*Pattern Recognition*, Vol. 26, No. 5, pp. 687 710, 1993
Printed in Great Britain

0031 3203/93 $6.00 + .00
Pergamon Press Ltd


# A ROTATION, SCALING, AND TRANSLATION INVARIANT PATTERN CLASSIFICATION SYSTEM

CEM YÜCEER and KEMAL OFLAZER†
Department of Computer Engineering and Information Science, Bilkent University, Bilkent, Ankara 06533, Turkey

(*Received* 25 *March* 1992; *in revised form* 21 *October* 1992; *received for publication* 28 *October* 1992)

**Abstract**—This paper describes a hybrid pattern classification system based on a pattern preprocessor and an artificial neural network classifier that can recognize patterns even when they are deformed by transformation of rotation, scaling, and translation or a combination of these. After a description of the system architecture we provide experimental results from three different classification domains: classification of letters in the English alphabet, classification of the letters in the Japanese Katakana alphabet, and classification of geometric figures. For the first problem, our system can recognize patterns deformed by a single transformation with well over 90% success ratio and with 89% success ratio when all three transformations are applied. For the second problem, the system performs very good for patterns deformed by scaling and translation but worse (about 75%) when rotations are involved. For the third problem, the success ratio is almost 100% when only a single transformation is applied and 88% when all three transformations are applied. The system is general purpose and has a reasonable noise tolerance.

Deformation invariant pattern classification  Pattern recognition  Artificial neural networks

## 1. INTRODUCTION

The recent interest in artificial neural networks, machine learning, and parallel computation has led to renewed research in the area of pattern recognition. Pattern recognition aims to extract information about the image and/or classify its contents. Systems having pattern recognition ability have many possible applications in a wide variety of areas, from simple object existence checks, through identity verification, to robot guidance in space exploration. Pattern classification, a subfield of pattern recognition, is concerned with determining whether the pattern in an input image belongs to one of the predefined classes. Early pattern classification research performed in the 1960s and 1970s focused on asymptotic properties of classifiers, on demonstrating convergence of density estimators, and on providing bounds for error rates. Many researchers studied parametric Bayesian classifiers where the form of input distributions is assumed to be known and parameters of distributions are estimated using techniques that require simultaneous access to all training data. These classifiers, especially those that assume Gaussian distributions, are still the most widely used since they are simple and are clearly described in a number of textbooks.[1,2] However, the thrust of recent research has changed. More attention is being paid to practical issues as pattern classification techniques are being applied to speech, vision, robotics, and artificial intelligence applications where real-time response with complex real world data is necessary. In all cases, pattern classification systems should be able to learn while or before performing, and make decisions depending on the recognition result.

Developing pattern recognition systems is usually a two-stage process. First the designer should carefully examine the characteristics of the pattern environment. This involves lengthy experimentation. The result is a set of features chosen to represent the original input image. Second, the designer should choose from a variety of techniques to classify the pattern which is now in feature representation. The stage of feature determination and extraction strictly determines the success of the system, since from thereon the image is represented by this feature form. Therefore, it is highly desired that the classification system itself should extract the necessary features to differentiate the example patterns that represent each class. In other words, the system should be automated to work by itself and should not depend on the human designer's success in defining the features. Further, these features should be chosen such that they should tolerate the differentiation between the patterns in the same class. The system should also have the ability to perform the classification in a rotation, scaling, and translation invariant manner. This effect is typical when the scanning device, suppose a camera, changes its orientation or distance from the specimen. Hence the image fed to the system may contain a pattern that is rotated, scaled, or translated compared to its original form when it was first presented to the system. For such a case, either the system should employ features that are invariant to such transformations or there should be a preprocessor to maintain the rotational, scaling, and translation invariancy. Even for a limited system designed for classifying only a deter-

† Author to whom all correspondence should be addressed.

mined type of patterns—an optical character classifier, or an identity verifier—it is hard to find features that extract useful information while maintaining the mentioned invariances. The problem will be impractical if such a system is intended for general purpose classification to classify any type of patterns.

The scope of this work has been to develop a general purpose pattern classification system which can classify patterns even if they are deformed by transformations like rotation, scaling, and translation or a combination of them. The system is based on a preprocessor front-end and an artificial neural networks back-end. Artificial neural networks have recently been used for automatic feature extraction and/or pattern classification mainly owing to their learning algorithms, generalization ability and noise tolerance.[3–8]

The paper is organized as follows: after a brief overview of the pattern classification problem we overview the relevant work in pattern classification and introduce some concepts from artificial neural networks. We then describe in detail the structure of the proposed pattern classification system and give experimental results on three different pattern classification domains: the English alphabet, the Japanese Katakana alphabet, and set of five geometric symbols. The summary and conclusions are followed by two appendices including the mathematical derivations of the two groups of formulae given in the text.

## 2. PATTERN RECOGNITION AND NEURAL NETWORKS

Pattern recognition deals with the identification or interpretation of the pattern in an image. It aims to extract information about the image and/or classify its contents. For some simple and frequently encountered patterns, the recognition process can be a straightforward task. However, when patterns are complex or when pattern characteristics cannot be predicted beforehand, then one needs a high-level system to perform pattern recognition. The problem attempted in this work is a subclass of the general pattern recognition problem. The aim is to classify the pattern in an input image using the information that was extracted from the example patterns previously supplied to the system. Inputs are in the form of digitized binary-valued two-dimensional (2D) images containing the pattern to be classified. This representation of the 2D image is defined and used throughout the text as the pixel-map form of the image.

### 2.1. *Overview of pattern classification*

Pattern classification is concerned with determining whether the pattern in an input image belongs to one of the predefined classes. There are two major pattern classification techniques: template matching and feature extraction. Template matching compares the pixel-map of the test pattern with a number of stored pixel-maps until an exact match or a match with *tolerable* error is found. It is a top-down process in the sense that the trial procedure does not depend on the test pattern in any way. On the other hand, feature extraction is the process of converting the pixel-map representation of the input image of a group of quantitative descriptors called feature values. These features are usually predefined by the designer and chosen to be independent from each other.[4,5,7] Pattern classification using feature extraction usually starts by detection of some limited number of features. These features are chosen to distinguish most of the previously stored patterns. If a non-acceptable result is obtained, classification continues with some other features until a unique decision is made. It is important to note that the feature extraction process which converts the input image to a set of quantitative values should preserve the discrimination information throughout and after the conversion. In order to attain a high success ratio in classification, features should satisfy the following two requirements:[7]

- *Small intraclass invariance.* Patterns with similar shapes and similar general characteristics should end up with numerically close numbers for the features.
- *Large interclass separation.* Patterns from different classes should evaluate to features which have quite different quantitative values. In other words, the patterns from different classes should differ in one or more features so that discrimination can be made.

Template matching gets both computation and memory intensive when the resolution of the stored patterns increases, or when the patterns get more complex. In addition, template matching is sensitive to the exact orientation, size, and location of the object unless rotation, scaling, and translation invariant autocorrelation techniques are used. Consider the case of complex patterns which are 2D perspective projections of three-dimensional (3D) objects, the patterns will be highly effected by the orientation and position of the objects, hence some form of generalized template matching will be required.

The computation and memory requirements for classification with feature extraction are less severe than template matching. In most applications, computers having moderate computing power are employed to generate the Fourier descriptors of the perimeter lines in a silhouette of the pattern. For higher level tasks, artificial intelligence techniques are used in analyzing the information embedded in the skeletonized form of the patterns. The fundamental problem with feature extraction is that important information may be lost in the extensive data reduction at the feature extraction stage.

### 2.2. *Pattern classification with neural networks*

There are two main approaches to pattern classification using artificial neural networks. The first approach uses feature extraction and then employs artificial neural networks to perform the classifica-

tion on the feature values. Examples of such work are extraction of image information using regular moments,[9] Zernike moments,[7] Fourier descriptors,[9-12] autoregressive models,[13] image representation by circular harmonic expansion,[14] syntactic pattern recognition applications,[15] Karhunen–Loéve expansion,[16] polar-coordinate Fourier transform,[17] transforming the image to another 2D representation.[18] Kollias *et al.* have transformed the input image to another 2D representation, called $(a, b)$ plane, and performed classification using higher-order networks.[18] Khotanzad *et al.* have used Zernike moments to achieve rotational invariancy in classification.[7] They have computed some finite number of Zernike moments of the given image and performed classification on these moment values. Le Cun *et al.* first skeletonized the input patterns and then scanned the whole image with a $7 \times 7$ window, with the window templates designed to detect certain predefined features.[4] The classification is performed on the computed feature values. Kirby *et al.* used the Karhunen–Loéve expansion of the input image as its feature representation and performed classification on the expansion.[16]

A second and newly developing approach lets the features be determined and extracted by the artificial neural network itself. Martin *et al.* fed a neural network size-normalized gray scale image and report that "the generalization performance is surprisingly insensitive to characteristics of the network architecture, as long as enough training samples are used and there is sufficient capacity to support training to high levels".[19]

### 2.3. *Overview of artificial neural networks*

Artificial neural networks are computational models inspired by the structure of the human brain. They are massively parallel networks comprising a large number of simple processing units, called artificial neurons. The neuron structure given in Fig. 1 is the basic building block of such networks. It performs a weighted summation over its inputs, which may be the output of other neurons or may be coming from the external environment. The threshold, a local value for each neuron, is added to the sum. Then an activation function—also called the limiting or squashing function—is applied on the resulting sum to determine the neuron's output value. Various sigmoids, and hard-limiting (thresholding) functions are widely used activation function types, some of which are shown in Fig. 2.

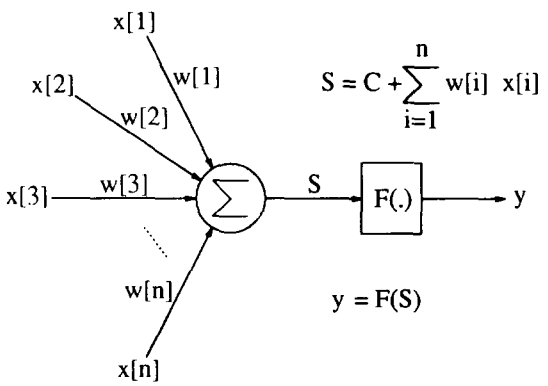



Fig. 1. Structure and function of a single artificial neuron.

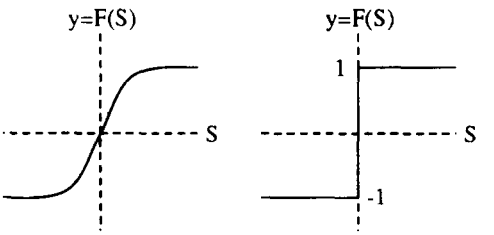



Fig. 2. Some activation functions for the artificial neuron.

The weights associated with the inter-neuron connections represent the influence of its inputs over its output value. Depending on the sign of the weight an input may *excite* or *inhibit* the neuron. Most neural network architectures learn to respond to their inputs correctly by going through a training process during which the weights between the neurons are adjusted.

The most common of the architectures of artificial neural networks is the multilayer feed-forward network architecture where neurons are grouped as layers and connections between neurons in consecutive layers are permitted. One end of the layered structure is called the input layer, while the other end the output layer. The inputs are fed from the input layer and the outputs are produced from the output layer. The hidden layer(s) between the input and output layers extract salient features from the input data and develop internal representations for those relevant features.

## 3. THE PATTERN CLASSIFICATION SYSTEM

Feed-forward type neural networks, when used as pattern classifiers directly, are highly sensitive to transformations like rotation, scaling and translation in the input. This behavior emerges from the fact that throughout the training phase the area of interest that the network mainly concentrates on is the region where the pattern lies. Hence the weights associated with the input pixels that are out of this region decay to zero during training. Although the neurons in the first layer perform a weighted summation over the values of all pixels, these null weights block the information that may arise from the corresponding pixels. Thus a transformation that pushes the pattern out of this region, degrades the classification performance of the network dramatically. Therefore, in order to obtain a high success ratio in classification the system should provide rotation, scaling, and translation correction before attempting to classify. Such invariancy can be achieved by a preprocessor with the following properties:

- The preprocessor should map the patterns that are distorted by transformations or noise to a reasonably stable output pattern.

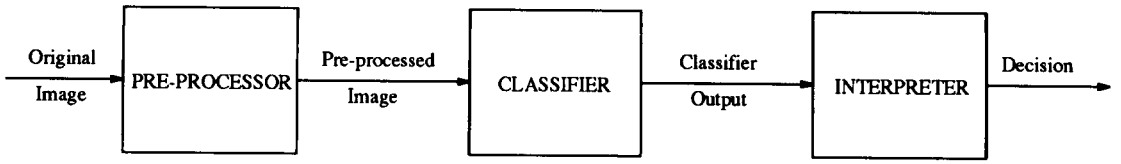



Fig. 3. Block diagram of RST, the pattern classification system.

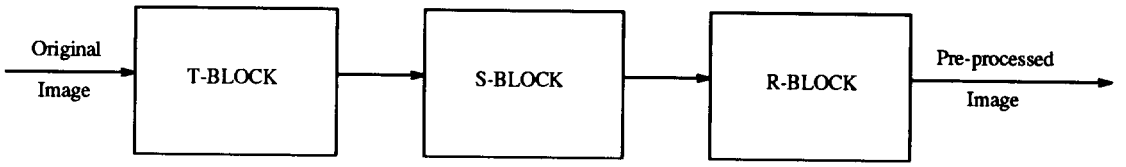



Fig. 4. Block diagram of the preprocessor.

• The mapping algorithm should be easy to compute. That is, it should not increase the overall complexity of the recognition system. A preprocessor that is much slower than the classifier and/or the interpreter would turn out to be a bottleneck increasing the classification time per pattern.

Selecting good features for relatively complex patterns, like the human face or finger prints, turns out to be impractical or even impossible.[16] The problem is more acute when there is no prior knowledge on the patterns to be classified. Therefore, a system to automatically extract the useful features is essential. Artificial neural networks extract information during the training process.[20] The hidden layers and the neurons in the hidden layers detect the relevant features in the images.

The pattern classification system introduced in this work, RST, has a modular structure consisting of three main blocks, a preprocessor, a classifier, and an interpreter. The blocks are cascaded in order such that the original image is first preprocessed, then classified, and finally the results are interpreted. Figure 3 shows the block diagram for the pattern classification system.

### 3.1. *PREP1: the preprocessor with radial scaling correction*

The function of the preprocessor is to provide rotational, scaling, and translation invariancy on the input image. Both the input and the output of the preprocessor are images in pixel-map form. The preprocessor has three cascaded blocks R-Block, S-Block, and T-Block. The block diagram of the system is given in Fig. 4. The R-Block maintains rotational invariancy, the S-Block maintains scaling invariancy, T-Block maintains translational invariancy.

The order in which the blocks are cascaded is determined mainly by the functional dependencies between these blocks. In the first implementation of the preprocessor, PREP1, the T-Block comes first, the S-Block second, and R-Block last. Since the scaling and rotation operations need a proper pivot point to function on, the T-Block is positioned before the two blocks. The origin of the pixel-map output by this block will be the pivot point for the scaling and rotation blocks. Further, placing the S-Block in front of the R-Block will bring the following two advantages:

• The S-Block will prevent the pattern from flowing out of the image by a rotation operation. Consider an image containing a line pattern where the line diagonally extends between the corner points. Since our grid is of rectangular type, not a circular grid, the two ends of this pattern will flow out of the image after a rotation of 45 deg. However, performing scaling correction beforehand would establish a radial boundary for the pattern.

• The S-Block will adjust the number of pixels that are on-pixels and will regulate the information flowing into the R-Block. The performance of the R-Block degrades when the number of on-pixels is only a small portion of the total number of pixels. Hence, performing scaling correction will enable a better performance in providing rotational invariancy.

3.1.1. *The T-Block.* The T-Block maintains translational invariancy by computing the center of gravity of the pattern and translating the image so that the center of gravity coincides with the origin. The resulting image is passed to the S-Block. The center of gravity is computed by averaging the $x$ and $y$ coordinates of the on-pixels, as formulated below.

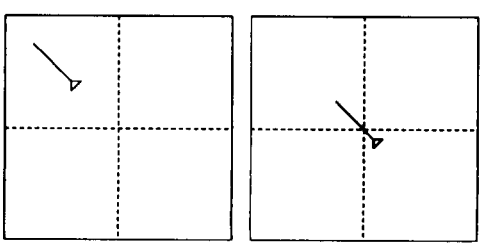


Fig. 5. A sample pattern before and after the T-Block.

Define $P$ to be the number of on-pixels:

$$P = \sum_{i=1}^{N} \sum_{j=1}^{N} f(x_i, y_j). \tag{1}$$

Then the center of gravity, $(x_{av}, y_{av})$, will be

$$x_{av} = \frac{1}{P} \sum_{i=1}^{N} \sum_{j=1}^{N} f(x_i, y_j) \cdot x_i, \quad y_{av} = \frac{1}{P} \sum_{i=1}^{N} \sum_{j=1}^{N} f(x_i, y_j) \cdot y_j \tag{2}$$

where function $f(x, y)$ gives the value of the pixel at the coordinates $(x, y)$. For digitized binary-valued 2D images this function will be either 0 or 1.†

The mapping function for the translation invariant image is

$$f_T(x_i, y_j) = f(x_i + x_{av}, y_j + y_{av}). \tag{3}$$

Figure 5 shows the function of T-Block on a sample pattern.

3.1.2. *The S-Block.* The S-Block maintains scaling invariancy by scaling the image so that the average radius for the on-pixels is equal to one-fourth of the grid size. The term radius for a pixel is defined to be the length of the straight line connecting the pixel and the origin. The scaling process will bring a radial boundary to the pattern in the image while adjusting the number of on-pixels. It thus disables any possible pattern deformation caused by rotation. The average radius is computed as:

$$r_{av} = \frac{1}{\sum_{i=1}^{N} \sum_{j=1}^{N} f_T(x_i, y_j)} \sum_{i=1}^{N} \sum_{j=1}^{N} f_T(x_i, y_j) \cdot \sqrt{(x_i^2 + y_j^2)} \tag{4}$$

and the scale factor, $s$, given by

$$s = \frac{r_{av}}{R} \tag{5}$$

where $R$ is equal to one-fourth of the grid size.

The mapping function for the scaling invariant image is

$$f_{TS}(x_i, y_j) = f_T(s \cdot x_i, s \cdot y_j). \tag{6}$$

Figure 6 shows the function of the S-Block on the sample image processed by the T-Block.

Equation (6), the mapping function for the S-Block, embeds an interpolation property. In this equation, the pixels of the output image are mapped back to their corresponding pixels in the input image. This is called *reverse mapping* and brings the interpolation property. Consider a case where the *forward mapping* technique is used, and two on-pixels that are adjacent in the input image are mapped to two apart pixels in the output image. There would be a discontinuity between these on-pixels in the output image. However, using *reverse mapping* both the apart pixels and the pixels between them are mapped back to one of the original pixels, thus maintaining the connectivity of the pattern. A sketch illustrating this discussion is given in Fig. 7.

† If $x$ or $y$, the arguments of the function, are not integers then they are rounded to the nearest integer to obtain the pixel coordinates.

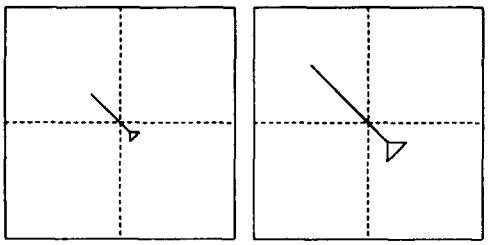


Fig. 6. The sample pattern before and after the S-Block.

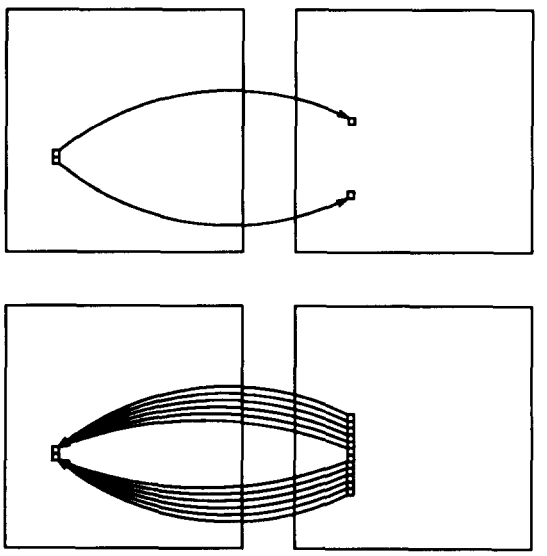


Fig. 7. The *forward mapping* technique and the interpolation property by the *reverse mapping* technique used in the S-Block.

3.1.3. *The R-Block.* The R-Block maintains rotational invariancy by rotating the image so that the direction of maximum variance coincides with the $x$-axis. The derivation of the function is based on the Karhunen–Loéve transformation which has been used in some applications.[8,16,21] The transformation exploits the following: given a set of vectors, the eigenvector that corresponds to the largest eigenvalue of the covariance matrix calculated from the set of vectors, points in the direction of maximum variance.[16,21] This property can be used to maintain rotational invariancy since detection of the maximum variance direction will also reveal the rotation angle. A general solution for any size of vectors would be impractical. However, for 2D vectors formed by the $x$ and $y$ coordinates of the on-pixels, the eigenvalues are easy to compute and a formula for the eigenvector corresponding to the greatest eigenvalue can be derived from the 2 by 2 covariance matrix. The rotation parameters are derived as follows.

Define:

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix} \tag{7}$$

$$P = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j) \tag{8}$$

$$T_{xx} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)\cdot x_i^2, \quad T_{yy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)\cdot y_j^2 \tag{9}$$

$$T_{xy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)\cdot x_i \cdot y_j. \tag{10}$$

The covariance matrix defined as

$$C = \frac{1}{P}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right)\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right)^{T} \tag{11}$$

can be simplified to

$$C = \left(\frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j) \times \begin{bmatrix} x_i \\ y_j \end{bmatrix}\begin{bmatrix} x_i \\ y_j \end{bmatrix}^{T}\right) - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\begin{bmatrix} m_x \\ m_y \end{bmatrix}^{T}. \tag{12}$$

Since translation invariancy has been maintained, the averages $m_x$ and $m_y$ are zero. Furthermore, the averaging term in front of the matrix can be eliminated since it does not change the direction of the eigenvectors. Therefore, the covariance matrix becomes

$$C = \begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix}. \tag{13}$$

Finally (detailed derivation is given in Appendix A), the sine and cosine of the rotation angle come out as

$$\sin\theta = \frac{(T_{yy} - T_{xx}) + \sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)}}{\sqrt{(8\cdot T_{xy}^2 + 2\cdot(T_{yy} - T_{xx})^2 + 2\cdot(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)})}} \tag{14}$$

$$\cos\theta = \frac{2\cdot T_{xy}}{\sqrt{(8\cdot T_{xy}^2 + 2\cdot(T_{yy} - T_{xx})^2 + 2\cdot(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)})}}. \tag{15}$$

The mapping function for the rotation invariant image is

$$f_{TSR}(x_i, y_j) = f_{TS}(\cos\theta\cdot x_i - \sin\theta\cdot y_j, \sin\theta\cdot x_i + \cos\theta\cdot y_j). \tag{16}$$

Figure 8 shows the function of the R-Block on the sample pattern processed by the S-Block.

It should be noted that the R-Block rotates a pattern until the computed orientation coincides with the $x$-axis. Since a pattern and its 180° rotated version will have the same orientation of maximal variance, the R-Block will not be able to differentiate between them and will apply the same angle of rotation on both patterns. The resulting mappings will conserve this 180° angle difference. Hence depending on its original orientation, a given pattern will be mapped to one of the two canonical patterns. These two canonical patterns will both represent the class. Hence for each original pattern we have two preprocessor outputs which are used in training the network.

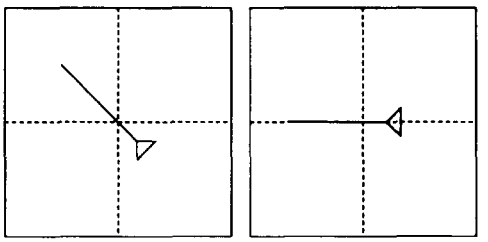


Fig. 8. The sample pattern before and after the R-Block.

### 3.2. *PREP2: the preprocessor with axial scaling correction*

PREP1, performs radial scaling correction which uses the same scaling factor along all directions. A different approach is to use different scaling factors along different axes. This type of scaling correction performs better in certain applications. For example, the patterns may have been scanned in with different resolutions in different dimensions. The preprocessor with axial scaling correction, called PREP2, has two main differences from the previous version:

- The main blocks are reordered, such that the T-Block comes first, the R-Block is second, and the S-Block is last.
- The scaling factors are computed using a different function.

In order to maintain rotational, scaling, and translational invariancy PREP2 computes various relevant parameters as (detailed derivation is given in Appendix B)

$$T_{xx} = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot x_i^2\right) - P\cdot x_{av}^2 \tag{17}$$

$$T_{yy} = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot y_j^2\right) - P\cdot y_{av}^2 \tag{18}$$

$$T_{xy} = \left( \sum_{i=1}^{N} \sum_{j=1}^{N} f(x_i, y_j) \cdot x_i \cdot y_j \right) - P \cdot x_{av} \cdot y_{av} \tag{19}$$

$$\sin\theta = \frac{(T_{yy} - T_{xx}) + \sqrt{((T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2)}}{\sqrt{(8 \cdot T_{xy}^2 + 2 \cdot (T_{yy} - T_{xx})^2 + 2 \cdot (T_{yy} - T_{xx}) \sqrt{((T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2)})}} \tag{20}$$

$$\cos\theta = \frac{2 \cdot T_{xy}}{\sqrt{(8 \cdot T_{xy}^2 + 2 \cdot (T_{yy} - T_{xx})^2 + 2 \cdot (T_{yy} - T_{xx}) \sqrt{((T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2)})}}. \tag{21}$$

$$s_x = \sqrt{\left( \frac{T_{xx}(\cos\theta)^2 + 2 \cdot T_{xy} \cdot \cos\theta \cdot \sin\theta + T_{yy}(\sin\theta)^2}{R_x \cdot P} \right)} \tag{22}$$

$$s_y = \sqrt{\left( \frac{T_{xx}(\sin\theta)^2 + 2 \cdot T_{xy} \cdot \cos\theta \cdot \sin\theta + T_{yy}(\cos\theta)^2}{R_y \cdot P} \right)} \tag{23}$$

where $s_x$ and $s_y$ are the scaling factors, and $R_x$ and $R_y$ are the desired deviation values along the corresponding axes. $R_x$ and $R_y$ are equal to the grid size.

The mapping function for the preprocessor with axial scaling correction is

$$f_{TRS}(x_i, y_j) = f(\cos\theta \cdot s_x \cdot x_i - \sin\theta \cdot s_y \cdot y_j + x_{av}, \sin\theta \cdot s_x \cdot x_i + \cos\theta \cdot s_y \cdot y_j + y_{av}). \tag{24}$$

Note that, for PREP1 the complete mapping function could be obtained only after two passes over the original image. However, for PREP2 the whole mapping function can be computed after a single pass over the original image.

### 3.3. *The classifier*

The current implementation of the system employs a multilayer feed-forward network for the classifier block. Such a network has a layered structure and only connections between neurons at subsequent layers are permitted. The training algorithm is the widely used *backpropagation algorithm*.[20] Since the input is an image in pixel-map form, the number of nodes in the input layer is fixed and equal to the number of pixels. Further, since the output neurons are organized such that each node represents a class, the number of output neurons is also fixed. Hence, given the input and output layer size one should decide on the number of hidden layers and the number of neurons in each hidden layer as well as the learning rate. In fact, this choice of network size is as critical as the choice of the learning rate. The decision depends on the type of problem dealt with, and on the size and variety of the example patterns. There are two useful rules to have in mind:

- One should choose the network large enough so that the neurons can develop features to distinguish the patterns belonging to different classes, while as

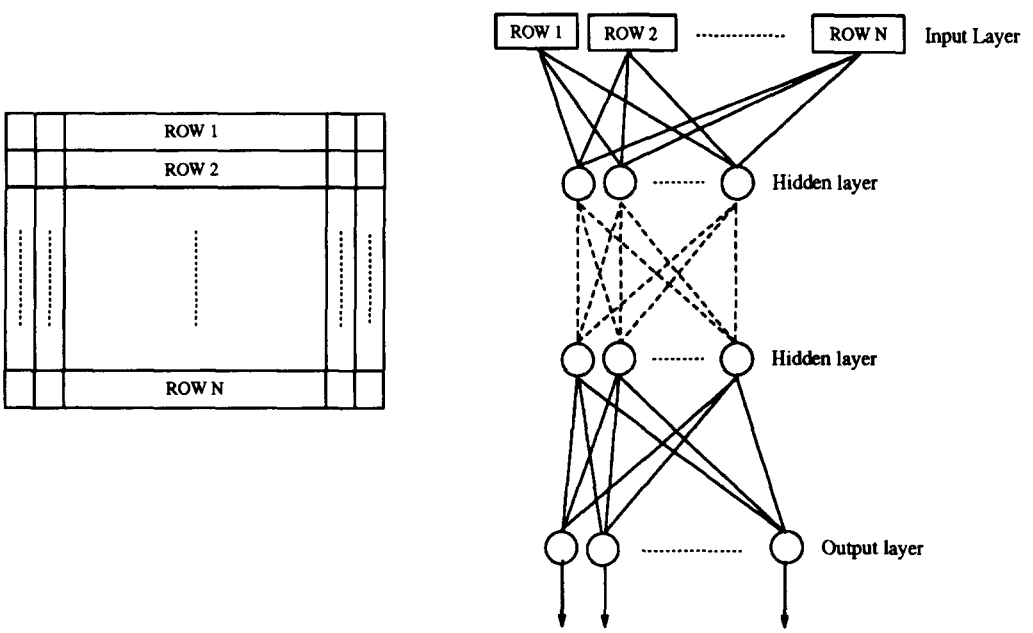



Fig. 9. The structure and image feeding strategy to the classifier.

compact as possible to avoid memorization of the example patterns.

- One should choose the learning rate small if two or more of the example patterns are similar, and large otherwise.

The general structure of the neural network classifier is given in Fig. 9. The output of the preprocessor, which is an image in pixel-map form, is converted to a linear array by cascading the rows of the image from the top row to the bottom row. The content of each array entry is the initial input value of the corresponding input node.

The classifier is trained as follows: the preprocessed forms of the images containing the sample patterns from each class are fed to the network, while the desired output classification is supplied to the output neurons. The artificial neural network learns to produce the desired outputs when the sample inputs are given to the system using the backpropagation algorithm. In the classification mode, the preprocessed image to be classified is presented to the network which then activates the appropriate output neurons.

### 3.4. *The interpreter*

One of the key determinants of the system performance is the success in interpretation of the classifier outputs. Since we have employed an artificial neural network for the classifier, the classification result will be in the form of activation values of the neurons in the output layer.

It is hard to decide on the method to be used so that none of the information that serves to distinguish the classes is ignored. The first alternative is to use a simple maximum finder block for the interpreter. However, the performance would be moderate since it will always decide on one of the classes whether the class chosen is dominant on others or not. Thresholding can be applied to the outputs so that a class is selected if the outputs exceed this predetermined threshold. However, this method may indicate multiple classes for certain inputs. A more promising method is to report *no discrimination* as long as the ratio of the maximum output to the next highest output remains under a predetermined threshold value. When the ratio exceeds the threshold, which means that the maximum output is dominant on the other outputs, the interpreter decides on the class with the maximum output. The interpreter block of our system employs this last approach. If the ratio of the maximum output to the next highest output exceeds the predetermined threshold, the interpreter reports that a discrimination could be made and the pattern belongs to the class with the maximum output. If not, then the interpreter reports that no unique discrimination could be made. This simple method has been observed to perform well in the evaluation of the classifier outputs.

### 3.5. *Computational requirements*

The computational requirements for the system presented above can be characterized as follows. Let $G$, $P$, $H$, and $O$ be the number of pixels in the image, the number of on-pixels in the image, the number of hidden units in the one-hidden-layer neural network and the number of output units, respectively. Ignoring any computation that does not depend on one of these and assuming that we have pixels either 0 or 1, for PREP1, we can write the total computational requirements as follows. The T-Block requires about $2P$ integer additions, $2G$ integer subtractions. The S-Block requires $2P$ integer multiplications, $2P$ integer additions and $P$ floating point square roots, and $2G$ floating point multiplications. The R-Block requires $3P$ integer multiplications, $3P$ integer additions, and $4G$ floating point multiplications and $2G$ floating point additions. The neural networks trained typically have very sparse weight matrices and this fact can be exploited. The forward pass through the network takes $P*H + H*O$ floating point multiplications and additions, and $H + O$ sigmoid function invocations. The overall process takes about:

- $9P + 2G$ integer additions;
- $5P$ integer multiplications;
- $2G + P \cdot H + H \cdot O$ floating point additions;
- $6G + P \cdot H + H \cdot O$ floating point multiplications;
- $P$ floating point square root computations; and
- $H + O$ sigmoid function computations.

Obviously the time parameters corresponding to these operations will change from system to system. On our Sparcstation environment we have noted a speed of about 7–10 character recognitions per second for the letter recognition problem discussed later. The requirements for the PREP2 case can be similarly derived.

## 4. EXPERIMENTAL RESULTS

Based on the formulations described in the previous sections, a general purpose pattern classification system has been developed on Sun Sparc workstations using the graphics environment SunView.[22,23] We have experimented with three problem domains: character recognition on the English alphabet, character recognition on the Japanese Katakana alphabet and, recognition of geometric objects.

### 4.1. *Character recognition on the English alphabet*

This classical problem is the classification of letters in the English alphabet. The artificial neural network chosen for classifying the English alphabet is a multi-

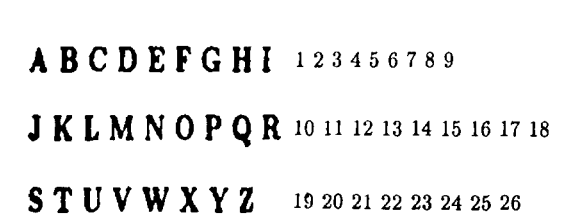
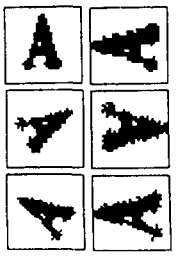


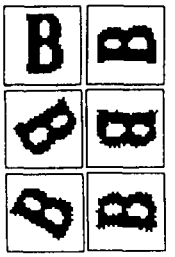

Fig. 10. The example patterns and corresponding class numbers for the 26 English letters.



Pattern is A with 0.927701
Candidate was Y with 0.021057
Discrimination ratio is 440.6 %

Pattern is A with 0.772268
Candidate was V with 0.053143
Discrimination ratio is 145.3 %

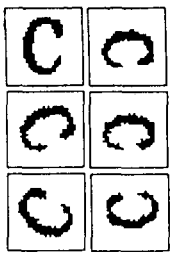
Pattern is A with 0.722949
Candidate was V with 0.029526
Discrimination ratio is 244.8 %



Pattern is B with 0.906775
Candidate was D with 0.030358
Discrimination ratio is 298.7 %

Pattern is B with 0.894358
Candidate was D with 0.036271
Discrimination ratio is 246.6 %

Pattern is B with 0.797724
Candidate was E with 0.026937
Discrimination ratio is 296.1 %

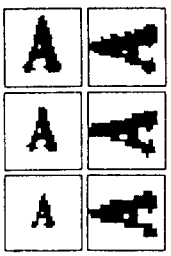


Pattern is C with 0.936130
Candidate was U with 0.031960
Discrimination ratio is 292.9 %

Pattern is C with 0.900459
Candidate was U with 0.030606
Discrimination ratio is 294.2 %

Pattern is C with 0.872540
Candidate was U with 0.032756
Discrimination ratio is 266.4 %

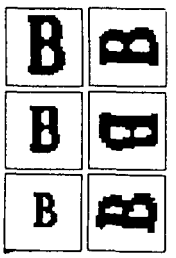
Fig. 11. Classification results with PREP1 for letters A, B, and C rotated by 0, 60, and −60 deg.



Pattern is A with 0.927701
Candidate was Y with 0.021057
Discrimination ratio is 440.6 %

Pattern is A with 0.605121
Candidate was V with 0.040005
Discrimination ratio is 151.3 %

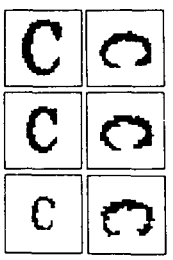
Pattern is A with 0.501417
Candidate was K with 0.168918
Discrimination ratio is 29.7 %



Pattern is B with 0.906775
Candidate was D with 0.030358
Discrimination ratio is 298.7 %

Pattern is B with 0.917553
Candidate was D with 0.075526
Discrimination ratio is 121.5 %

Pattern is B with 0.733702
Candidate was E with 0.061952
Discrimination ratio is 118.4 %

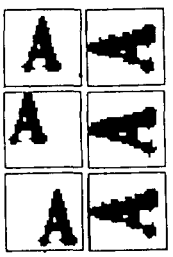


Pattern is C with 0.936130
Candidate was U with 0.031960
Discrimination ratio is 292.9 %

Pattern is C with 0.836518
Candidate was U with 0.038429
Discrimination ratio is 217.7 %

Pattern is C with 0.789987
Candidate was L with 0.041864
Discrimination ratio is 188.7 %

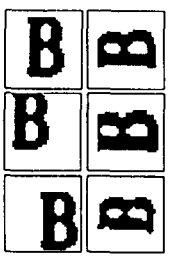
Fig. 12. Classification results with PREP1 for letters A, B, and C scaled by a factor of 1, 0.8, and 0.6.



Pattern is A with 0.927701
Candidate was Y with 0.021057
Discrimination ratio is 440.6 %

Pattern is A with 0.770125
Candidate was R with 0.041848
Discrimi ation ratio is 184.0 %

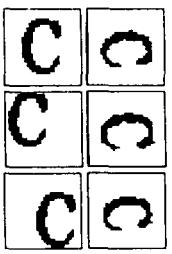
Pattern is A with 0.927701
Candidate was Y with 0.021057
Discrimination ratio is 440.6 %



Pattern is B with 0.906775
Candidate was D with 0.030358
Discrimination ratio is 298.7 %

Pattern is B with 0.514692
Candidate was F with 0.029781
Discrimination ratio is 172.8 %

Pattern is B with 0.848754
Candidate was D with 0.099396
Discrimination ratio is 85.4 %



Pattern is C with 0.936130
Candidate was U with 0.031960
Discrimination ratio is 292.9 %

Pattern is C with 0.707669
Candidate was U with 0.042863
Discrimination ratio is 165.1 %

Pattern is C with 0.905177
Candidate was O with 0.019897
Discrimination ratio is 454.9 %

Fig. 13. Classification results with PREP1 for letters A, B, and C translated diagonally by 0, 6, and −6 pixels.

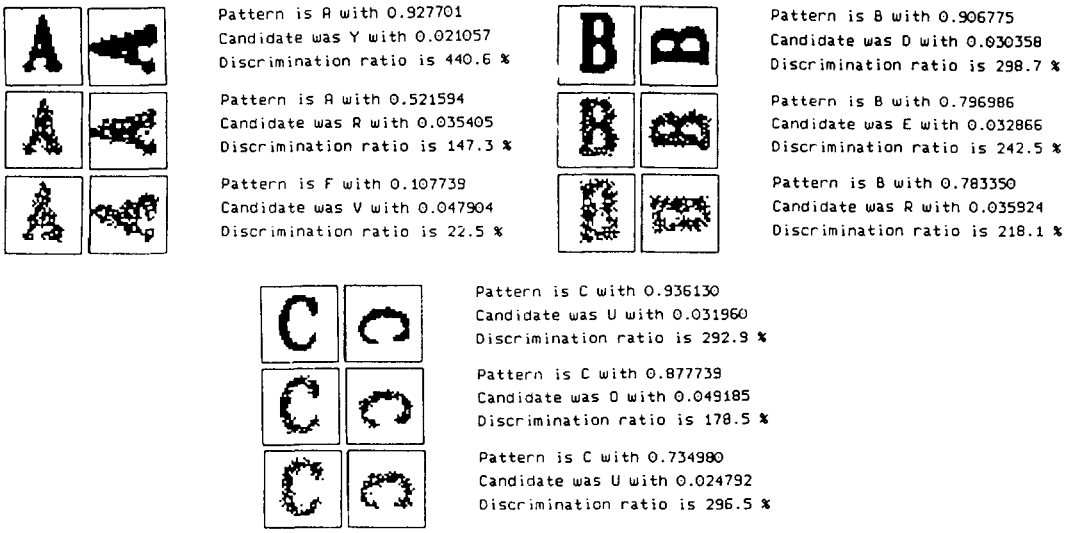



Fig. 14. Classification results with PREP1 for letters A, B, and C with 0, 20, and 40% noise.

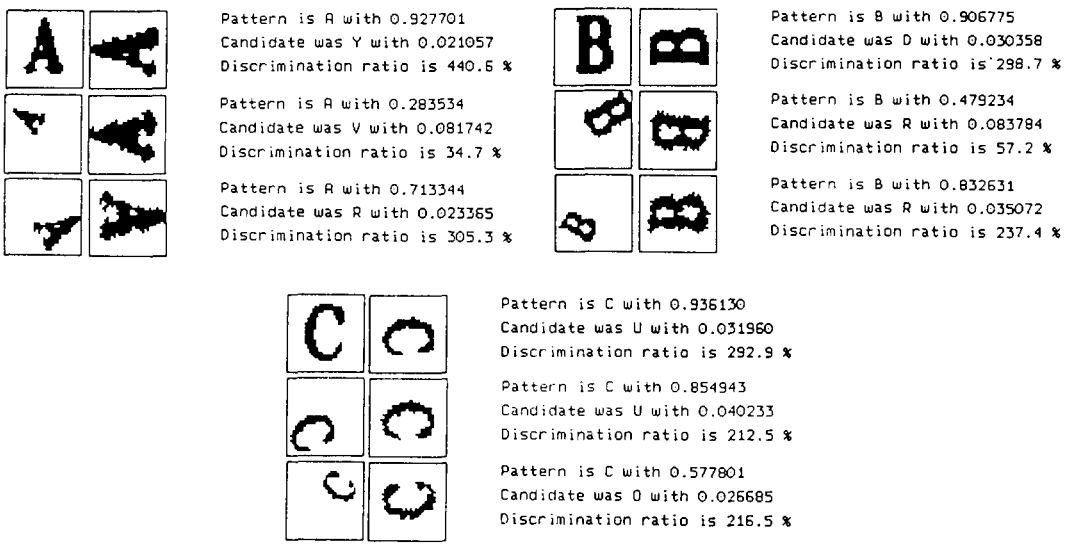



Fig. 15. Classification results with PREP1 for letters A, B, and C with random translation, scaling, and rotation applied.

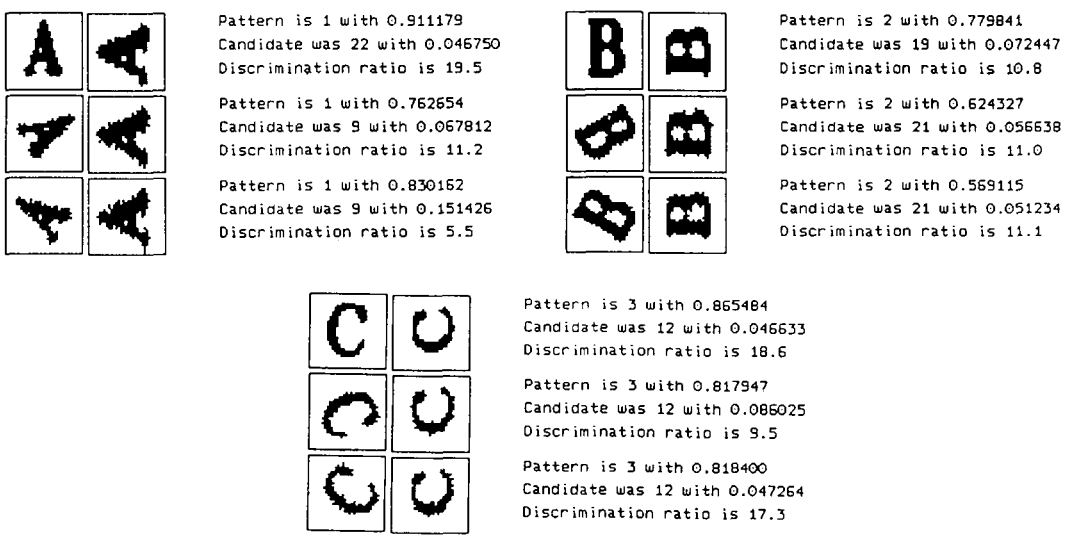
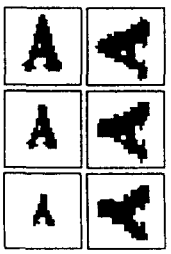


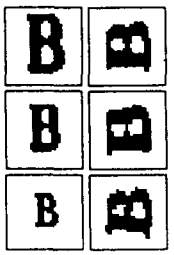

Fig. 16. Classification results with PREP2 on rotated letters.



Pattern is 1 with 0.911179
Candidate was 22 with 0.046750
Discrimination ratio is 19.5

Pattern is 1 with 0.770966
Candidate was 22 with 0.062864
Discrimination ratio is 12.3

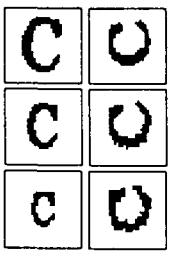
Pattern is 1 with 0.425590
Candidate was 25 with 0.185589
Discrimination ratio is 2.3



Pattern is 2 with 0.779841
Candidate was 19 with 0.072447
Discrimination ratio is 10.8

Pattern is 2 with 0.161977
Candidate was 4 with 0.078825
Discrimination ratio is 2.1

Pattern is 2 with 0.634949
Candidate was 18 with 0.071779
Discrimination ratio is 8.8

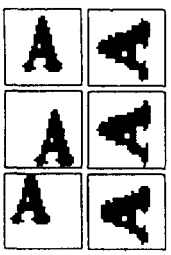


Pattern is 3 with 0.865484
Candidate was 12 with 0.046633
Discrimination ratio is 18.6

Pattern is 3 with 0.815480
Candidate was 12 with 0.074951
Discrimination ratio is 10.9

Pattern is 3 with 0.709035
Candidate was 12 with 0.067718
Discrimination ratio is 10.5

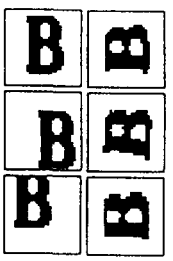
Fig. 17. Classification results with PREP2 on scaled letters.



Pattern is 1 with 0.911179
Candidate was 22 with 0.046750
Discrimination ratio is 19.5

Pattern is 1 with 0.807283
Candidate was 25 with 0.077792
Discrimination ratio is 10.4

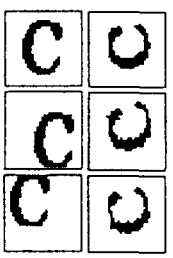
Pattern is 1 with 0.781698
Candidate was 9 with 0.074611
Discrimination ratio is 10.5



Pattern is 2 with 0.779841
Candidate was 19 with 0.072447
Discrimination ratio is 10.8

Pattern is 2 with 0.426662
Candidate was 18 with 0.191974
Discrimination ratio is 2.2

Pattern is 2 with 0.496783
Candidate was 19 with 0.050108
Discrimination ratio is 9.9

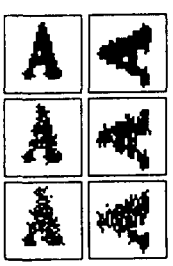


Pattern is 3 with 0.865484
Candidate was 12 with 0.046633
Discrimination ratio is 18.6

Pattern is 3 with 0.712326
Candidate was 12 with 0.211969
Discrimination ratio is 3.4

Pattern is 3 with 0.671936
Candidate was 12 with 0.099324
Discrimination ratio is 6.8

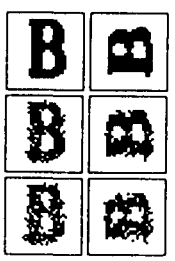
Fig. 18. Classification results with PREP2 on translated letters.



Pattern is 1 with 0.911179
Candidate was 22 with 0.046750
Discrimination ratio is 19.5

Pattern is 1 with 0.721102
Candidate was 25 with 0.119924
Discrimination ratio is 6.0

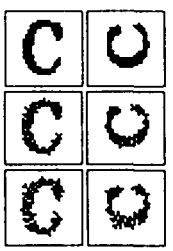
Pattern is 1 with 0.775800
Candidate was 25 with 0.131237
Discrimination ratio is 5.9



Pattern is 2 with 0.779841
Candidate was 19 with 0.072447
Discrimination ratio is 10.8

Pattern is 2 with 0.689617
Candidate was 18 with 0.058586
Discrimination ratio is 11.8

Pattern is 2 with 0.105599
Candidate was 4 with 0.052336
Discrimination ratio is 2.0

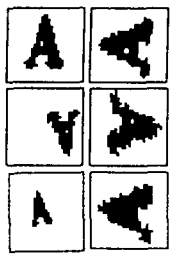


Pattern is 3 with 0.865484
Candidate was 12 with 0.046633
Discrimination ratio is 18.6

Pattern is 3 with 0.753888
Candidate was 7 with 0.223764
Discrimination ratio is 3.4

Pattern is 3 with 0.788500
Candidate was 7 with 0.127391
Discrimination ratio is 6.2

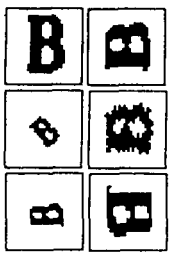
Fig. 19. Classification results with PREP2 on noisy letters.



Pattern is 1 with 0.911179
Candidate was 22 with 0.046750
Discrimination ratio is 19.5

Pattern is 1 with 0.353124
Candidate was 25 with 0.109775
Discrimination ratio is 3.2

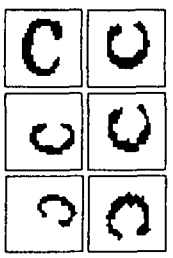
Pattern is 1 with 0.750400
Candidate was 25 with 0.078528
Discrimination ratio is 9.6



Pattern is 2 with 0.779841
Candidate was 19 with 0.072447
Discrimination ratio is 10.8

Pattern is 2 with 0.304866
Candidate was 4 with 0.141030
Discrimination ratio is 2.2

Pattern is 2 with 0.285387
Candidate was 18 with 0.099175
Discrimination ratio is 2.9

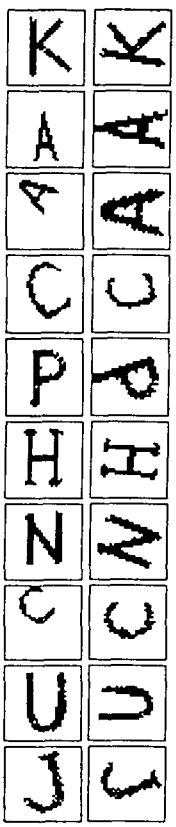


Pattern is 3 with 0.865484
Candidate was 12 with 0.046633
Discrimination ratio is 18.6

Pattern is 3 with 0.652949
Candidate was 16 with 0.076594
Discrimination ratio is 8.5

Pattern is 3 with 0.692806
Candidate was 12 with 0.083135
Discrimination ratio is 8.3

Fig. 20. Classification results with PREP2 on letters with random translation, scaling, and rotation applied.



Pattern is K with 0.739685
Candidate was R with 0.026135
Discrimination ratio is 28.3

Pattern is K with 0.138077
Candidate was R with 0.045842
Discrimination ratio is 3.0

Could not discriminate ...
Candidates were R with 0.179557
and V with 0.093892

Pattern is C with 0.395336
Candidate was J with 0.067349
Discrimination ratio is 5.9

Pattern is P with 0.619575
Candidate was V with 0.035673
Discrimination ratio is 17.4

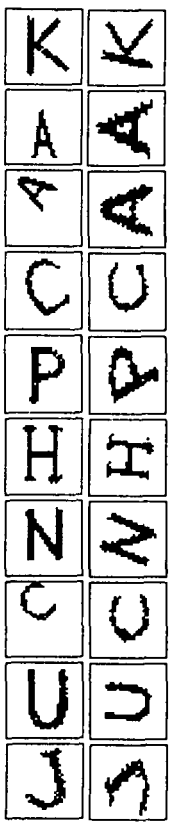
Pattern is H with 0.504894
Candidate was K with 0.091623
Discrimination ratio is 5.5

Pattern is T with 0.242300
Candidate was M with 0.058771
Discrimination ratio is 4.1

Pattern is C with 0.446178
Candidate was G with 0.058094
Discrimination ratio is 7.7

Pattern is O with 0.333856
Candidate was N with 0.055340
Discrimination ratio is 6.0

Pattern is L with 0.370625
Candidate was V with 0.065969
Discrimination ratio is 5.6



Pattern is 11.K with 0.200723
Candidate was 18.R with 0.088717
Discrimination ratio is 2.3

Pattern is 1.A with 0.328702
Candidate was 18.R with 0.134324
Discrimination ratio is 2.4

Could not discriminate ...
Candidates were 1.A with 0.156431
and 18.R with 0.11446

Pattern is 3.C with 0.239498
Candidate was 12.L with 0.062020
Discrimination ratio is 3.9

Pattern is 16.P with 0.253201
Candidate was 22.V with 0.038780
Discrimination ratio is 6.5

Pattern is 21.U with 0.388433
Candidate was 18.R with 0.064422
Discrimination ratio is 6.0

Pattern is 22.V with 0.367734
Candidate was 21.U with 0.111176
Discrimination ratio is 3.3

Pattern is 3.C with 0.754746
Candidate was 12.L with 0.044923
Discrimination ratio is 16.8

Pattern is 21.U with 0.179703
Candidate was 17.Q with 0.060315
Discrimination ratio is 3.0

Could not discriminate ...
Candidates were 17.Q with 0.061081
and 7.G with 0.036380

Fig. 21. Classification results, with PREP1 (left) and PREP2 (right), on ten hand-crafted English letters.

layer feed-forward network with 1024 input nodes, each one corresponding to one of the pixels of the $32 \times 32$ input image and 26 output neurons, one for each letter. An output value close to 1 is interpreted as a strong membership, while a value close to 0 will point to a loose membership. The number of hidden layers and the number of neurons in each hidden layer has been found by trial-and-error, which is typical for most multilayer feed-forward network applications.[3–8] After some experimentation we have found that a network with 1024 input nodes, 20 neurons in the single hidden layer, and 26 neurons in the output layer performs the best for a number of test cases.

In the training phase, the network is trained on the

| ア | カ | ガ | サ | ザ | タ | ダ | ナ | ハ | バ | パ | マ | ラ | ワ | ファ | ン |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | ka | ga | sa | za | ta | da | na | ha | ba | pa | ma | ra | wa | fa | n |
| イ | キ | ギ | シ | ジ | チ | ヂ | ニ | ヒ | ビ | ピ | ミ | リ | | フィ | |
| i | ki | gi | shi | ji | chi | ji | ni | hi | bi | pi | mi | ri | | fi | |
| ウ | ク | グ | ス | ズ | ツ | ヅ | ヌ | フ | ブ | プ | ム | ル | | | |
| u | ku | gu | su | zu | tsu | zu | nu | fu | bu | pu | mu | ru | | | |
| エ | ケ | ゲ | セ | ゼ | テ | デ | ネ | ヘ | ベ | ペ | メ | レ | | フェ | |
| e | ke | ge | se | ze | te | de | ne | he | be | pe | me | re | | fe | |
| オ | コ | ゴ | ソ | ゾ | ト | ド | ノ | ホ | ボ | ポ | モ | ロ | | フォ | ヲ |
| o | ko | go | so | zo | to | do | no | ho | bo | po | mo | ro | | fo | o |
| ヤ | キャ | ギャ | シャ | ジャ | チャ | ヂャ | ニャ | ヒャ | ビャ | ピャ | ミャ | リャ | | | |
| ya | kya | gya | sha | ja | cha | ja | nya | hya | bya | pya | mya | rya | | | |
| ユ | キュ | ギュ | シュ | ジュ | チュ | ヂュ | ニュ | ヒュ | ビュ | ピュ | ミュ | リュ | | | |
| yu | kyu | gyu | shu | ju | chu | ju | nyu | hyu | byu | pyu | myu | ryu | | | |
| ヨ | キョ | ギョ | ショ | ジョ | チョ | ヂョ | ニョ | ヒョ | ビョ | ピョ | ミョ | リョ | | | |
| yo | kyo | gyo | sho | jo | cho | jo | nyo | hyo | byo | pyo | myo | ryo | | | |

Fig. 22. The 111 symbols of the Japanese Katakana alphabet.

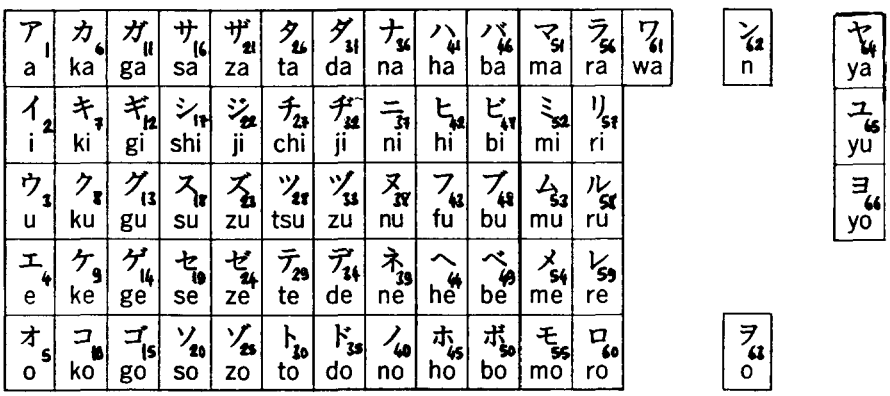



Fig. 23. The 66 unique patterns and their corresponding class numbers.

canonical example patterns (which are the outputs of the preprocessor) until it manages to successfully classify the letters. The example patterns for the English letters and their corresponding classes are given in Fig. 10.

Figures 11–15 present examples of the system performance using the preprocessor PREP1. Input images are 32 × 32 pixels. The first column is the original image given to the system. The second column is the preprocessed version of the original image, and finally the third column is the resulting decision. The class name and value of the corresponding output neuron are given. Figures 16–20 give the performance using the preprocessor PREP2. Figure 21 compares the classification results for two versions of the system, one with PREP1 and the other with PREP2. The input patterns are ten hand-crafted English letters.

Table 1 shows the (average) percentage of letters correctly classified after undergoing 100 random transformations of the type stated in the first column.†

### 4.2. *Character recognition on the Japanese Katakana alphabet*

This problem is the classification of symbols in the Japanese Katakana alphabet shown in Fig. 22. Since the 111 Katakana symbols are in fact combined forms of 66 unique patterns, the system is trained only on these patterns. Figure 23 shows the example patterns

† Combined denotes an input that is distorted from the original by a random scaling, rotation and translation. The noise is applied to an undeformed pattern by flipping the on-pixels with a certain probability. In the last row, noise is applied to the distorted pattern.

Table 1. Percentage of correct classification for letters under various distortions

| Transformation | PREP1 (%) | PREP2 (%) |
|---|---|---|
| Rotation | 91 | 89 |
| Scaling | 98 | 94 |
| Translation | 100 | 100 |
| Combined | 89 | 79 |
| 20% noise | 98 | 96 |
| 40% noise | 92 | 84 |
| Combined and 20% noise | 77 | 60 |

Table 2. Percentage of correct classification for Japanese Katakana symbols under various distortions

| Transformation | PREP1 (%) |
|---|---|
| Rotation | 75 |
| Scaling | 92 |
| Translation | 100 |
| Combined | 68 |
| 20% noise | 93 |
| 40% noise | 76 |
| Combined and 20% noise | 57 |

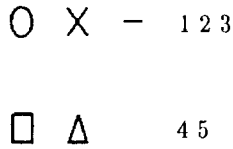



Fig. 24. The patterns and class numbers for the five main geometric symbols: circular, cross, line, square-like, and triangular.

and the corresponding class numbers. The network for the Japanese Katakana alphabet is similar to the network for the English alphabet except for the number of output neurons. With some experimentation, a network with only one hidden layer having 20 nodes has been chosen.

Figures 26–30 give the performance of the system with preprocessor PREP1. Table 2 shows the (average) percentage of Katakana symbols correctly classified after undergoing 100 random transformations of the type stated in the first column.

### 4.3. *Classification of geometric symbols*

This is the problem of classification of five main geometric symbols: circle, cross, line, rectangle, and triangle. The original patterns from each class is given

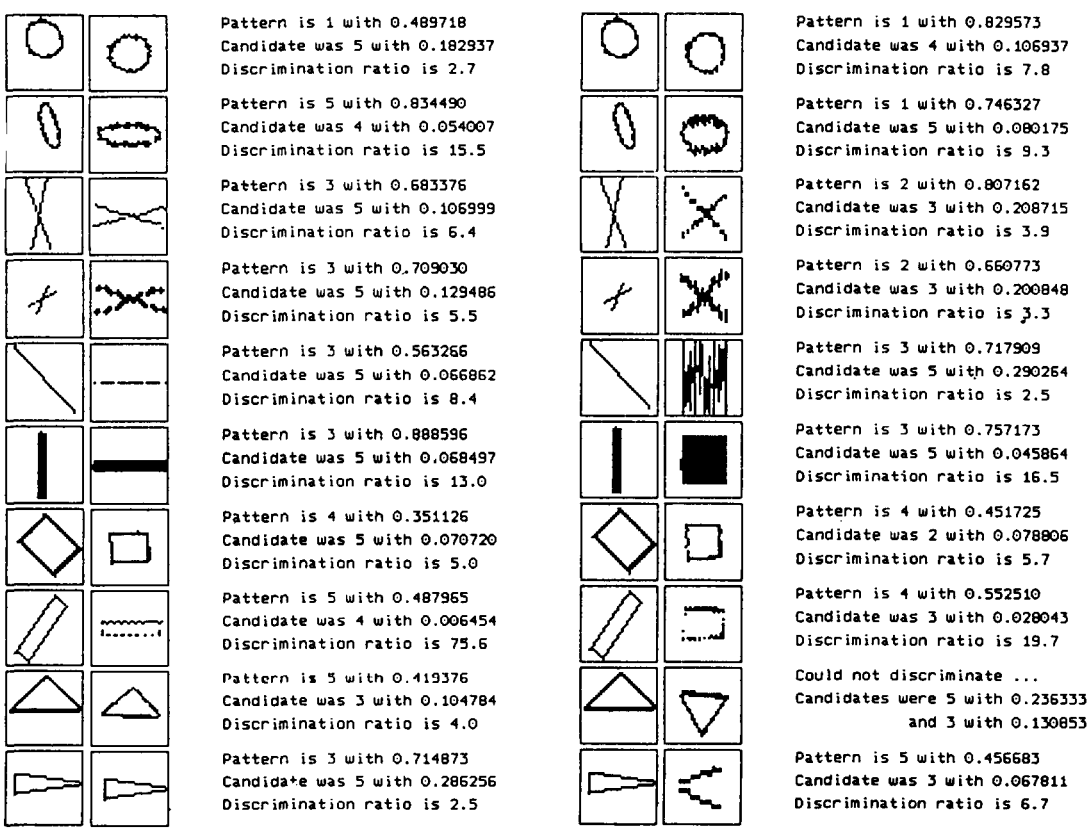
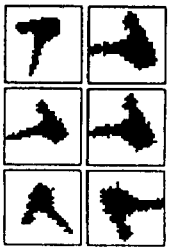


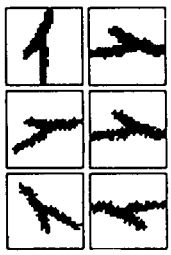

Fig. 25. Classification results, with PREP1 (left) and PREP2 (right), on distorted patterns of the five main geometric symbols.



Pattern is 1 with 0.820121
Candidate was 14 with 0.075892
Discrimination ratio is 10.8

Pattern is 1 with 0.554672
Candidate was 34 with 0.054674
Discrimination ratio is 10.1

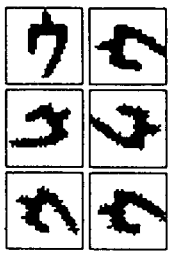
Pattern is 1 with 0.580281
Candidate was 5 with 0.071503
Discrimination ratio is 8.1



Pattern is 2 with 0.902589
Candidate was 13 with 0.048572
Discrimination ratio is 18.6

Pattern is 2 with 0.565567
Candidate was 33 with 0.072772
Discrimination ratio is 7.8

Pattern is 2 with 0.469407
Candidate was 40 with 0.017480
Discrimination ratio is 26.9

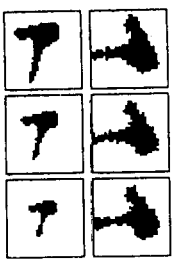


Pattern is 3 with 0.900058
Candidate was 6 with 0.056437
Discrimination ratio is 15.9

Pattern is 3 with 0.771208
Candidate was 6 with 0.072263
Discrimination ratio is 10.7

Pattern is 3 with 0.753924
Candidate was 6 with 0.087926
Discrimination ratio is 8.6

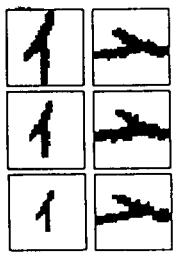
Fig. 26. Classification results with PREP1 for symbols from Class 1, 2, and 3 rotated by 0, 60, and $-60$ deg.



Pattern is 1 with 0.820121
Candidate was 14 with 0.075892
Discrimination ratio is 10.8

Pattern is 1 with 0.486271
Candidate was 34 with 0.042003
Discrimination ratio is 11.6

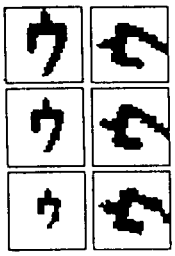
Pattern is 1 with 0.525178
Candidate was 34 with 0.077539
Discrimination ratio is 6.8



Pattern is 2 with 0.902589
Candidate was 13 with 0.048572
Discrimination ratio is 18.6

Pattern is 2 with 0.446612
Candidate was 40 with 0.203349
Discrimination ratio is 2.2

Pattern is 2 with 0.812504
Candidate was 33 with 0.035828
Discrimination ratio is 22.7

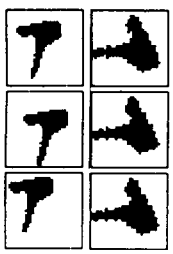


Pattern is 3 with 0.900058
Candidate was 6 with 0.056437
Discrimination ratio is 15.9

Pattern is 3 with 0.798072
Candidate was 66 with 0.085245
Discrimination ratio is 9.4

Pattern is 3 with 0.642452
Candidate was 51 with 0.253839
Discrimination ratio is 2.5

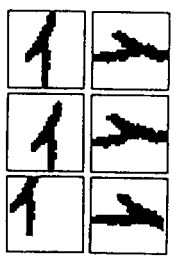
Fig. 27. Classification results with PREP1 for symbols from Class 1, 2, and 3 scaled by a factor of 1, 0.8, and 0.6.



Pattern is 1 with 0.820121
Candidate was 14 with 0.075892
Discrimination ratio is 10.8

Pattern is 1 with 0.801266
Candidate was 14 with 0.066163
Discrimination ratio is 12.1

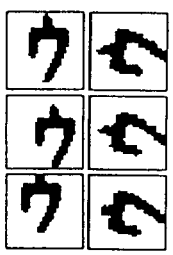
Pattern is 1 with 0.813754
Candidate was 14 with 0.074115
Discrimination ratio is 11.0



Pattern is 2 with 0.902589
Candidate was 13 with 0.048572
Discrimination ratio is 18.6

Pattern is 2 with 0.455013
Candidate was 35 with 0.069810
Discrimination ratio is 6.5

Pattern is 40 with 0.249899
Candidate was 9 with 0.099151
Discrimination ratio is 2.5

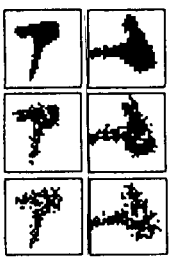


Pattern is 3 with 0.900058
Candidate was 6 with 0.056437
Discrimination ratio is 15.9

Pattern is 3 with 0.696134
Candidate was 51 with 0.196200
Discrimination ratio is 3.5

Pattern is 3 with 0.854651
Candidate was 66 with 0.055580
Discrimination ratio is 15.4

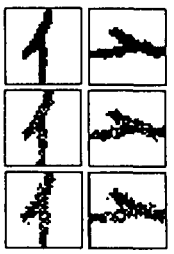
Fig. 28. Classification results with PREP1 for symbols from Class 1, 2, and 3 translated diagonally by 0, 6, and $-6$ pixels.



Pattern is 1 with 0.820121
Candidate was 14 with 0.075892
Discrimination ratio is 10.8

Pattern is 1 with 0.786378
Candidate was 14 with 0.054121
Discrimination ratio is 14.5

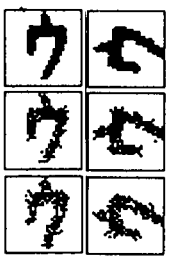
Pattern is 1 with 0.679368
Candidate was 21 with 0.038395
Discrimination ratio is 17.7



Pattern is 2 with 0.902589
Candidate was 13 with 0.048572
Discrimination ratio is 18.6

Pattern is 2 with 0.714926
Candidate was 40 with 0.069100
Discrimination ratio is 10.3

Pattern is 2 with 0.265836
Candidate was 6 with 0.080215
Discrimination ratio is 3.3

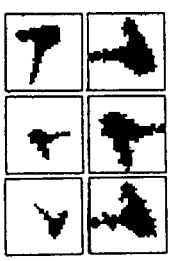


Pattern is 3 with 0.900058
Candidate was 6 with 0.056437
Discrimination ratio is 15.9

Pattern is 3 with 0.767802
Candidate was 53 with 0.215734
Discrimination ratio is 3.6

Could not discriminate ...
Candidates were 3 with 0.334035
and 22 with 0.293054

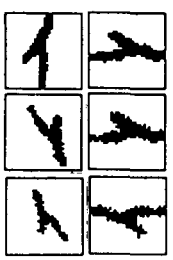
Fig. 29. Classification results with PREP1 for symbols from Class 1, 2, and 3 with 0, 20, and 40% noise.



Pattern is 1 with 0.820121
Candidate was 14 with 0.075892
Discrimination ratio is 10.8

Pattern is 1 with 0.640245
Candidate was 34 with 0.052577
Discrimination ratio is 12.2

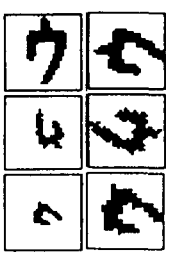
Pattern is 1 with 0.463797
Candidate was 13 with 0.140803
Discrimination ratio is 3.3



Pattern is 2 with 0.902589
Candidate was 13 with 0.048572
Discrimination ratio is 18.6

Pattern is 2 with 0.572148
Candidate was 13 with 0.103430
Discrimination ratio is 5.5

Pattern is 2 with 0.758859
Candidate was 40 with 0.109365
Discrimination ratio is 6.9



Pattern is 3 with 0.900058
Candidate was 6 with 0.056437
Discrimination ratio is 15.9

Pattern is 3 with 0.386634
Candidate was 6 with 0.082848
Discrimination ratio is 4.7

Pattern is 3 with 0.537339
Candidate was 66 with 0.110447
Discrimination ratio is 4.9

Fig. 30. Classification results with PREP1 for symbols from Class 1, 2, and 3 with random translation, scaling, and rotation applied.

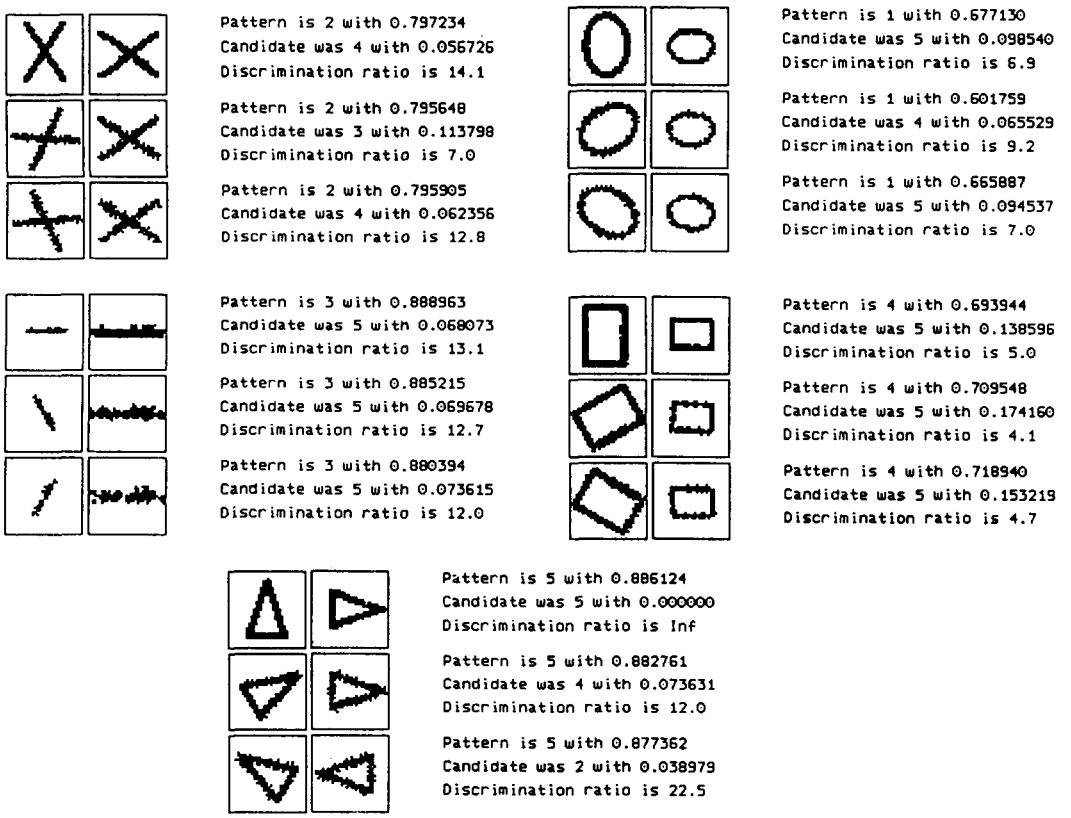



Fig. 31. Classification results with PREP1 for geometric symbols rotated by 0, 60, and −60 deg.

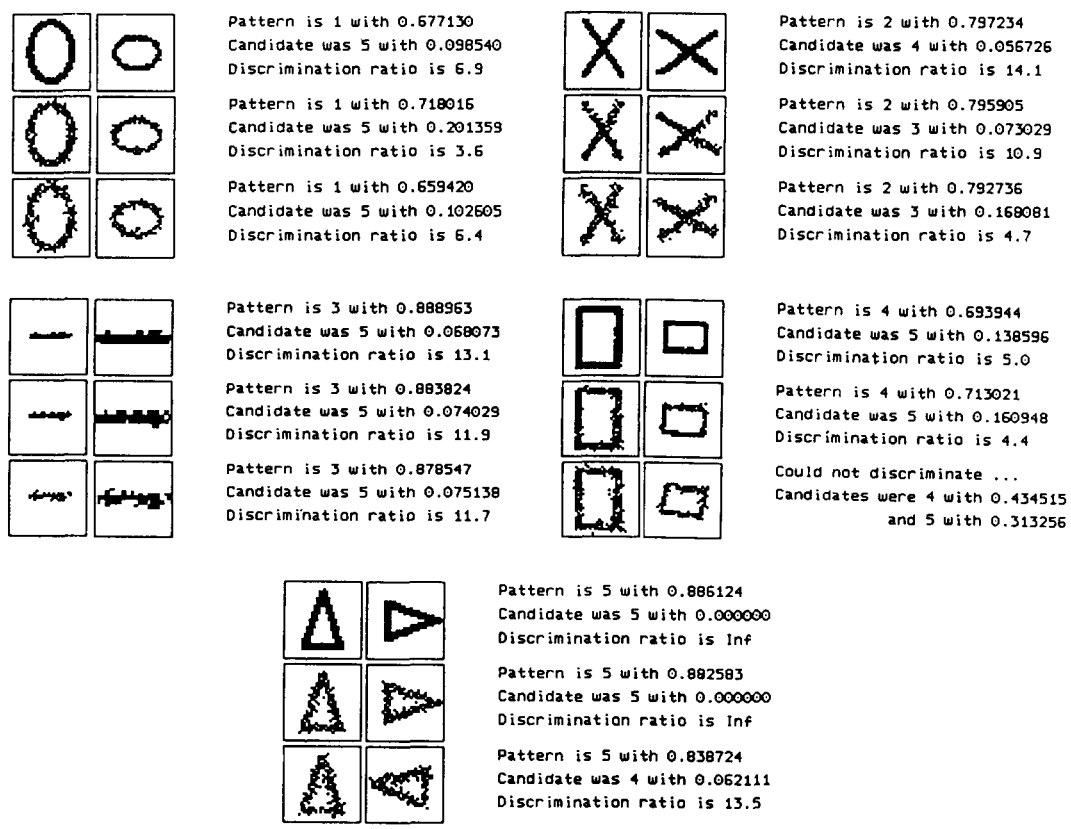



Fig. 32. Classification results with PREP1 for geometric symbols scaled by a factor of 1, 0.8, and 0.6.

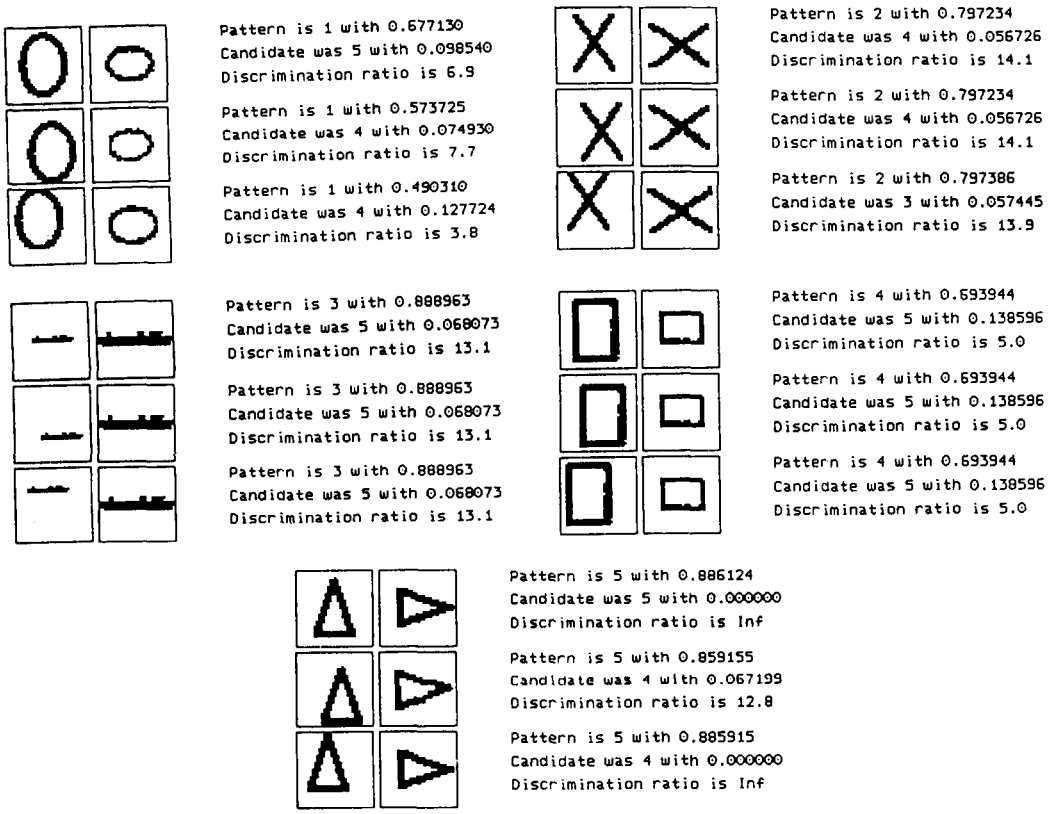



Fig. 33. Classification results with PREP1 for geometric symbols translated diagonally by 0, 6, and $-6$ pixels.

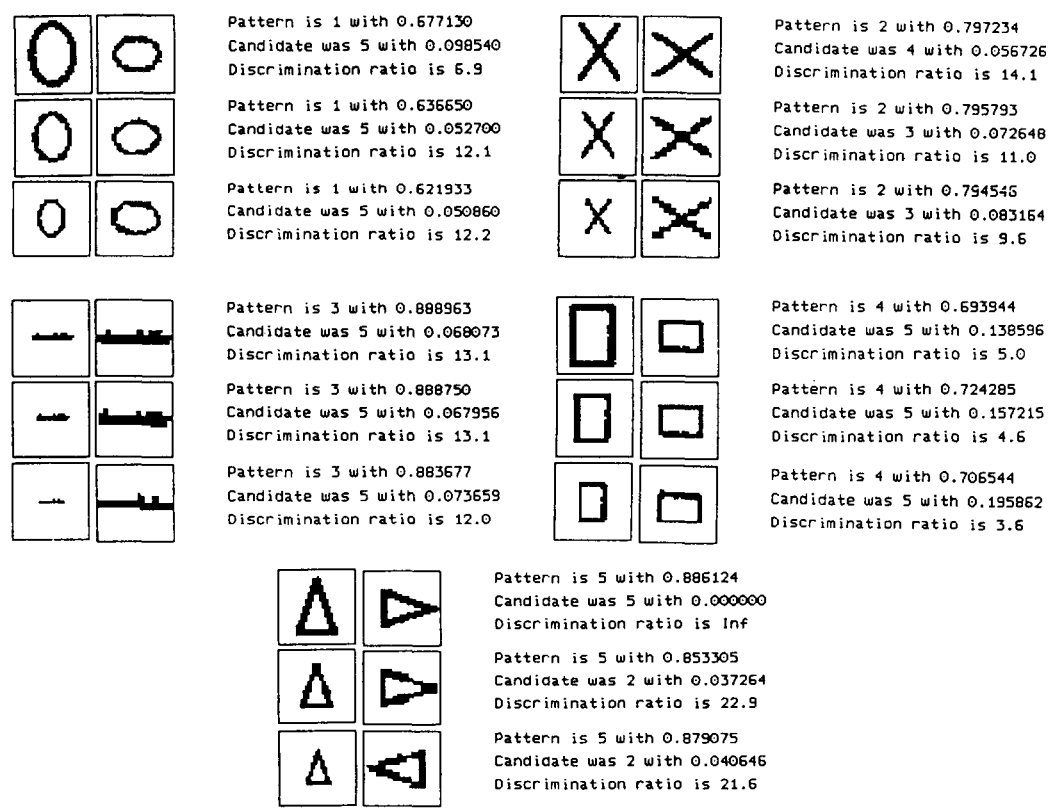
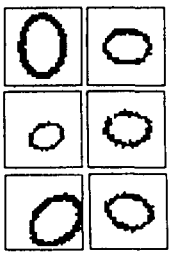


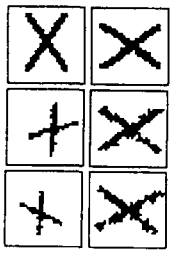

Fig. 34. Classification results with PREP1 for geometric symbols with 0, 20, and 40% noise.



Pattern is 1 with 0.677130
Candidate was 5 with 0.098540
Discrimination ratio is 6.9

Pattern is 1 with 0.627392
Candidate was 5 with 0.059374
Discrimination ratio is 10.6

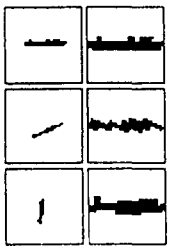
Pattern is 1 with 0.696513
Candidate was 5 with 0.204566
Discrimination ratio is 3.4



Pattern is 2 with 0.797234
Candidate was 4 with 0.056726
Discrimination ratio is 14.1

Pattern is 2 with 0.794498
Candidate was 3 with 0.060785
Discrimination ratio is 13.1

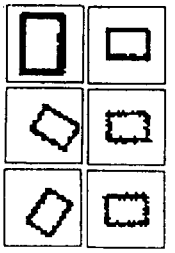
Pattern is 2 with 0.780405
Candidate was 3 with 0.220720
Discrimination ratio is 3.5



Pattern is 3 with 0.888963
Candidate was 5 with 0.068073
Discrimination ratio is 13.1

Pattern is 3 with 0.885966
Candidate was 5 with 0.071898
Discrimination ratio is 12.3

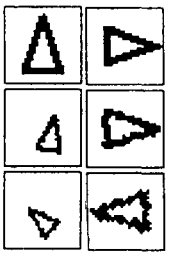
Pattern is 3 with 0.883736
Candidate was 5 with 0.075221
Discrimination ratio is 11.7



Pattern is 4 with 0.693944
Candidate was 5 with 0.138596
Discrimination ratio is 5.0

Pattern is 4 with 0.606711
Candidate was 5 with 0.133236
Discrimination ratio is 4.6

Pattern is 4 with 0.719280
Candidate was 5 with 0.159010
Discrimination ratio is 4.5

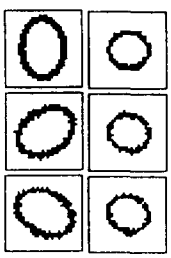


Pattern is 5 with 0.886124
Candidate was 5 with 0.000000
Discrimination ratio is Inf

Pattern is 5 with 0.859256
Candidate was 4 with 0.060094
Discrimination ratio is 14.3

Pattern is 5 with 0.870000
Candidate was 4 with 0.069743
Discrimination ratio is 12.5

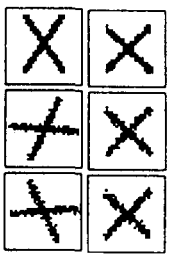
Fig. 35. Classification results with PREP1 for geometric symbols with random translation, scaling, and rotation applied.



Pattern is 1 with 0.879950
Candidate was 5 with 0.053595
Discrimination ratio is 16.4

Pattern is 1 with 0.877091
Candidate was 4 with 0.055626
Discrimination ratio is 15.8

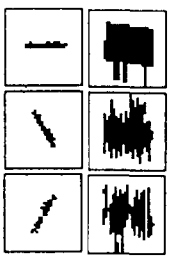
Pattern is 1 with 0.863999
Candidate was 4 with 0.061732
Discrimination ratio is 14.0



Pattern is 2 with 0.876442
Candidate was 5 with 0.136934
Discrimination ratio is 6.4

Pattern is 2 with 0.833144
Candidate was 4 with 0.124221
Discrimination ratio is 6.7

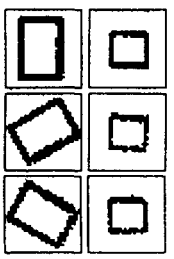
Pattern is 2 with 0.862304
Candidate was 3 with 0.158863
Discrimination ratio is 5.4



Pattern is 3 with 0.871819
Candidate was 5 with 0.031803
Discrimination ratio is 27.4

Pattern is 3 with 0.800754
Candidate was 5 with 0.034429
Discrimination ratio is 23.3

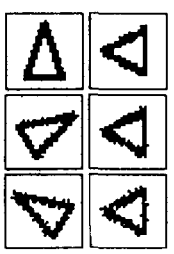
Pattern is 3 with 0.755228
Candidate was 5 with 0.334764
Discrimination ratio is 2.3



Pattern is 4 with 0.643865
Candidate was 3 with 0.025258
Discrimination ratio is 25.5

Pattern is 4 with 0.621688
Candidate was 3 with 0.028381
Discrimination ratio is 21.9

Pattern is 4 with 0.647699
Candidate was 3 with 0.024492
Discrimination ratio is 26.4



Pattern is 5 with 0.589879
Candidate was 3 with 0.041384
Discrimination ratio is 14.3

Pattern is 5 with 0.580918
Candidate was 3 with 0.046978
Discrimination ratio is 12.4

Pattern is 5 with 0.549789
Candidate was 3 with 0.053978
Discrimination ratio is 10.2

Fig. 36. Classification results with PREP2 for rotated geometric symbols.

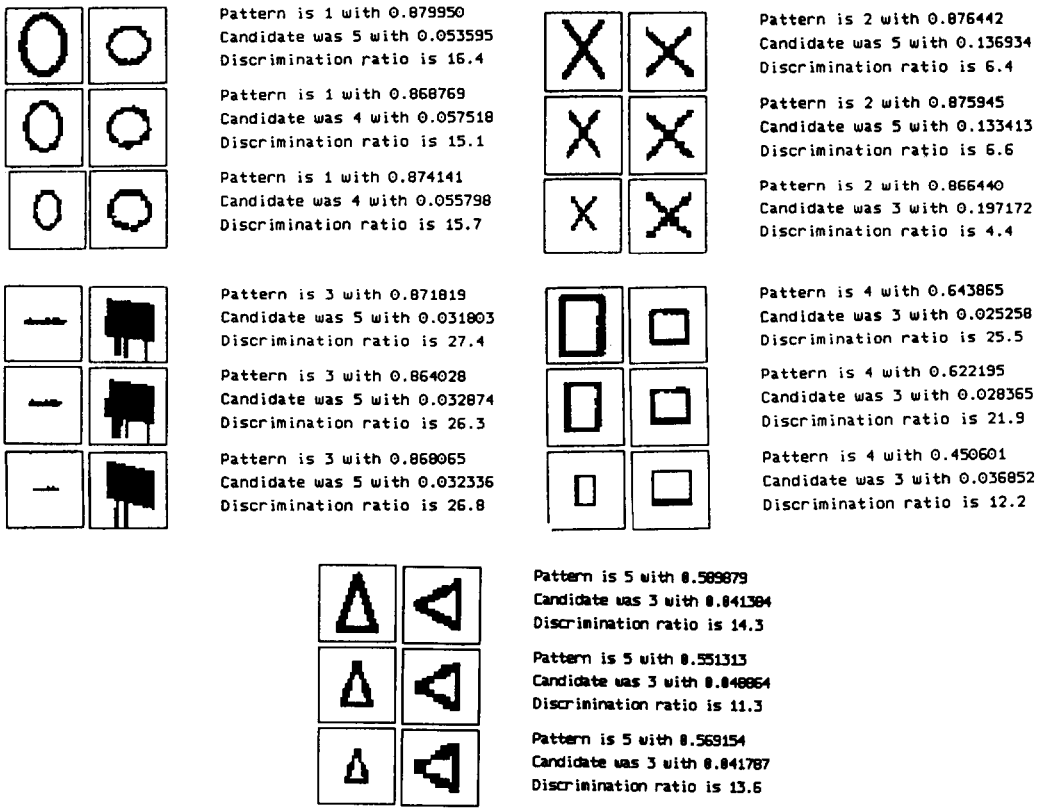



Fig. 37. Classification results with PREP2 for scaled geometric symbols.

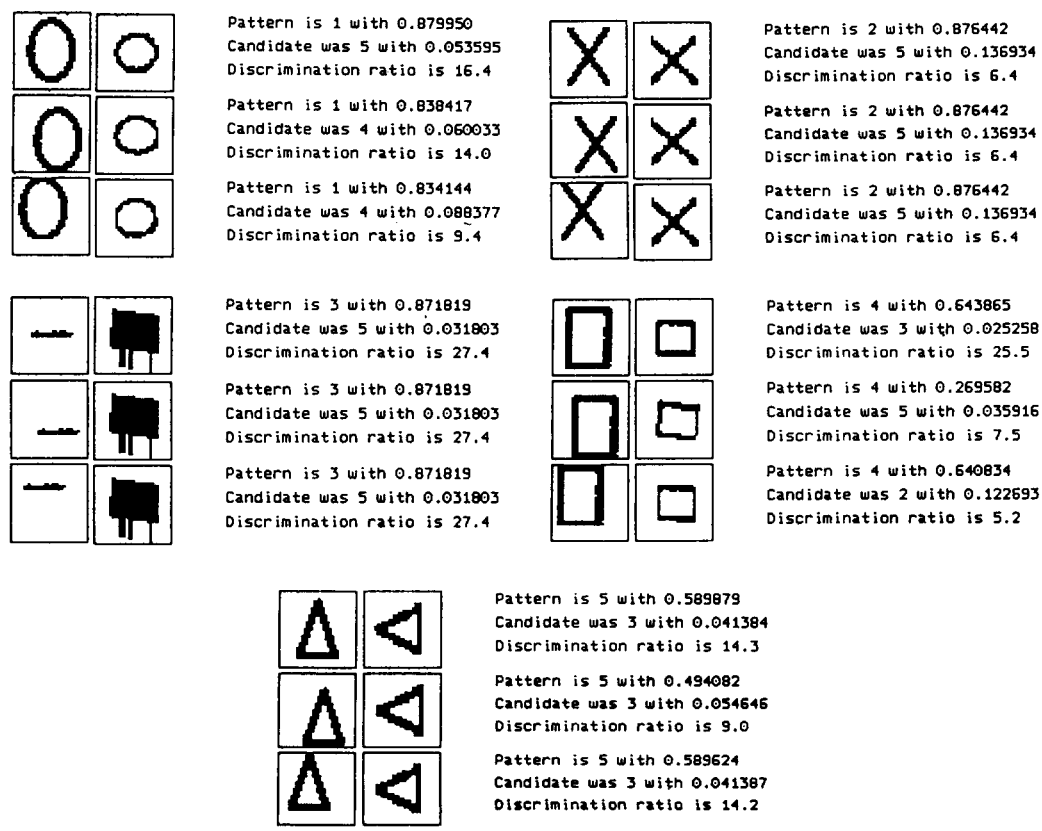



Fig. 38. Classification results with PREP2 for translated geometric symbols.

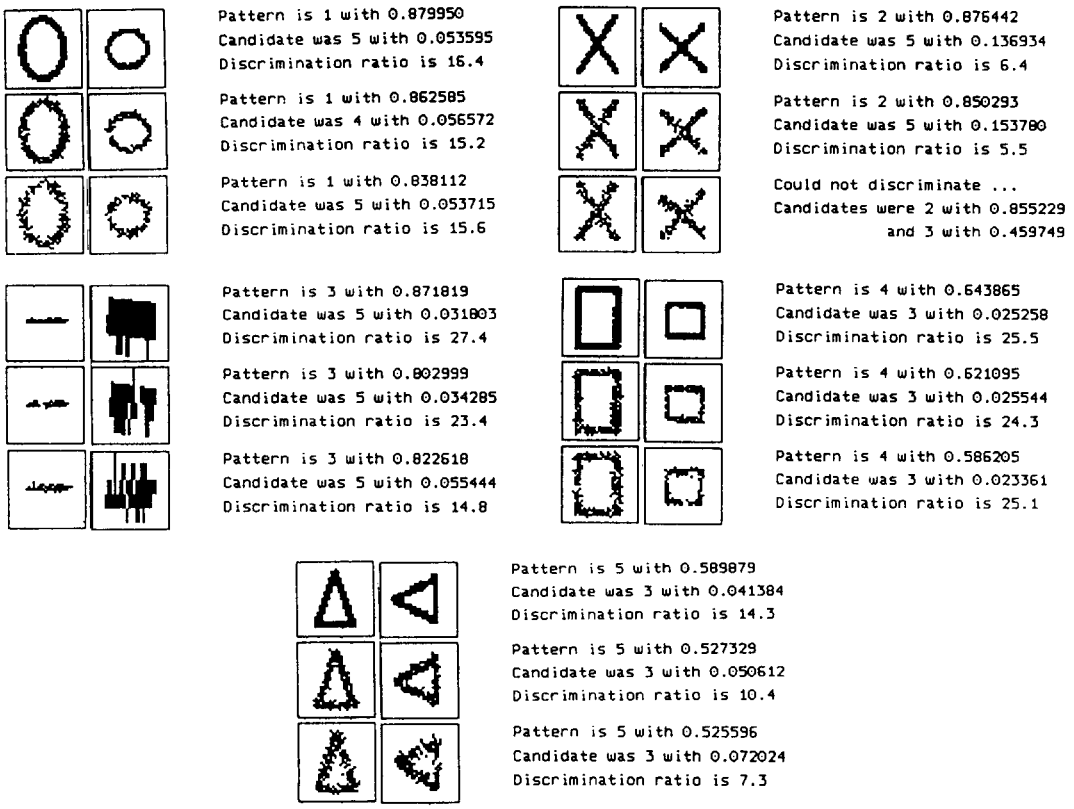



Fig. 39. Classification results with PREP2 for noisy geometric symbols.

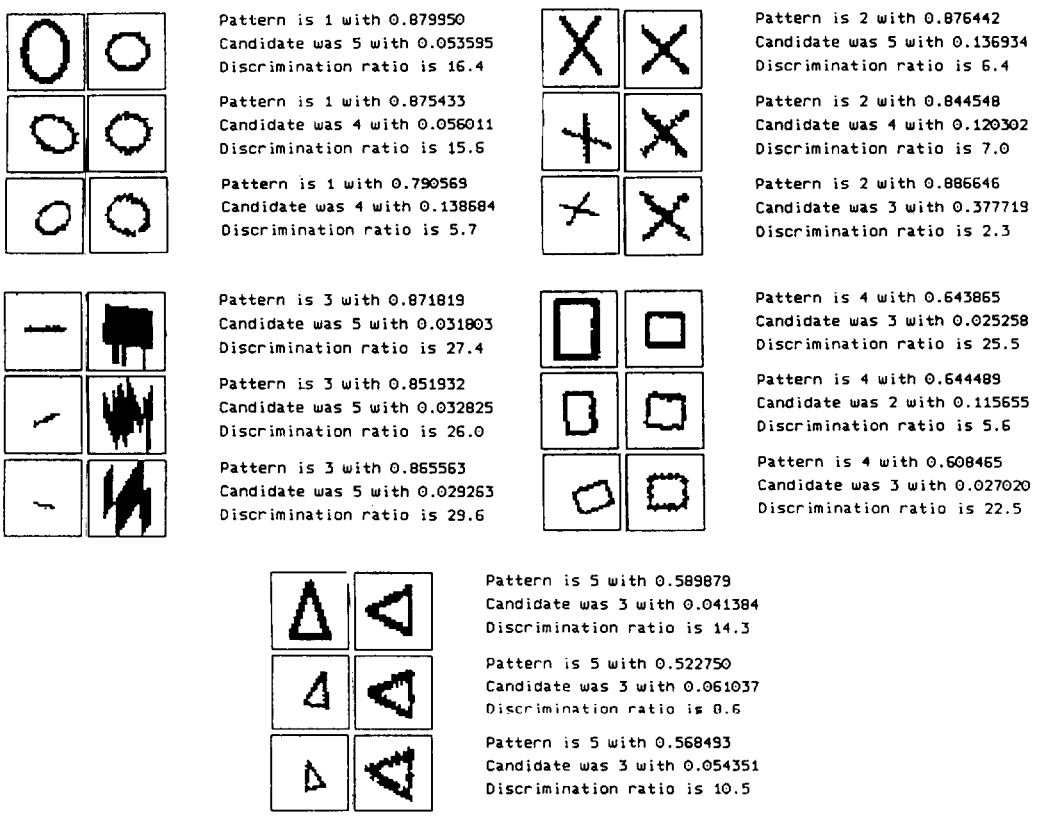



Fig. 40. Classification results with PREP2 geometric symbols with random translation, scaling and rotation applied.

Table 3. Percentage of correct classification for geometric symbols under various distortions.

| Transformation | PREP1 (%) | PREP2 (%) |
|---|---|---|
| Rotation | 98 | 94 |
| Scaling | 100 | 100 |
| Translation | 100 | 100 |
| Combined | 88 | 84 |
| 20% noise | 100 | 100 |
| 40% noise | 98 | 97 |
| Combined and 20% noise | 89 | 78 |

Table 4. Results for the three problems with PREP1

| Transformation | Letters (%) | Katakana (%) | Symbols (%) |
|---|---|---|---|
| Rotation | 91 | 75 | 98 |
| Scaling | 98 | 92 | 100 |
| Translation | 100 | 100 | 100 |
| Combined | 89 | 68 | 88 |
| 20% noise | 98 | 93 | 100 |
| 40% noise | 92 | 76 | 98 |
| Combined and 20% noise | 77 | 57 | 89 |

Table 5. Results for the letter and geometric symbol problems with PREP2

| Transformation | Letters (%) | Symbols (%) |
|---|---|---|
| Rotation | 89 | 94 |
| Scaling | 94 | 100 |
| Translation | 100 | 100 |
| Combined | 79 | 84 |
| 20% noise | 96 | 100 |
| 40% noise | 84 | 97 |
| Combined and 20% noise | 60 | 78 |

in Fig. 24. Figures 31–35 give the performance on single grid images of the geometric symbols, using PREP1. Figures 36–40 give the performance using PREP2.

Figure 25 gives the classification results for the two versions of the preprocessor on the distorted versions of the geometric symbols. PREP1 could detect only 50% of the distorted patterns, while PREP2 managed to successfully classify 90%. The performance difference emerges from the axial scaling correction ability of this preprocessor.

Table 3 shows the (average) percentage of geometric symbols correctly classified after undergoing 100 random transformations of the type stated in the first column.

Tables 4 and 5 present results for the two preprocessors respectively, for the three problems above.

## 5. DISCUSSION AND CONCLUSIONS

In this work we have presented a hybrid pattern classification system which can classify patterns independent of any deformations of translation, scaling and rotation. The system uses a preprocessor which maps input patterns to a set of canonical patterns which is then classified by a multilayer neural network. The artificial neural network is trained using the popular backpropagation algorithm. The use of the preprocessor reduces the number of training patterns to two per example pattern to be classified instead of a much larger number if the networks were to be trained on all possible distortions of the patterns. Results from three different applications were presented. In classification of letters of the English alphabet the system was able to correctly classify 89% of the inputs which were deformed by random rotations, translations and scaling. The performance is much better when distortions were only of one kind, with 100% of the inputs distorted by only translations correctly classified. In the recognition of geometric figures, the system was able to correctly classify 88% of the inputs which were deformed by all three kinds of distortions while the performance was almost perfect when the random distortions were only of one kind. The overall performance for recognition of the Japanese Katakana alphabet was worse compared to the other two applications. Of the inputs with combined deformations 68% were correctly classified. Even though the performance for inputs which are distorted only by translations or scaling is very good, the performance for rotationally distorted inputs was about 75%. The main reason for this loss of performance is that some of the letters in this case are very similar to each other differing by very small visual features. When patterns that are already deformed are processed with the preprocessors, a certain amount of superfluous visual features may be introduced during mapping between images. The system presented here is independent of the application domain and leaves features extraction to the neural network and thus can be applied in other domains with relative simplicity.

## REFERENCES

1. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. Wiley, New York (1973).
2. K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, New York (1972).
3. E. Barnard and D. Casasent, Shift invariance and the neocognitron, *Neural Networks* **3**, 403–410 (1990).
4. Y. Le Cun *et al.*, Handwritten digit recognition: applications of neural network chips and automatic learning, *IEEE Commun. Mag.* 41–46 (November 1989).
5. Y. Le Cun *et al.*, Handwritten zip code recognition with multilayer networks, *Proc. 10th Int. Conf. on Pattern Recognition*, Atlantic City, New Jersey, U.S.A., pp. 35–40. IEEE Computer Society Press, Los Alamos (1990).
6. K. Fukushima, S. Miyake and T. Ito, Neocognitron: a neural network model for a mechanism of visual pattern recognition, *IEEE Trans. Syst. Man Cybern.* **SMC-13**, 826–834 (1983).
7. A. Khotanzad and J. Lu, Classification of invariant image representations using a neural network, *IEEE Trans. Acoustics Speech Signal Process.* **38**(6), 1028–1038 (June 1990).
8. H. A. Malki and A. Moghaddamjoo, Using the Karhunen–Loéve transformations in the back-propagation training algorithm, *IEEE Trans. Neural Networks* **2**(1), 162–165 (January 1991).
9. A. P. Reeves, R. J. Prokop, S. E. Andrews and F. Kuhl, Three-dimensional shape analysis using moments and

Fourier descriptors, *IEEE Trans. Pattern Analysis Mach. Intell.* **10**(6), 937–943 (November 1988).
10. A. Kryzak, S. Y. Leung and C. Y. Suen, Reconstruction of two-dimensional patterns by Fourier descriptors, *Proc. 9th ICPR*, Rome, Italy, pp. 555–558 (November 1988).
11. E. Persoon and K. S. Fu, Shape discrimination using Fourier descriptors, *IEEE Trans. Syst. Man Cybern.* **SMC-7**, 170–179 (March 1977).
12. C. T. Zhan and C. T. Roskies, Fourier descriptors for plane closed curves, *IEEE Trans. Comput.* **C-21**, 269–281 (March 1972).
13. R. L. Kashyap and R. Chellappa, Stochastic models for closed boundary analysis: representation and reconstruction, *IEEE Trans. Inf. Theory* **IT-27**, 627–637 (September 1981).
14. Y. N. Hsu, H. H. Arsenault and G. April, Rotational invariant digital pattern recognition using circular harmonic expansion, *Appl. Optics* **21**, 4012–4015 (1982).
15. K. S. Fu, *Syntactic Pattern Recognition and Application*. Prentice-Hall, Englewood Cliffs, New Jersey (1982).
16. M. Kirby and L. Sirovich, Application of the Karhunen–Loéve procedures for the characterization of human faces, *IEEE Trans. Pattern Analysis Mach. Intell.* **12**(1), 103–108 (January 1990).
17. S. Casasent and D. Psaltis, Position, rotation, and scale invariant correlation, *Appl. Optics* **15**, 1795–1799 (1976).
18. S. Kollias, A. Tirakis and T. Millios, An efficient approach to invariant recognition of images using higher-order neural networks, *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula and J. Kangas, eds, pp. 87–92. Elsevier Science, North-Holland, Amsterdam (1991).
19. G. L. Martin and J. A. Pittman, Recognition hand-printed letters and digits using backpropagation learning, *Neural Computation* No. 3, 258–267 (1991).
20. D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation, *Parallel Distributed Processing*, D. E. Rumelhart and J. L. McClelland, eds, Vol. 1, pp. 318–362. MIT Press, Bradford Books, Cambridge, Massachusetts (1986).
21. R. C. Gonzales and P. Wintz, *Digital Image Processing*, pp. 122–130. Addison-Wesley, Reading, Massachusetts (1987).
22. K. Oflazer and D. Ercoşkun, Implementation of back-propagation learning algorithm on a hypercube parallel processor system, *Proc. ISCIS V, The Fifth Int. Symp. on Computer and Information Science*, Türkiye (1990).
23. C. Yüceer and K. Oflazer, A rotation, scaling, and translation invariant pattern classification system, *Proc. ISCIS VI, The Sixth Int. Symp. on Computer and Information Science*, Side, Türkiye, Mehmet Baray and Bülent Özgüç, eds, Vol. 2, pp. 859–869 (October 1991).

## APPENDIX A. DERIVATION OF THE R-BLOCK FUNCTIONS

Define

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix} \tag{A1}$$

$$P = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j) \tag{A2}$$

$$T_{xx} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)\cdot x_i^2, \quad T_{yy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)\cdot y_j^2 \tag{A3}$$

$$T_{xy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)\cdot x_i \cdot y_j. \tag{A4}$$

The covariance matrix defined as

$$C = \frac{1}{P}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right)\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right)^{T} \tag{A5}$$

can be simplified to

$$C = \left(\frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{TS}(x_i, y_j) \times \begin{bmatrix} x_i \\ y_j \end{bmatrix}\begin{bmatrix} x_i \\ y_j \end{bmatrix}^{T}\right) - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\begin{bmatrix} m_x \\ m_y \end{bmatrix}^{T}. \tag{A6}$$

For our application, the averages $m_x$ and $m_y$ are zero since we have provided translational invariancy. Furthermore, the averaging fraction in front of the matrix can be eliminated since it does not change the direction of the eigenvectors of the correlation matrix. Therefore, the correlation matrix can be written as

$$C = \begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix}. \tag{A7}$$

The eigenvalues of this matrix are computed from

$$(\lambda I - C) = \begin{bmatrix} (\lambda - T_{xx}) & -T_{xy} \\ -T_{xy} & (\lambda - T_{yy}) \end{bmatrix} \tag{A8}$$

$$(\lambda - T_{xx})(\lambda - T_{yy}) - T_{xy}^2 = 0 \tag{A9}$$

$$\lambda_{1,2} = \frac{T_{yy} + T_{xx} \pm \sqrt{((T_{yy} + T_{xx})^2 - 4\cdot(T_{xx}\cdot T_{yy} - T_{xy}^2))}}{2} \tag{A10}$$

then the corresponding eigenvectors can be derived as

$$\begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix} \tag{A11}$$

$$T_{xx}\cdot x + T_{xy}\cdot y = \lambda \cdot x$$

$$T_{xy}\cdot x + T_{yy}\cdot y = \lambda \cdot y \tag{A12}$$

$$\frac{y}{x} = \frac{\lambda - T_{xx}}{T_{xy}}. \tag{A13}$$

Substituting the greatest eigenvalue, one gets the slope of the eigenvector as

$$\frac{y}{x} = \frac{T_{yy} - T_{xx} + \sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)}}{2\cdot T_{xy}}. \tag{A14}$$

Hence, the sine and cosine for this slope can be derived from

$$\sin\theta = \frac{y}{\sqrt{(x^2 + y^2)}} \tag{A15}$$

$$= \frac{(T_{yy} - T_{xx}) + \sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)}}{\sqrt{(8\cdot T_{xy}^2 + 2\cdot(T_{yy} - T_{xx})^2 + 2\cdot(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)})}} \tag{A16}$$

$$\cos\theta = \frac{x}{\sqrt{(x^2 + y^2)}} \tag{A17}$$

$$= \frac{2\cdot T_{xy}}{\sqrt{(8\cdot T_{xy}^2 + 2\cdot(T_{yy} - T_{xx})^2 + 2\cdot(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)})}}. \tag{A18}$$

The mapping function of the R-Block is

$$f_{TSR}(x_i, y_j) = f_{TS}(\cos\theta \cdot x_i - \sin\theta \cdot y_j, \sin\theta \cdot x_i + \cos\theta \cdot y_j). \tag{A19}$$

## APPENDIX B. DERIVATION OF THE PREP2 FUNCTIONS

Define

$$P = \sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j) \tag{B1}$$

$$x_{av} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot x_i}{P} \tag{B2}$$

$$y_{av} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot y_j}{P}. \tag{B3}$$

Recalling from equation (9), the formula for $T_{xx}$ which was

$$T_{xx} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_T(x_i, y_j)\cdot x_i^2 \tag{B4}$$

can be written as

$$T_{xx} = \sum_{i=1}^{N}\sum_{k=1}^{N} f(x_i, y_j)(x_i - x_{av})^2 \tag{B5}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot x_i^2 - 2\cdot x_{av}\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot x_i + x_{av}^2\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j) \tag{B6}$$

$$= \left(\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot x_i^2\right) - P\cdot x_{av}^2. \tag{B7}$$

Similarly

$$T_{yy} = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot y_j^2\right) - P\cdot y_{av}^2 \tag{B8}$$

$$T_{xy} = \left(\sum_{i=1}^{N}\sum_{j=1}^{M} f(x_i, y_j)\cdot x_i\cdot y_j\right) - P\cdot x_{av}\cdot y_{av}. \tag{B9}$$

The sine and cosine of the rotation angle remain unchanged

$$\sin\theta = \frac{(T_{yy} - T_{xx}) + \sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)}}{\sqrt{(8\cdot T_{xy}^2 + 2\cdot(T_{yy} - T_{xx})^2 + 2\cdot(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)})}} \tag{B10}$$

$$\cos\theta = \frac{2\cdot T_{xy}}{\sqrt{(8\cdot T_{xy}^2 + 2\cdot(T_{yy} - T_{xx})^2 + 2\cdot(T_{yy} - T_{xx})\sqrt{((T_{yy} - T_{xx})^2 + 4\cdot T_{xy}^2)})}}. \tag{B11}$$

In order to maintain scaling correction on the translation and rotation invariant image, we define the scaling factors as

$$s_x = \sqrt{\left(\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{TR}(x_i, y_j)\cdot x_i^2}{R_x\cdot P}\right)} \tag{B12}$$

$$s_y = \sqrt{\left(\frac{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{TR}(x_i, y_j)\cdot y_j^2}{R_y\cdot P}\right)} \tag{B13}$$

The numerator term can be written as

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f_{TR}(x_i, y_j)\cdot x_i^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)[(x_i - x_{av})\cos\theta + (y_i + y_{av})\sin\theta]^2 \tag{B14}$$

which simplifies into

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)[(x_i^2 - x_{av}^2)\cos^2\theta + 2(x_i y_j - x_{av}y_{av})\cos\theta\sin\theta + (y_i^2 - y_{av}^2)\sin^2\theta] \tag{B15}$$

substituting $T_{xx}$, $T_{xy}$, and $T_{yy}$ one gets

$$s_x = \sqrt{\left(\frac{T_{xx}\cdot(\cos\theta)^2 + 2\cdot T_{xy}\cdot\cos\theta\cdot\sin\theta + T_{yy}\cdot(\sin\theta)^2}{R_x\cdot P}\right)} \tag{B16}$$

similarly the scaling factor $s_y$ can be derived as

$$s_y = \sqrt{\left(\frac{T_{xx}\cdot(\sin\theta)^2 + 2\cdot T_{xy}\cdot\cos\theta\cdot\sin\theta + T_{yy}\cdot(\cos\theta)^2}{R_y\cdot P}\right)}. \tag{B17}$$

**About the Author**—CEM YÜCEER is a Ph D. student at the Department of Computer Engineering and Information Science at Bilkent University, Ankara, Turkey. He obtained a B.S. degree in electrical engineering from Middle East Technical University, Ankara, Turkey and an M.S. degree in computer engineering and information science from Bilkent University. His research interests are in neural networks and pattern recognition.

**About the Author**—KEMAL OFLAZER is with the Department of Computer Engineering and Information Science at Bilkent University, Ankara, Turkey. He obtained his Ph.D. from the Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A. He obtained his M.S. degree in computer engineering and his B.S. degree in electrical engineering from Middle East Technical University, Ankara, Turkey. His current research interests are in artificial neural networks, connectionist modeling of higher level symbolic processes, computational linguistics and parallel processing for nonnumeric applications.