'91 韓國自動制御學術會議論文集 1991. 10.22~24

신경회로망을 이용한 물체 인식

김 홍 봉　　남 광 희

포항공대 전자전기공학과

Object Recognition of One D.O.F. Tools By a Backpropagation Neural Network

Hong-Bong Kim　　Kwang-Hee Nam

Dept. of Electronic and Electrical Eng.　POSTECH

Abstract

We consider the object recognition of industrial tools which have one degree of freedom. In the case of pliers, the shape varies as the jaw angle varies. Thus, a feature vector made from the boundary image also varies along with the jaw angle. But a pattern recognizer should have the ability of classifying objects without any regards to the angle variation. For a pattern recognizer we have utilized a backpropagation neural net. Feature vectors were made from Fourier descriptors of boundary images by truncating the high frequency components, and they were used as inputs to the neural net for training and recognition. In our experiments, backpropagation neural net outperforms the minimum distance rule which is widely used in the pattern recognition. The performance comparison also made under noisy environments.

1. Introduction

A major task in the robot vision is the recognition of industrial parts and tools in factory environments. Since the information about the handling objects are usually known in advance, the problem of object recognition *(or pattern recognition)* turns out to be that of object classification or pattern matching.

Two major steps are involved in the object recognition. The first one is a low level vision processing in which we detect the edge or the boundary of the objects and make a feature vector for each set of visual image data. This process is often called feature extraction.

The second step is the classification of feature vectors which came out from the low level vision processing.

Most of industrial tools have one degree of freedom. For example, a plier can have different jaw angles (angle between the noses of a plier). Thus, as we vary the angle its image also varies. But, an object classifier should have the ability of telling that those images were made from one object. In this paper we are considering a method of recognizing tools which have one degree of freedom. For this purpose, we utilize a backpropagation neural net which is widely used in various fields such as robot vision, voice recognition, robotics, and so on[4,5,6,9].

In this paper we take the following procedure for object recognition: Firstly, we obtain an object image by utilizing a camera positioned above the object. Each image lies in 256×256 pixels and each pixel has 8 bit resolution. By thresholding the pixel data we obtain a binary image. With a border following algorithm we extract the edge from the binary image[13]. Then, boundary data comprises two sets of position data, i.e., x position data and y position data. Using these data we make Fourier descriptor of the object, and it was already shown that the Fourier descriptor is invariant with respect to the image rotation, translation, and size scaling [7,8,13]. That is, the Fourier descriptor gives a unique feature vector under the image transformations such as translation, rotation, and size scaling. We use Fourier descriptor as feature vector.

Secondly, we train the neural net with the feature vectors of some sample objects. Applying the feature vector of an unknown object to a trained neural net as an input, we can recognize that object.

For test objects we have used three kinds of pliers. We also compared the performances of backpropagation neural net with that of a minimum distance rule. We also checked the performance when the noise effects are significant.

2. Feature Extraction and Fourier Descriptors

For object recognition, we need a certain value, namely, a feature vector which represents the object. Since the main information of an object can be found in the boundary of the object, we utilize the boundary data of a given image for making Fourier descriptor. The reason for using Fourier descriptor as a feature vector is that it gives a unique value irrespective of the image rotation, translation, and size scaling. Another advantage of utilizing Fourier descriptor is that we can compress the data size easily by eliminating the high order components. In the following we illustrate the procedures of obtaining Fourier descriptor.

A. Data Collection

We assume that a single tool lies in each image plane, thus there is no possibility of objects' occlusion and touching.

An object image is captured in 256 × 256 pixels and each pixel has an 8 bit gray level resolution. The pixel data are stored as a 256 × 256 matrix in Sun-4/Sparc station. By thresholding the image data we obtain a binary image, from which we obtain boundary coordinates $(x[m], y[m])$ for $1 \leq m \leq L$ with the use of *boder following algorithm*[13]. Figure 1 illustrates the procedure for obtaining boundary image of a plier.

B. The Fourier Descriptors

Taking the discrete Fourier transformation of the data, $x[m]$, $y[m]$, $1 \leq m \leq L$, we obtain the Fourier coefficients for $0 \leq k \leq L-1$:

$$a[k] = \frac{1}{L} \sum_{m=1}^{L} x[m] e^{-jk(\frac{2\pi}{L})m}, \qquad (1a)$$

$$b[k] = \frac{1}{L} \sum_{m=1}^{L} y[m] e^{-jk(\frac{2\pi}{L})m}. \qquad (1b)$$

We discard the d-c components, $a[0]$ and $b[0]$ since they carry the information about the position of the image center. Note that the Fourier coefficients, $a[k]$, $b[k]$, $1 \leq k \leq L-1$ are not rotation invariant. To obtain a rotation invariance, we define $r[k]$ by

$$r[k] = \sqrt{|a[k]|^2 + |b[k]|^2}, \qquad 1 \leq k \leq L-1, \qquad (2)$$

where $|a[k]|$ denotes the absolute value of a complex number $a[k]$. Note that $|a[k]|^2$, $|b[k]|^2$ for $1 \leq k \leq L-1$ signify the energy spectrums of x and y, respectively and that $r[k]$ is invariant with respect to the image rotation and translation. In order to provide an invariance to the size scaling, we need a sort of normalization. We define $s[k]$ by

$$s[k] = \frac{r[k]}{r[1]} \quad 1 \leq k \leq L-1. \qquad (3)$$

Then $s[k]$ is invariant to the size scaling, as well as to rotation and translation.

The low frequency part of $s[k]$ determines the global shape of an object, while the high frequency part does the detailed shape. Since the high frequency part are easily degraded by noise and does not contribute much to the object recognition compared with the low frequency part, we usually discard the high frequency components. Discarding the high frequency part, we can compress the data size significantly, and this is one of the reasons for using Fourier descriptor.

The data points of the boundary image in Figure 1 amount to more than 500, i.e., $L > 500$. But, as we can see in the Table 1 the coefficients, $a[k]$, $b[k]$, $r[k]$, $s[k]$ vanish significantly as n increases. In our study, only first 16 components, $(s[1], \cdots, s[16])$ were used as a feature vector of an object among more than 500 coefficients.

3. Construction of Neural Network

Neural networks designed for classification require supervised training. Also, for the supervised training there should be a provision in the neural net for specifying class labels (or vectors) for each of training patterns. Backpropagation(BP) neural net is the most popular one among many neural net paradigms that satisfy the requirements for pattern classification[9][12].

For the effective use of the neural net, it is reasonable to bound the number of input nodes, as well as the nodes in the hidden layers. As was mentioned earlier, a feature vector is made with the first 16 components $(s[1], \cdots, s[16])$ out of more than 500 Fourier descriptors. Hence it is very natural that the input layer has 16 nodes in this work.

We have utilized three kinds of pliers as test objects for our experiment. In our experiment we choose three output nodes. We construct a BP neural net as shown in Figure 2. It has two hidden layers: the first hidden layer has 20 nodes while the second one 12. Each processing element (neuron or node) is fully connected between the adjacent layers.

We denote by (i,n) the i^{th} node of the n^{th} layer. Also we let

N_n : the total number of nodes in the n^{th} layer;

M : the total number of target vectors (output patterns or class labels);

x_{in} : output of the node (i,n);

w_{ij}^{n-1} : link weight from the node $(j,n-1)$ to the node (i,n);

θ_{in} : offset (bias) of the node (i,n);

t_{ip} : the i^{th} element of target vector, p.

The operation of the node (i,n) is characterized by

$$z_{in} = \sum_{j=1}^{N_{n-1}} w_{ij}^{n-1} x_{j\,n-1} + \theta_{in}, \tag{4}$$

$$x_{in} = f_n(z_{in}), \tag{5}$$

where x_{i1} is the i^{th} element of the input vector. Function $f_n : \mathbb{R} \to \mathbb{R}$ is chosen as

$$f(x) = \begin{cases} x, & \text{if } n = 1 \quad \text{(input layer);} \\ \dfrac{1}{1+e^{-x}}, & \text{if } n = 2,3,4 \\ & \text{(hidden layer, output layer).} \end{cases} \tag{6}$$

From the above discussion it follows that $N_1 = 16$, $N_2 = 20$, $N_3 = 12$, $N_4 = 3$, and $M = 3$ in our example.

We choose as an error function

$$E_p = \frac{1}{2}\sum_{i=1}^{N_4}(t_{ip} - x_{i4})^2, \tag{7}$$

where x_{i4} represents the i^{th} element of the actual output pattern produced by the presentation of an input. Applying the steepest decent (gradient) rule, we obtain

$$w_{ij}^n(t+1) = w_{ij}^n(t) + \eta\delta_{in+1}^p x_{jn}, \tag{8}$$

where η is a positive real constant and

$$\delta_{in}^p = \begin{cases} (1-x_{in})x_{in}\displaystyle\sum_{k=1}^{N_{n+1}} \delta_{k\,n+1}^p w_{ki}, \\ \qquad for\ \text{n=2,3}(hiddenlayers); \\ (t_{ip} - x_{i4})x_{i4}(1-x_{i4}), \\ \qquad for\ \text{n=4}(outputlayer). \end{cases} \tag{9}$$

Constant η determines the learning rate and δ_{in} represents the i^{th} element of the error in the n^{th} layer.

But, instead of (9) we utilize the following modified rule as our weight update algorithm:

$$w_{ij}^n(t+1) = w_{ij}^n(t) + \eta\delta_{in+1}^p x_{jn} + \alpha\Delta w_{ij}^n(t), \tag{10}$$

where α is a positive real constant which determines the effect of the past weight and $\Delta w_{ij}^n(t) \equiv w_{ij}^n(t) - w_{ij}^n(t-1)$ is often called *momentum term* [5]. Equation (10) can be written equivalently as $\Delta w_{ij}^n(t+1) = \eta\delta_{in+1}^p x_{jn} + \alpha\Delta w_{ij}^n(t)$.

4. Simulation Results

A. Training of the Nerual Net

Figure 3 shows the boundary images of three model objects (pliers). Four pictures were taken for each plier with different position, orientation, scale, and jaw angle. The four boundary images in each row of Figure 3 were made from the same object, but different jaw angles make differences in shape. We name the plier in the first row of Figure 3 plier 1. In the same way, we name the one in the second row plier 2 and the one in the third row plier 3. We have utilized all twelve boundary images in Figure 3 for the neural net training.

The target vectors (output patterns or class labels) are set to $(t_{11}, t_{21}, t_{31}) = (0.95, 0.05, 0.05)$ for plier 1, $(t_{12}, t_{22}, t_{32}) = (0.05, 0.95, 0.05)$ for plier 2, and $(t_{13}, t_{23}, t_{33}) = (0.05, 0.05, 0.95)$ for plier 3. The reason for not choosing $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ is that if we choose these values the training time grows indefinitely large.

Since there are twelve sets of training patterns, we evaluate an error sum by summing twelve sets of output errors. Figure 4 shows the reduction of the error sum as the training of the neural net proceeds. After 4000 iterations with feature vectors made from the samples in Figure 3, the error reduced to 0.008138. At this time we let $\eta = 0.5$ and $\alpha = 0.3$.

B. Recognition

For the performance test of the neural net we made another 20 images for each object by varying position, orientation, scale, and the jaw angle. We define an output vector $(out[1], out[2], out[3])$ by thresholding neural net outputs in such a way that for $i = 1, 2, 3$,

$$out[i] = \begin{cases} 1, & \text{if } x_{i4} \geq 0.7; \\ 0, & \text{if } x_{i4} \leq 0.3; \\ \text{unknown} & \text{if } 0.3 < x_{i4} < 0.7. \end{cases} \tag{11}$$

If $(out[1], out[2], out[3]) = (1, 0, 0)$, we regard the applied object as plier 1. In the same manner if $(out[1], out[2], out[3]) = (0, 1, 0)$, we regard the applied object is plier 2, and if $(out[1], out[2], out[3]) = (0, 1, 0)$, plier 3. There are cases which do not belong to any of the above three classes. That is, if $out[i]$ =unknown as a result of $0.3 < x_{i3} < 0.7$ for some $1 \leq i \leq 3$, then we claim that the test object is not one of the trained ones.

In pattern recognition, the minimum distance rule is widely used. We have compared the performance of the neural net with that of minimum distance rule. The minimum distance rule classifies the object based on the relative distance between the Fourier descriptors of an unknown object and the stored patterns. In our example, the procedure for classifying objects with the minimum distance rule is as follows: Firstly, we make reference patterns, F^1, F^2, F^3 by taking averages of the feature vectors $(s[1], \cdots, s[16])$ in each row of Figure 3. For a given unknown object we obtain a feature vector T, and measure the Euclidean distance D^k between T and F^k, i.e.,

$$D^k = \|T - F^k\| = \sqrt{(T - F^k)^T (T - F^k)} \tag{12}$$

$$k = 1, 2, 3.$$

If $D^1 < D^2$, D^3 then we regard the unknown object is plier 1. In the same way if $D^2 < D^1$, D^3 then we regard the unknown object is plier 2, and if $D^3 < D^1$, D^2 then plier 3.

Some of the test object images are shown in Figure 5. Table 2 shows the accuracies of recognition of the neural net and the minimum distance rule. In the case of the neural net the rate of success is 100%, while in the case of the minimum distance rule the rate of success is not 100%.

Inherently, the minimum distance rule does not have the ability of telling untrained objects from the trained ones. But, neural net can distinguish untrained objects. We also have made experiments with untrained objects. Figure 6 shows the untrained samples that we have used. Applying objects (a), (b) in Figure 6, the neural net outputs (x_{14}, x_{24}, x_{34}) =(0.436413, 0.563798, 0.000506) (x_{14}, x_{24}, x_{34}) =(0.624573, 0.401319,0.000265) respectively. According to (11), neural net output tells us that those objects are not the trained ones.

C. Recognition of Noisy Objects.

To check the performance of the neural net in the noisy environment, we made noisy objects by adding Gaussian random noise to the boundary coordinates, $x[m]$, $y[m], 1 \leq m \leq L$ of the object. Figure 7 shows the boundary images after adding Gaussian noise to those in Figure 5. We performed the experiments with 20 different test images for each plier for the cases where the noise variances are $\sigma^2 = 9$ and $\sigma^2 = 25$. The classification results are shown in Table 3. One can also notice that even when the noise effect is significant the neural net outperforms the minimum distance rule.

5. Conclusion and future research

A neural net approach for recognition of industrial tools has been proposed which have one degree of freedom. The shape of industrial tools such as plier varies as the jaw angle varies. Hence in this case, a problem is whether an object classifier can classify objects without any regards to the angle variation. We have utilized three pliers for our experiment. The results show that the trained backpropagation neural net classifies 60 test objects with zero percent error which were made by varying position, orientation, size and the jaw angle of pliers. We also checked the performance of the neural net with noisy input data. At this time, though the rate of accuracy of the neural net was not 100%, it outperforms the minimum distance rule. When an untrained object is applied, the neural net can claim it as an unknown object, while the minimum distance rule cannot. More work needs to be done concerning recognition of objects partially occluded.

References

[1] M. C. Fairhurst, Computer Vision for Robotic Systems. Cambridge: Prentice Hall, 1988.

[2] D. H. Ballard and C. M. Brown, Computer Vision. Englewood Cliffs, NJ: Prentice Hall, 1982.

[3] Berthold, Klaus, Paul and Horn, Robot Vision. Cambridge, Massachusetts: MIT Press, 1986.

[4] P. D. Wasserman, Neural Computing. NY: Van Nostrand Reinhold, 1989.

[5] D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. vol.1, Cambridge, MA: MIT Press, 1986.

[6] Robert Hecht-Nielson, Neurocomputing. :Addison Wesley, 1990.

[7] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. on Comput.*, vol.**C-21**, no.3, March 1972.

[8] E. Persoon and K. S. Fu, "Shape descrimination using Fourier descriptors," *IEEE Trans. Syst., Man, Cybern.*, vol. **SMC-7**, no.3, March 1977.

[9] Richard P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, April 1987.

[10] Alan V. Oppenheim, Alan S. Willsky and Ian T. Young, Signals And Systems.:Prentice Hall, 1983.

[11] Y. Lamdan, Jacob T. Schwartz and Haim J. Wolfson, "Affine invariant model-based object recognition," *IEEE Trans. Robotics and Automat.*, vol.**6**, no.5, October 1990, pp. 578–589.

[12] Lalit Gupta, Mohammad R. Sayeh and Ravi Tammana, "A neural network approach to robust shape recognition," *Pattern Recognition*, vol.**23**, no.6, 1990, pp. 563–568.

[13] M. Shridhar and A. Badreldin, "High accuracy character recognition algorithm using Fourier and topological descriptors," *Pattern Recognition*, vol. **17**, no.5, 1984, pp. 515–524.

[14] L. Gupta and M. D. Srinath, "Contour sequence moments for the classification of closed planar shapes," *Pattern Recognition*, vol.**20**, no.3, 1987, pp. 267–272.

[15] J. T. Tou and R. C. Gonzalez, Pattern Recognition Principles.:Addison Wesley, 1974.

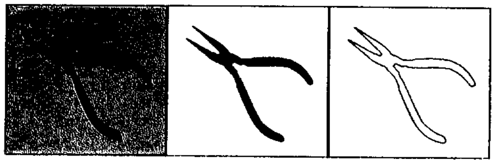


Fig. 1. Procedure for obtaining boundary image. (a) digitized image of a plier, (b) image after thresholding, (c) boundary image.

n	a[n]	b[n]	r[n]	s[n]
1	27.174304	28.716726	39.535973	1.000000
2	20.910113	16.690584	26.754596	0.676715
3	10.900999	7.807503	13.408537	0.339148
4	1.977567	2.017708	2.825229	0.071460
5	2.242817	2.320525	3.227238	0.081628
6	4.547157	3.575404	5.784475	0.146309
7	2.796431	2.463459	3.726749	0.094262
8	0.677361	0.196267	0.705223	0.017837
9	2.577942	1.967037	3.242687	0.082019
10	0.452130	0.675030	0.812457	0.020550
11	0.511461	0.714221	0.878467	0.022219
12	0.464167	0.398693	0.611888	0.015477
13	0.110593	0.298788	0.318598	0.008058
14	0.652296	0.533528	0.842691	0.021315
15	0.098410	0.065227	0.118064	0.002986
16	0.384241	0.305962	0.491176	0.012414

Table 1. Fourier descriptor of the object in Fig.1.

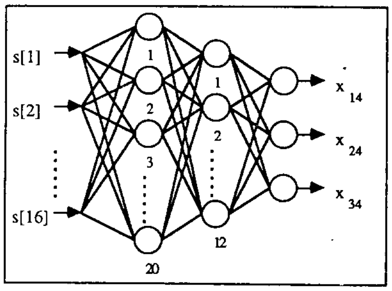


Fig. 2. Structure of two hidden layered BP neural net.

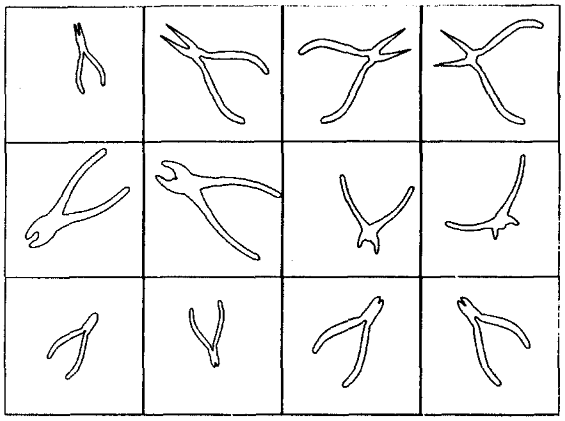


Fig. 3. Boundary images of plier 2, and plier 3 used for neural net training. The images of plier 1 lie in the first row, plier 2 in the second row, and plier 3 in third row.

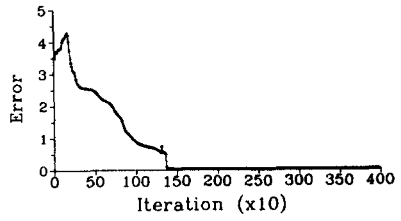


Fig. 4. Error versus iterations. After 4000 iterations, the error reduced to 0.008138.

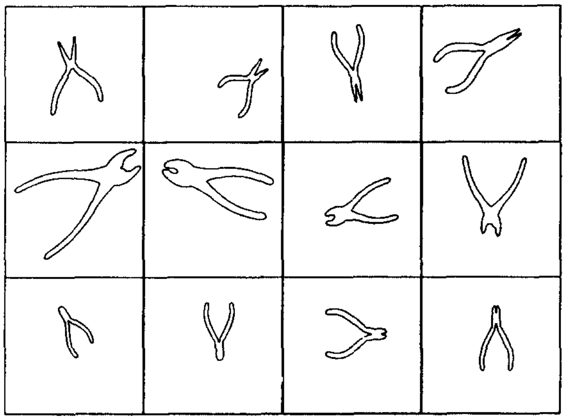
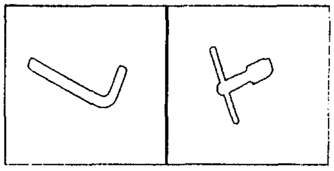


Fig. 5. Some of the test objects.

object	neural network	Minimum distance
1	100 %	85 %
2	100 %	95 %
3	100 %	100 %

Table 2. The accurate recognition rates of the neural net and the minimum distance rule.



(a) (b)

Fig. 6. The boundary images of untrained objects.

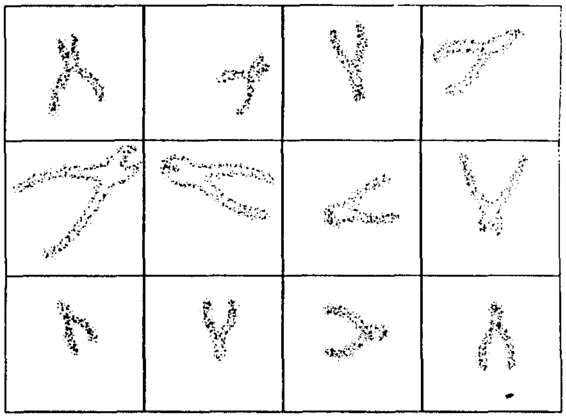


Fig. 7. Boundary images after adding Gaussian random noise(σ^2 = 9,zero mean) to the boundary images in Fig.5.

object	neural network	Minimum distance
1	95 %	80 %
2	100 %	95 %
3	100 %	100 %

(a)

object	neural network	Minimum distance
1	95 %	83.3%
2	100 %	95 %
3	100 %	100 %

(b)

Table 3. The accuracies of the neural net and the minimum distance rule (a) $\sigma^2 = 9$ and (b) $\sigma^2 = 25$.