phi_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$$

$$\phi_4 = (\mu_{30} + \mu_{12})^2 + (3\mu_{21} + \mu_{03})^2$$

$$\phi_5 = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] + (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03}) \times [3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2]$$

$$\phi_6 = (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})$$

$$\phi_7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] - (\mu_{30} - 3\mu_{12})(\mu_{21} - \mu_{03}) \times [3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2].$$

The above moments can be normalized to become invariant under a scale change by substituting the central moments μ_{pq} with their normalized counterparts η_{pq} in the above equations. The normalized central moments η_{pq} are defined as follows:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \text{ where } \gamma = \frac{p+q}{2} + 1 \text{ for } p + q = 2, 3, \ldots$$

In our simulations each object was characterized using its invariant moments. Tables 4(b) and (d) show the experimental results obtained with the invariant moments, using the minimum distance classifier. Although its performance was at comparable levels with the Fourier descriptors method and the best ANN approach in the case of different objects, its performance was much worse in the case of the similar objects set.

9. CONCLUSIONS

In this paper, several artificial neural network based approaches for the recognition of 2D dimensional objects represented by translation, scale and rotation invariant boundary representations were introduced. Two different ANN approaches employing both supervised and unsupervised learning, represented by a multilayer ANN with two hidden layers and the Kohonen ANN respectively, were utilized. The predict back-propagation rule* was used for the training of the multilayer ANN. In addition, for comparison reasons two other traditional methods, the Fourier descriptors and the invariant moments, were also implemented. Through extensive experimentation with noiseless as well as noisy objects from two different object sets, the following conclusions were reached: the best ANN approach was the centroidal profile representation using either a multilayer or a Kohonen ANN. Furthermore, the best ANN approach outperformed both of the traditional methods. Finally, the results obtained on the different objects set were much better than the corresponding results obtained on the similar objects set.

* Any back-propagation algorithm can be utilized, since the predict back-propagation rule does not affect the accuracy of the results but only increases the training speed of the algorithm.

REFERENCES

1. D. Ballard and C. Brown, *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ (1982).
2. R. Chin and C. Dyer, Model-based recognition in robot vision, *Comput. Surv.* **18**, 67–108 (1986).
3. H. Freeman, On the encoding of arbitrary geometric configurations, *IEEE Trans. Elec. Comput.* **10**, 260–268 (1961).
   cessing. Computer Science Press, Potomac, MD (1981).
5. H. Freeman, Shape description via the use of critical points, *Pattern Recognition* **10**, 159–166 (1978).
6. P. Nahin, The theory and measurement of a silhouette descriptor for image pre-processing and recognition. *Pattern Recognition* **6**, 85–95 (1974).
7. C. Zahn and R. Roskies, Fourier descriptors for plane closed curves, *IEEE Trans. Comput.* **21**, 269–281 (1972).
8. F. Mokhtarian and A. Mackworth, Scale-based description and recognition of planar curves and two-dimensional shapes, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 34–43 (1986).
9. C. Teh and R. Chin, On image analysis by the method of moments, *IEEE Trans. Pattern Anal. Mach. Intell.* **10**, 291–310 (1988).
10. R. Kashyap and R. Chellapa, Stochastic models for closed boundary analysis: representation and reconstruction, *IEEE Trans. Inf. Theory* **27**, 109–119 (1981).
11. R. Lippman, An introduction to computing with neural nets, *IEEE ASSP Mag.*, 109–119 (April 1987).
12. G. Carpenter, Neural network models for pattern recognition and associative memory, *Neural Networks* **2**, 243–257 (1989).
13. S. Fahlman and G. Hinton, Connectionist architectures for artificial intelligence, *IEEE Comput.* 100–109 (1987).
14. T. Sejnowski and C. Rosenberg, NETtalk: a parallel network that learns to read aloud, in J. A. Anderson and E. Rosenfeld, *NEUROCOMPUTING Foundations and Research*. MIT Press, Cambridge, MA (1988).
15. A. Khotanzad and J. Lu, Classification of invariant image representations using a neural network, *IEEE Trans. Acoust. Speech Signal Process.* **38**, 1028–1038 (1990).
16. G. Papadourakis, G. Bebis and M. Georgiopoulos, Machine printed character recognition using neural networks, *Int. Neural Network Conf.*, Paris (1990).
17. D. Touretzky and D. Pomerleau, What's hidden in the hidden layers?, *Byte Mag.* 227–233 (August 1989).
18. D. E. Rumelhart, J. L. McClelland and the PDP Research Group, *Parallel Distributed Processing (PDP), Explorations in the Microstructure of Cognition*, Vol. 1: *Foundations*. MIT Press, Cambridge, MA (1986).
19. G. Bebis, G. Papadourakis and M. Georgiopoulos, Back propagation: increasing rate of convergence by predictable pattern loading, *Intell. Syst. Rev.* **1**, 14–30 (1989).
20. T. Kohonen, *Self-organization and Associative Memory*. Springer-Verlag, Berlin (1987).

21. T. Peli, An algorithm for recognition and localization of rotated and scaled objects, *Proc. IEEE* **69**, 483–485 (1981).
22. S. Dubois and F. Glanz, An autoregressive model approach to two-dimensional shape classification, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 55–66 (1986).
23. F. Attneave, Some informational aspects of visual perception, *Psychol. Rev.* **61**, 183–193 (1954).
24. H. Guggenheimer, *Differential Geometry*. Dover Publications, New York (1977).
25. J. Tou and R. Gonzalez, *Pattern Recognition Principles*. Addison-Wesley, Reading, MA (1974).
26. T. Kohonen, K. Masisara and T. Saramaki, Phonotopic maps—insightful representation of phonological features for speech representation, *Proc. IEEE 7th Int. Conf. Pattern Recognition*, Canada (1984).
27. Z. You and A. Jain, Performance evaluation of shape matching via chord length distribution, *Comput. Graphics Image Process.* **28**, 129–142 (1984).
28. T. Wallace and P. Wintz, An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors, *Comput. Vision Graphics Image Process.* **13**, 496–513 (1980).
29. M. Hu, Visual pattern recognition by moment invariants. *IRE Trans. Inf. Theory* **8**, 179–187 (1962).

About the Author—GEORGE N. BEBIS received the B.Sc. degree in Mathematics from the University of Crete in 1987. He is currently associated with the Department of Computer Science at the University of Crete and is pursuing his M.Sc. degree. His research interests include computer vision, pattern recognition, artificial neural networks and parallel computing. He is a student member of IEEE and the International Student Society for Neural Networks.

About the Author—GEORGE M. PAPADOURAKIS was born in Patras, Greece, in 1959. He received the B.Sc. degree from the Michigan Technological University in 1978, his Master's degree from the University of Cincinnati in 1981, and his Ph.D. degree from the University of Florida in 1986, all in Electrical Engineering. From 1986 until 1988 he was an Assistant Professor in the Department of Computer Engineering at the University of Central Florida, Orlando, Florida. Since 1988 he has been a Visiting Professor in the Department of Computer Science at the University of Crete, and a researcher at the Institute of Computer Science, FORTH–Hellas, Iraklion, Greece. His current research interests include artificial neural networks, digital signal processing, computer architectures, and computer graphics. He is a member of IEEE, the International Neural Networks Society, the Technical Chamber of Greece and the Computer Society of Greece.