

A GENERAL PURPOSE  
ROTATION, SCALING, AND TRANSLATION  
INVARIANT  
PATTERN CLASSIFICATION SYSTEM

A THESIS SUBMITTED TO  
THE DEPARTMENT OF COMPUTER ENGINEERING AND  
INFORMATION SCIENCE  
AND THE INSTITUTE OF ENGINEERING  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE

by  
Cem Yüceer  
1992

QA  
76-87  
-Y83  
1992



# Corrections

Equation (3.4):

$$f_T(x_i, y_j) = f(x_i + x_{av}, y_j + y_{av})$$

Equation (3.6):

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

Equations (3.17) and (A.19):

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

Equations (3.26) and (B.16):

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

Equations (3.27) and (B.17):

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

Equation (3.28):

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

Equation (B.12):

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

Equation (B.13):

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

A GENERAL PURPOSE  
ROTATION, SCALING, AND TRANSLATION  
INVARIANT  
PATTERN CLASSIFICATION SYSTEM

A THESIS SUBMITTED TO  
THE DEPARTMENT OF COMPUTER ENGINEERING AND  
INFORMATION SCIENCE  
AND THE INSTITUTE OF ENGINEERING  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE

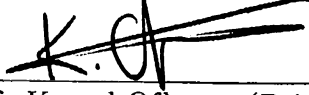
by  
Cem Yüceer  
1992

Cem Yüceer  
tarafından hazırlanmıştır.

3.9945

GA  
76.87  
.983  
1992

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.




Assoc. Prof. Kemal Oflazer (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



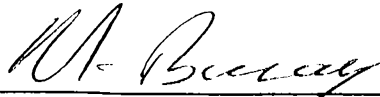
Prof. Mehmet Baray

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Uğur Halıcı

Approved by the Institute of Engineering:



Prof. Mehmet Baray, Director of Institute of Engineering

# ABSTRACT

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

Cem Yüceer

M.S. in Computer Engineering and Information Science

Supervisor: Assoc. Prof. Kemal Oflazer

1992

Artificial neural networks have recently been used for pattern classification purposes. In this work, a general purpose pattern classification system which is rotation, scaling, and, translation invariant is introduced. The system has three main blocks; a Karhunen-Loève transformation based preprocessor, an artificial neural network based classifier, and an interpreter. Through experimentation on the English alphabet, the Japanese Katakana alphabet, and some geometric symbols the power of the system in maintaining invariances and performing pattern classification has been shown.

Keywords: Rotational invariancy, scaling invariancy, translational invariancy, general purpose pattern classification, artificial neural networks, Karhunen-Loève.

## ÖZET

### GENEL AMAÇLI DÖNME, ÖLÇEKLENME VE ÖTELENME DEĞİŞİMSİZ ÖRÜNTÜ SINIFLANDIRMA SİSTEMİ

Cem Yüceer

Bilgisayar Mühendisliği ve Enformatik Bilimleri Bölümü

Yüksek Lisans

Tez Yöneticisi: Doçent Kemal Oflazer

1992

Yapay sinir ağları son çalışmalarda örüntü sınıflandırma amaçları için kullanılmıştır. Bu çalışmada genel amaçlı dönme, ölçeklenme ve ötelenme değişimsiz örüntü sınıflandırma sistemi sunulmaktadır. Sistemin üç ana öbeği vardır; Karhunen-Loève dönüşümü temelli önışlemci, yapay sinir ağı temelli sınıflandırıcı ve yorumlayıcı. İngiliz abecesi, Japon Katakana abecesi ve bazı geometrik simgeler üzerindeki deneysel çalışmalarla sistemin değişimsizliği sağlama ve örüntü sınıflandırma gücü gösterilmiştir.

Anahtar Kelimeler: Dönme değişimsizliği, ölçeklenme değişimsizliği, ötelenme değişimsizliği, genel amaçlı örüntü sınıflandırma, yapay sinir ağları, Karhunen-Loève.



## ACKNOWLEDGEMENT

I wish to express deep gratitude to my supervisor Assoc. Prof. Kemal Oflazer for his guidance throughout the development of this study. I am grateful to Prof. Mehmet Baray and Assoc. Prof. Uğur Halıcı for their remarks and comments on the thesis. It is a great pleasure to express my thanks to my family for providing morale support and to all my friends for their valuable discussions.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Pattern Recognition and Neural Networks</b>	<b>4</b>
2.1	Overview of Pattern Classification . . . . .	5
2.2	Overview of Previous Works on Pattern Classification with Neural Networks . . . . .	6
2.3	Overview of Artificial Neural Networks . . . . .	8
<b>3</b>	<b>The Pattern Classification System</b>	<b>11</b>
3.1	PREP1: The Preprocessor with Radial Scaling Correction	13
3.1.1	The T-Block . . . . .	14
3.1.2	The S-Block . . . . .	15
3.1.3	The R-Block . . . . .	16
3.2	PREP2: The Preprocessor with Axial Scaling Correction . . . . .	19

CONTENTS	ii
3.3 The Classifier . . . . .	21
3.4 The Interpreter . . . . .	23
<b>4 Experimental Results</b>	<b>25</b>
4.1 Character Recognition on the English Alphabet . . . . .	26
4.2 Character Recognition on the Japanese Katakana Alphabet . . . . .	41
4.3 Classification of Geometric Symbols . . . . .	45
<b>5 Conclusions</b>	<b>59</b>
<b>A Derivation of the R-Block Functions</b>	<b>62</b>
<b>B Derivation of the PREP2 Functions</b>	<b>65</b>

# List of Figures

2.1	Structure and function of a single artificial neuron.	8
2.2	Some activation functions for the artificial neuron.	9
3.1	Block diagram of RST, the pattern classification system. . . . .	12
3.2	Block diagram of the preprocessor. . . . .	13
3.3	A sample pattern before and after the T-Block. . . . .	15
3.4	The sample pattern before and after the S-Block. . . . .	16
3.5	The <i>forward mapping</i> technique and the interpolation property by the <i>reverse mapping</i> technique used in the S-Block. . . . .	17
3.6	The sample pattern before and after the R-Block.	19
3.7	The structure and image feeding strategy to the classifier.	22
4.1	The example patterns and corresponding class numbers for the 26 English letters. . . . .	28

4.2	Classification results with PREP1 for letters A, B, and C rotated by 0, 60, and -60 degrees. . . . .	28
4.3	Classification results with PREP1 for letters A, B, and C scaled by a factor of 1, 0.8, and 0.6. . . . .	29
4.4	Classification results with PREP1 for letters A, B, and C translated diagonally by 0, 6, and -6 pixels. . . . .	29
4.5	Classification results with PREP1 for letters A, B, and C with 0%, 20%, and 40% noise. . . . .	30
4.6	Classification results with PREP1 for letters A, B, and C with random translation, scaling, and rotation applied. . . . .	30
4.7	Image1, a $512 \times 512$ pixel image of English text (left), the classification results with PREP1 for $Th = 0$ (right top), and for $Th = 2$ (right bottom). . . . .	31
4.8	Image2, a $512 \times 512$ pixel image of English text (left), the classification results with PREP1 for $Th = 0$ (right top), and for $Th = 2$ (right bottom). . . . .	32
4.9	Image3, a $512 \times 512$ pixel image of English text (left), the classification results with PREP1 for $Th = 0$ (right top), and for $Th = 2$ (right bottom). . . . .	32
4.10	Classification results with PREP2 on rotated letters. . . . .	33
4.11	Classification results with PREP2 on scaled letters. . . . .	34
4.12	Classification results with PREP2 on translated letters. . . . .	34
4.13	Classification results with PREP2 on noisy letters. . . . .	35

4.14	Classification results with PREP2 on letters with random translation, scaling, and rotation applied. . . . .	35
4.15	Image1 (left), the classification results with PREP2 for $Th = 0$ (right top), and for $Th = 2$ (right bottom).	36
4.16	Image2 (left), the classification results with PREP2 for $Th = 0$ (right top), and for $Th = 2$ (right bottom).	36
4.17	Image3 (left), the classification results with PREP2 for $Th = 0$ (right top), and for $Th = 2$ (right bottom). . . . .	37
4.18	Classification results, with PREP1 (left) and PREP2 (right), on 10 handcrafted English letters.	38
4.19	Classification results for English text of seven different fonts: New Century Schoolbook, Bookman, Boston, Courier, Helvetica, Monaco, and Times. . . . .	39
4.20	Classification results for English text of seven different fonts using a network trained on letters of both <i>Roman</i> and <i>Monaco</i> fonts.	40
4.21	The 111 symbols of the Japanese Katakana Alphabet. . . . .	42
4.22	The 66 unique patterns and their corresponding class numbers. . . . .	42
4.23	Classification results with PREP1 for symbols from Class 1, 2, and 3 rotated by 0, 60, and -60 degrees.	43
4.24	Classification results with PREP1 for symbols from Class 1, 2, and 3 scaled by a factor of 1, 0.8, and 0.6. . . . .	43
4.25	Classification results with PREP1 for symbols from Class 1, 2, and 3 translated diagonally by 0, 6, and -6 pixels.	44

4.26	Classification results with PREP1 for symbols from Class 1, 2, and 3 with 0%, 20%, and 40% noise. . . . .	44
4.27	Classification results with PREP1 for symbols from Class 1, 2, and 3 with random translation, scaling, and rotation applied.	45
4.28	Image4, a $512 \times 512$ pixel image of Japanese text (left) and the classification result with PREP1 (right). . . . .	46
4.29	The patterns and class numbers for the five main geometric symbols; circular, cross, line, square-like, and triangular.	46
4.30	Classification results with PREP1 for geometric symbols rotated by 0, 60, and -60 degrees.	47
4.31	Classification results with PREP1 for geometric symbols scaled by a factor of 1, 0.8, and 0.6. . . . .	48
4.32	Classification results with PREP1 for geometric symbols translated diagonally by 0, 6, and -6 pixels.	49
4.33	Classification results with PREP1 for geometric symbols with 0%, 20%, and 40% noise. . . . .	50
4.34	Classification results with PREP1 for geometric symbols with random translation, scaling, and rotation applied.	51
4.35	Classification results with PREP2 for rotated geometric symbols.	52
4.36	Classification results with PREP2 for scaled geometric symbols. .	53
4.37	Classification results with PREP2 for translated geometric symbols. . . . .	54

## LIST OF FIGURES

vii

- |      |  |    |
|------|--|----|
| 4.38 | Classification results with PREP2 for noisy geometric symbols.   | 55 |
| 4.39 | Classification results with PREP2 for geometric symbols with random translation, scaling, and rotation applied.        | 56 |
| 4.40 | Classification results, with PREP1 (left) and PREP2 (right), on patterns formed by merging two geometric symbols.      | 57 |
| 4.41 | Classification results, with PREP1 (left) and PREP2 (right), on distorted patterns of the five main geometric symbols. | 58 |



# List of Tables

4.1	Number of training epochs and success ratios for some network configurations. . . . .	26
4.2	Performance results for English text of seven different fonts. . . .	41

# Chapter 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 '60s and '70s 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 [5, 7]. 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 featural representation. The stage of feature determination and extraction strictly determines the success of the system, since from thereon the image is represented by this featural 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 translational invariancy. Even for a limited system designed for classifying only a determined 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 invariancies. The problem will be impractical if such a system is intended for general purpose classification, or to say it is aimed to classify any type of patterns.

The scope of this work was to develop a general purpose pattern classification system which is rotation, scaling, and translation invariant. In order to differentiate any type of patterns, a neural network based system has been developed to automatically select the necessary features, provided that some number of example patterns from each class are supplied. Artificial neural networks have been chosen for the feature extraction and class determination process for their widely applied learning behaviors [1, 3, 4, 8, 12, 16]. The rotational, scaling, and translational invariances has been maintained by a newly developed Karhunen-Loève transformation based preprocessor.

The text is organized as follows: Chapter 2 has three subsections. The first is on the pattern classification problem. The second overviews the previous research on classifying patterns using artificial neural networks. The last subsection supplies basic information on artificial neural networks. Chapter 3 describes the proposed pattern classification system. The three main blocks – preprocessor, classifier, and interpreter – are defined. The preprocessor is further divided into three subblocks, called T-Block, S-Block, and R-Block. The definitions are clarified with mathematical formulations. Chapter 4 gives the experimental results on three different pattern classification problems, the English alphabet, the Japanese Katakana alphabet, and five main geometric symbols. The conclusions in Chapter 5 are followed by two appendices including the mathematical derivations of the two group of formulae given in the text.

## Chapter 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. A computer vision system must incorporate a pattern recognition capability. For some simple and frequently encountered patterns, the recognition process can be a straightforward task. However, when patterns are complex or when pattern characteristics can not be predicted beforehand then one needs a high-level system to perform pattern recognition. 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 according to the information that was extracted from the example patterns previously supplied to the system. Inputs are in the form of digitized binary-valued 2-D images containing the pattern to be classified. This representation of the 2-D 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. Two pattern classification techniques are major, template matching and feature extraction. In the simplest case, template matching refers to the comparison of 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 to a group of quantitative descriptors called feature values. These features are usually predefined by the designer and chosen to be independent from each other [3, 4, 12].

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. That is, in order to attain a high success ratio in classification features should satisfy the following two requirements [12]:

- **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 increase, 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 auto-correlation techniques are used. Consider the case of complex patterns which are 2-D perspective projections of 3-D 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. That is, the stored image should be an averaged version of a number of templates. Statistical decision theory helps in the mathematical construction of the generalized template and the recognition analysis.

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 would be 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 Overview of Previous Works on Pattern Classification with Neural Networks

There are two main groups of previous works on pattern classification with artificial neural networks. The first group, defines a featural representation for the image. In other words, the input image is transformed to some predefined features before classification is performed – a process called feature extraction.

Examples of such work are extraction of image information using regular moments [20], Zernike moments [12], Fourier descriptors [15, 19, 20, 23], autoregressive models [11], image representation by circular harmonic expansion [10], syntactic pattern recognition applications [6], Karhunen-Loève expansion [13], polar-coordinate Fourier transform [2], transforming the image to another 2-D representation [14]. Kollias et.al., in their work have transformed the input image to another 2-D representation, called  $(a,b)$  plane, and performed classification using higher-order networks [14]. Khotanzad et.al., in their work have used Zernike moments to achieve rotational invariancy in classification [12]. They have computed some finite number of Zernike moments of the given image and performed classification on these moment values. Le Cun et.al., have first skeletonized the pattern and then scanned the whole image with a  $7 \times 7$  window, where the window templates were designed to detect some predefined features [3]. The classification was performed on the computed values. Kirby et.al., have used the Karhunen-Loève expansion of the input image as its featural representation and performed classification on the expansion [13]. Among these works, the common approach is converting the input image to some compact and transformation invariant form, and then classifying this representation using artificial neural networks. The transformation mentioned is functionally the process of feature extraction, since the original image is converted to a form in which classification information is easily detected. However note that, features for distinguishing letters are not same as the features for distinguishing complex patterns such as fingerprints or faces. Hence, a general pattern classifier which is capable of performing successful classifications for large number of problems will be impractical unless the classification system is designed to extract the features itself.

Second and newly developing group prefers the features to be determined and extracted by the artificial neural network itself. Martin et.al., in their work have fed the neural network by size-normalized gray scale images [17]. They report that *“the generalization performance is surprisingly insensitive to characteristics of the network architecture, as long as enough training samples are used and there*



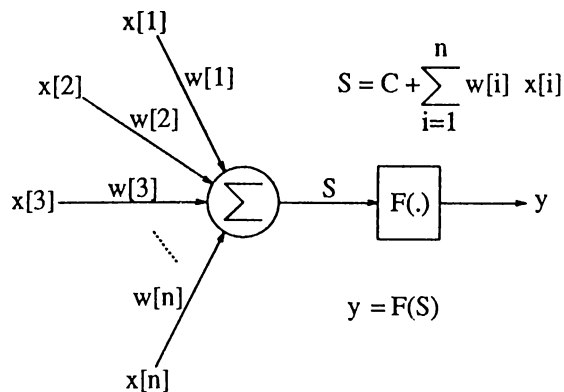


Figure 2.1: Structure and function of a single artificial neuron.

*is sufficient capacity to support training to high levels.”*

For the system introduced through this text, an artificial neural network has been employed to extract the essential features from the input set and to perform the classification. The system also includes a preprocessor block that maps the rotated, scaled, and translated input images to a canonical form which is then classified by the neural network. Finally, the outputs of the classifier are interpreted by the interpreter block.

## 2.3 Overview of Artificial Neural Networks

Artificial neural networks are computational models inspired from the structure of the brain. They are massively parallel networks comprising large number of simple processing units, called artificial neurons. Artificial neural networks constitute an alternative knowledge representation paradigm for artificial intelligence and cognitive science. The neuron structure given in Figure 2.1 is the basic building block of such networks. It performs a weighted summation over its inputs, where these inputs may be the outputs of other neurons or may be external to the network. The threshold, a local value for each neuron, is added to the sum. Then

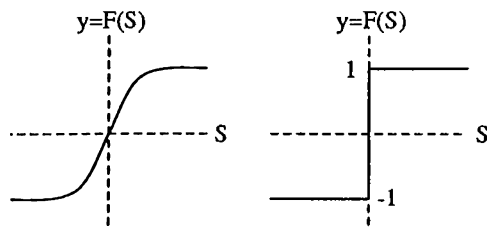


Figure 2.2: Some activation functions for the artificial neuron.

an activation function – also called as the limiting or squashing function – is applied on the resulting sum to determine the neuron’s output value. Widely used activation function types are various sigmoids, and hard-limiting (thresholding) functions, some of which are shown in Figure 2.2.

The weights associated with the inter-neuron connections represent the influence of its inputs over its output – or activation – value. Depending on the sign of the weight an input may *excite* or *inhibit* the neuron. The learning behavior of artificial neural networks are accomplished by the learning algorithms continuously updating these weights until the given inputs produce the desired outputs. There are two main types of learning algorithms:

- **Supervised learning** : The user should supply the desired outputs while feeding the input pattern,
- **Unsupervised learning** : The network itself gathers the necessary information from the input patterns, hence develops its own representation.

There is a variety of interconnection topologies between artificial neurons. The most common one is the class of multilayer feed-forward networks, where neurons are grouped as layers and connections between neurons in subsequent layers are permitted. One end of the layered structure is named as the input layer, while the other end as output layer. The inputs are fed from the input layer and the outputs are expected from the output layer. Using artificial neural networks for pattern

classification purposes has become very popular in recent years. Especially feed-forward type nets are frequently applied to many recognition problems [1, 3, 4, 8, 12, 16].

## Chapter 3

# The Pattern Classification System

Feed-forward type neural networks, when used as a pattern classifier, are highly sensitive to transformations like rotation, scaling, and translation. This behavior emerges from the fact that throughout the training phase the area of interest concentrates mainly on the region where the pattern lies. Hence the weights associated with the input pixels that are out of this region finally decays to zero. Since 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. Note that, a transformation of any kind may push the pattern out of this region degrading 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 transformationally distorted or noisy versions of a pattern to a unique and stable output pattern.

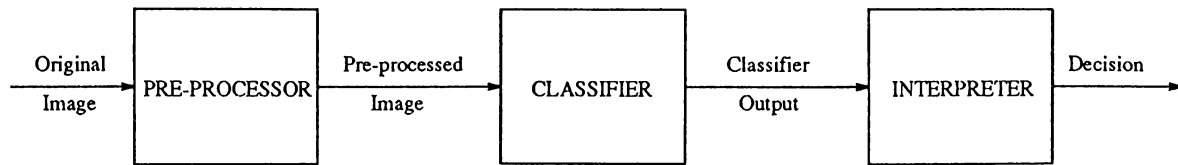


Figure 3.1: Block diagram of RST, the pattern classification system.

- 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 functioning 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 [13]. 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 can extract information during the training process [21]. The hidden layers and the neurons in the hidden layers detect the relevant features in the images.

RST, the pattern classification system introduced in this work 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 3.1 shows the block diagram for the pattern classification system.

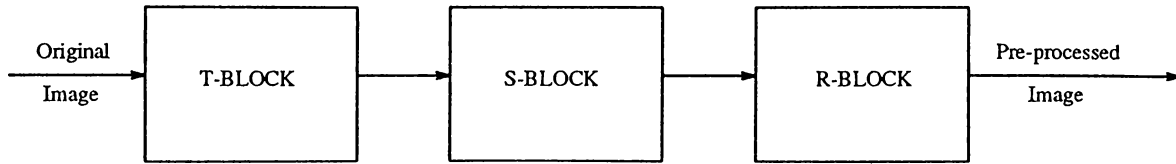


Figure 3.2: Block diagram of the preprocessor.

### 3.1 PREP1: The Preprocessor with Radial Scaling Correction

Feature extraction does not take place in the preprocessor. Its function is to provide rotational, scaling, and translational invariancy on the input image. Both the input and the output of the preprocessor are images in pixel-map form. Certain functions are performed on the input image and the resulting modified image is generated at the output. The preprocessor has three cascaded blocks, namely R-Block, S-Block, and T-Block. The block diagram is given in Figure 3.2. R-Block is to maintain rotational invariancy, while S-Block is for scaling invariancy, and T-Block is for 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, named PREP1, T-Block comes first, 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. T-Block will maintain translational invariancy and the origin will be the pivot point for the scaling and rotation blocks. Further, placing S-Block in front of the R-Block will bring the following two advantages:

- S-Block will mostly prevent the pattern from flowing out of the image by a rotation operation. Consider an image containing a bone pattern where the bone 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 degrees. However, performing scaling correction beforehand would bring a radial boundary for the pattern decreasing the effect of the mentioned problem.

- S-Block will adjust the number of pixels that are active – called on-pixels –, and will regulate the information flowing into the R-Block. The performance of 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

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. 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:

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

$$P = \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \quad (3.1)$$

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

$$x_{av} = \frac{1}{P} \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i \quad (3.2)$$

$$y_{av} = \frac{1}{P} \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot y_j \quad (3.3)$$

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

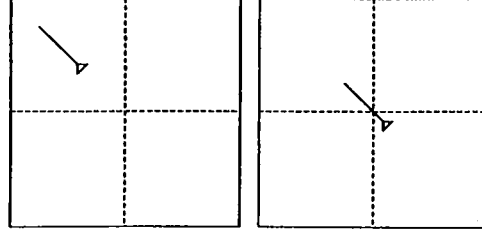


Figure 3.3: A sample pattern before and after the T-Block.

The mapping function for the translation invariant image is:

$$f_T(x_i, y_j) = f(x_i - x_{av}, y_j - y_{av}) \quad (3.4)$$

Function  $f_T(\cdot)$  is similar to  $f(\cdot)$  except that it is for the output image of the T-Block which is translation invariant. Figure 3.3 shows the function of the T-Block on a sample pattern.

### 3.1.2 The S-Block

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 and regulates the information flowing into the R-Block. The average radius is computed as:

$$r_{av} = \frac{1}{\sum_{i=1}^N \sum_{j=1}^N f_T(x_i, y_j)} \sum_{i=1}^N \sum_{j=1}^N f_T(x_i, y_j) \cdot \sqrt{x_i^2 + y_j^2} \quad (3.5)$$

and the scale factor,  $s$ , is given by:

$$s = \frac{R}{r_{av}} \quad (3.6)$$

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



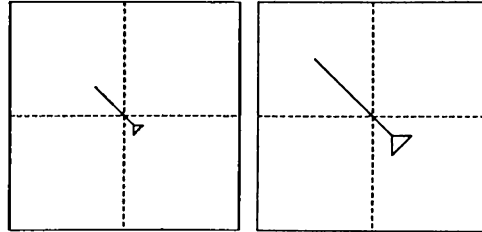


Figure 3.4: The sample pattern before and after the S-Block.

The mapping function for the scaling invariant image is:

$$f_{TS}(x_i, y_j) = f_T(s \cdot x_i, s \cdot y_j) \quad (3.7)$$

Function  $f_{TS}(\cdot)$  is similar to  $f_T(\cdot)$  except that it is for the output image of the S-Block which is scaling invariant. Figure 3.4 shows the function of the S-Block on the sample image processed by the T-Block.

Equation 3.7, the mapping function for the S-Block, embeds the nice 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 disconnection 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 Figure 3.5.

### 3.1.3 The R-Block

R-block maintains rotational invariancy by rotating the image so that the direction given by *R-Block function* coincides with the *x-axis*. The *R-Block function* computes the direction of maximum variance for pixels in a given input image.

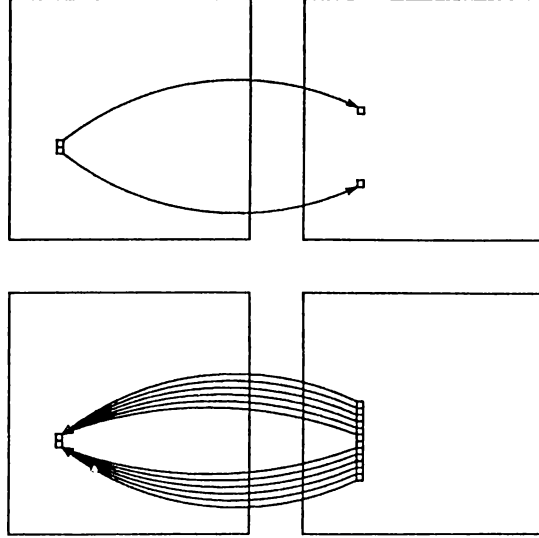


Figure 3.5: The *forward mapping* technique and the interpolation property by the *reverse mapping* technique used in the S-Block.

The derivation of the function is based on the Karhunen-Loève transformation which has been referred in some applications [9, 13, 16]. The transformation states: Given a set of vectors, the eigenvector that corresponds to the greatest eigenvalue of the covariance matrix calculated from the set of vectors, points in the direction of maximum variance [9, 13]. 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 2-D 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. Hence, from the above discussions, the rotation parameters can be derived as:

Define:

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \frac{1}{\sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j)} \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix} \quad (3.8)$$

$$P = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \quad (3.9)$$

$$T_{xx} = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \cdot x_i^2 \quad T_{yy} = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \cdot y_j^2 \quad (3.10)$$

$$T_{xy} = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \cdot x_i \cdot y_j \quad (3.11)$$

The covariance matrix defined as:

$$C = \frac{1}{P} \left( \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix} \right) \left( \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix} \right)^T \quad (3.12)$$

can be simplified to:

$$C = \left( \frac{1}{\sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j)} \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix} \begin{bmatrix} x_i \\ y_j \end{bmatrix}^T \right) - \begin{bmatrix} m_x \\ m_y \end{bmatrix} \begin{bmatrix} m_x \\ m_y \end{bmatrix}^T \quad (3.13)$$

Since translational 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 comes out as:

$$C = \begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix} \quad (3.14)$$

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

$$\sin \theta = \frac{(T_{yy} - T_{xx}) + \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}{\sqrt{2 \cdot [\sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx})] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}} \quad (3.15)$$

$$\cos \theta = \frac{2 \cdot T_{xy}}{\sqrt{2 \cdot [\sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx})] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}} \quad (3.16)$$

The mapping function for the rotation invariant image is:

$$f_{TSR}(x_i, y_j) = f_{TS}(\cos \theta \cdot x_i + \sin \theta \cdot y_j, -\sin \theta \cdot x_i + \cos \theta \cdot y_j) \quad (3.17)$$

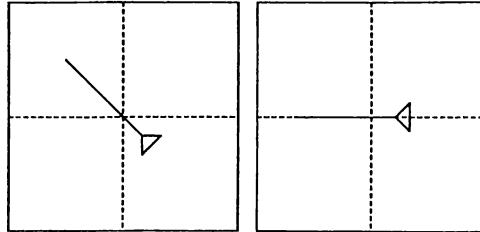


Figure 3.6: The sample pattern before and after the R-Block.

Function  $f_{TSR}(\cdot)$  is similar to  $f_{TS}(\cdot)$  except that it is for the output image of the R-Block which is rotation invariant. Figure 3.6 shows the function of the R-Block on the sample pattern processed by the S-Block.

## 3.2 PREP2: The Preprocessor with Axial Scaling Correction

PREP1, the preprocessor introduced in the previous section, performs radial scaling correction. That is, PREP1 uses the same scaling factor along all directions. A different approach would be using different scaling factors along different axes. This type of scaling correction will uniquely map patterns regardless if they are thinner, thicker, shorter, or taller than their original forms when they were first presented. Hence the preprocessor with axial scaling correction, named PREP2, is developed after this approach. PREP2 has two main differences from the previous version:

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

In order to maintain rotational, scaling, and translational invariancy PREP2 computes the corresponding values as (detailed derivation is given in Appendix B):

$$P = \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \quad (3.18)$$

$$x_{av} = \frac{1}{P} \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i \quad (3.19)$$

$$y_{av} = \frac{1}{P} \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot y_j \quad (3.20)$$

$$T_{xx} = \left( \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i^2 \right) - P \cdot x_{av}^2 \quad (3.21)$$

$$T_{yy} = \left( \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot y_j^2 \right) - P \cdot y_{av}^2 \quad (3.22)$$

$$T_{xy} = \left( \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i \cdot y_j \right) - P \cdot x_{av} \cdot y_{av} \quad (3.23)$$

$$\sin \theta = \frac{(T_{yy} - T_{xx}) + \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}{\sqrt{2 \cdot \left[ \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx}) \right] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}} \quad (3.24)$$

$$\cos \theta = \frac{2 \cdot T_{xy}}{\sqrt{2 \cdot \left[ \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx}) \right] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}} \quad (3.25)$$

$$s_x = \sqrt{\frac{R_x \cdot P}{T_{xx}(\cos \theta)^2 + 2 \cdot T_{xy} \cdot \cos \theta \cdot \sin \theta + T_{yy}(\sin \theta)^2}} \quad (3.26)$$

$$s_y = \sqrt{\frac{R_y \cdot P}{T_{xx}(\sin \theta)^2 - 2 \cdot T_{xy} \cdot \cos \theta \cdot \sin \theta + T_{yy}(\cos \theta)^2}} \quad (3.27)$$

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:

$$f_{TRS}(x_i, y_j) = f(s_x \cdot (\cos \theta \cdot (x_i - x_{av}) + \sin \theta \cdot (y_j - y_{av})), s_y \cdot (-\sin \theta \cdot (x_i - x_{av}) + \cos \theta \cdot (y_j - y_{av}))) \quad (3.28)$$

Function  $f_{TRS}(\cdot)$  is similar to  $f(\cdot)$  except that it is for the output image of PREP2 which is rotation, scaling, and translation invariant. There is a one step mapping from the original image function to the preprocessor output function. 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* [21]. 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 the 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 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.

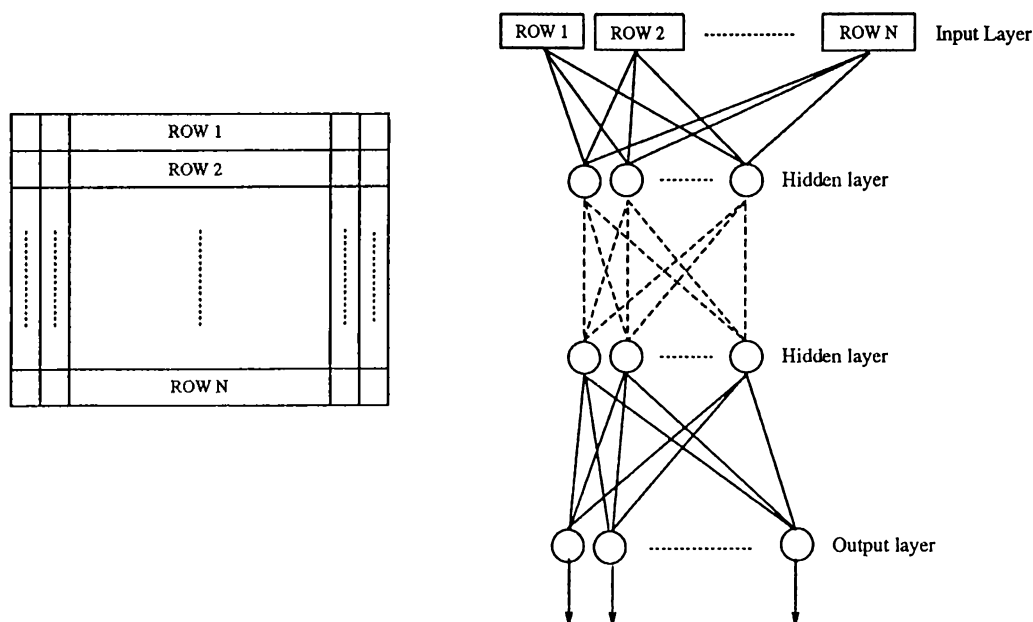


Figure 3.7: The structure and image feeding strategy to the classifier.

The general structure of the neural network classifier is given in Figure 3.7. 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 has two phases:

1. **The training phase** : The preprocessed forms of the images containing the sample patterns from each class are fed to the network, while the desired output patterns are supplied to the output neurons. The artificial neural network learns to produce the desired outputs when the sample inputs are given to the system.
2. **The classification phase** : There is no learning in the classification phase, instead the network only runs through forward passes mapping preprocessed images to output patterns. The artificial neural network is initialized with

the weights obtained at the training phase. The preprocessed image is fed to the classifier and activation values of the neurons in the consecutive layers are calculated towards the output layer. Finally, the output pattern, formed by the activation values of the output neurons, is calculated and the forward pass is over. The output pattern contains the classification information. These values are passed to the interpreter for the decision to be made.

### 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. In other words, the membership of the input pattern to each class will be represented by the activation value of the corresponding output neuron. A value close to 1 would be interpreted as a strong membership, while a value close to 0 would point a loose membership.

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. Since some input images will not contain any meaningful pattern or will contain patterns irrelevant with the previously sample patterns, a maximum finder block will fail to classify by deciding on one of the classes. Some level of thresholding can be applied to the outputs so that none of the classes is selected if none of the outputs exceeds this predetermined threshold. However this method may indicate multiple classes for certain inputs. This will be typical when two or more of the classifier's outputs exceed the threshold. 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 the introduced pattern classification system employs the last mentioned method. If the ratio of the maximum output to the next highest output exceeds the predetermined number, the interpreter reports that a discrimination could be made and the pattern belong to the class with the maximum output. If not, then the interpreter reports that no unique discrimination could be made. This simple method performs well in the evaluation of the classifier outputs.

## Chapter 4

# Experimental Results

Based on the formulations described in the previous sections, a general purpose pattern classification system has been developed on Sun workstations using the graphics environment SunView [22]. The system is capable of performing rotational, scaling, and translational transformations as well as adding noise on an input image. There are two type of inputs to the system: first, in the form of a single grid image; second, as a camera scanned raster image. It directly performs classification on the single grid image, while for the raster image it scans and detects patterns and performs classifications thereon. The artificial neural network block uses the resulting weights of the backpropagation simulator HYPERBP developed on the iPSC/2 hypercube multicomputer [18]. Hence, in order to experiment on some new pattern classification problem the process goes on as follows: The example patterns are first preprocessed, then the preprocessed images are copied into iPSC/2 to be used as input patterns for the backpropagation simulation. The resulting weights are copied back to Sun machines and used in the classification system thereon.

Network configuration	Number of epochs	Success ratio
1024 - 5 - 26	no convergence	-
1024 - 10 - 26	160	72%
1024 - 15 - 26	100	86%
1024 - 20 - 26	50	93%
1024 - 25 - 26	50	90%
1024 - 30 - 26	40	93%
1024 - 10 - 5 - 26	no convergence	-
1024 - 10 - 10 - 26	290	72%
1024 - 20 - 5 - 26	no convergence	-
1024 - 20 - 10 - 26	230	87%
1024 - 20 - 20 - 26	110	84%

Table 4.1: Number of training epochs and success ratios for some network configurations.

## 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 English alphabet is a multilayer feed-forward network. There are 1024 input nodes, each one corresponding to one of the pixels of the  $32 \times 32$  input image. An output neuron is reserved for each pattern class, making up a total of 26 output neurons. These output neurons will compute the membership function for the pattern to the corresponding class. A value close to 1 will be interpreted as a strong membership, while a value close to 0 will point a loose membership. The number of hidden layers and the number of neurons in each hidden layer is found by trial-and-error, which is typical for most multilayer feed-forward network applications [1, 3, 4, 8, 12, 16]. Table 4.1 summaries the performances for different network configurations. The representation used in *network configuration* is such that the first number

shows the number of input nodes, the last number shows the number of output neurons, and the numbers between them show the number of neurons in the corresponding hidden layer. The *success ratio* is the averaged percentage of the correctly classified patterns for three  $512 \times 512$  pixel images, namely Image1, Image2, and Image3 given in figures Figure 4.7 through Figure 4.9. *Number of epochs* is the number of epochs needed in the training phase until the network successfully classifies all of the example patterns. One *epoch* refers to one complete pass over all the example patterns. As seen from Table 4.1, the 1024 - 20 - 26 network has the best *success ratio* vs. *number of neurons* figure. Hence, the network has been chosen to have 1024 input nodes, 20 neurons in the single hidden layer, and 26 neurons in the output layer.

In the training phase, the network is trained on the input example patterns until it manages to successfully classify the letters. Note that R-Block rotates the pattern until the computed orientation coincides with the *x-axis*. Since some pattern and its 180 degrees rotated version will have the same orientation, R-Block will not be able to differentiate between them and will perform same angle of rotation on both patterns. The resulting mappings will conserve the 180 degrees 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. Therefore, the training set is formed of 52 example patterns where there are two preprocessed patterns for each class. being the preprocessed forms of the letters. The example patterns for the English letters and their corresponding classes are given in Figure 4.1.

Figure 4.2 through Figure 4.6 give the system performance on single grid images. The employed preprocessor is PREP1. Images are  $32 \times 32$  pixels. First column is the original image given to the system. Second column is the preprocessed version of the original image, and finally third column is the resulting decision. The class name and value of the corresponding output neuron are given.

Figure 4.7 through Figure 4.9 give the system performance on camera scanned

A B C D E F G H I 1 2 3 4 5 6 7 8 9  
 J K L M N O P Q R 10 11 12 13 14 15 16 17 18  
 S T U V W X Y Z 19 20 21 22 23 24 25 26

Figure 4.1: The example patterns and corresponding class numbers for the 26 English letters.

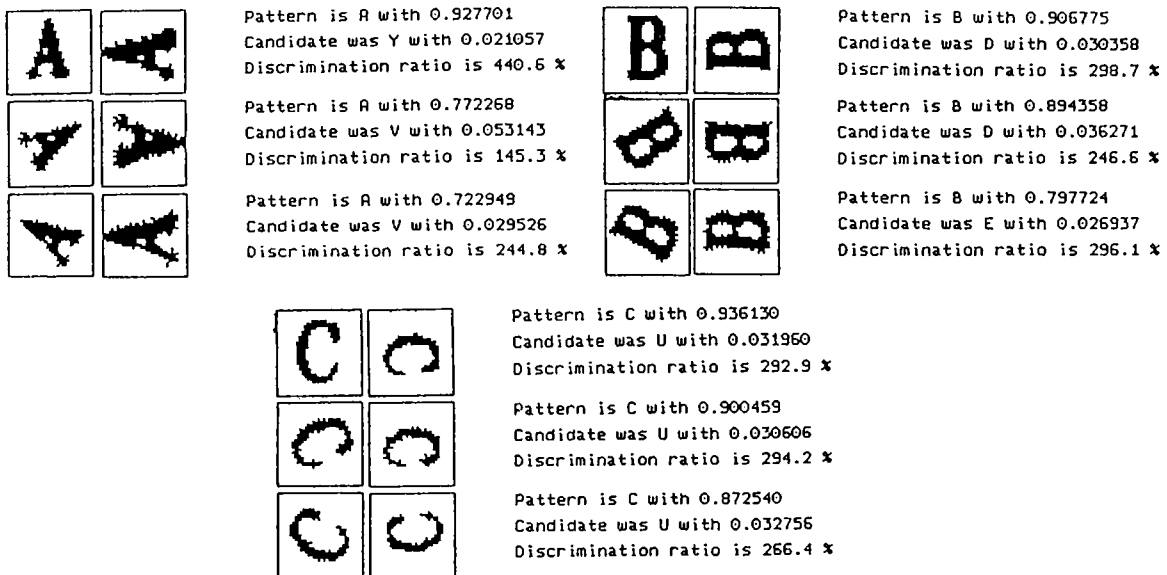


Figure 4.2: Classification results with PREP1 for letters A, B, and C rotated by 0, 60, and -60 degrees.

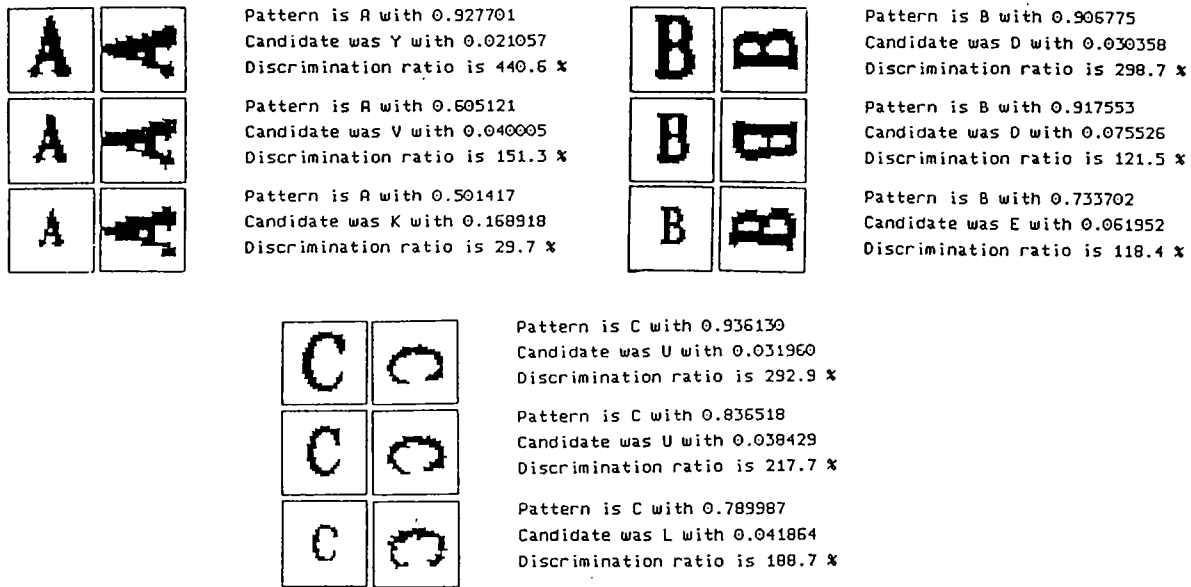


Figure 4.3: Classification results with PREP1 for letters A, B, and C scaled by a factor of 1, 0.8, and 0.6.

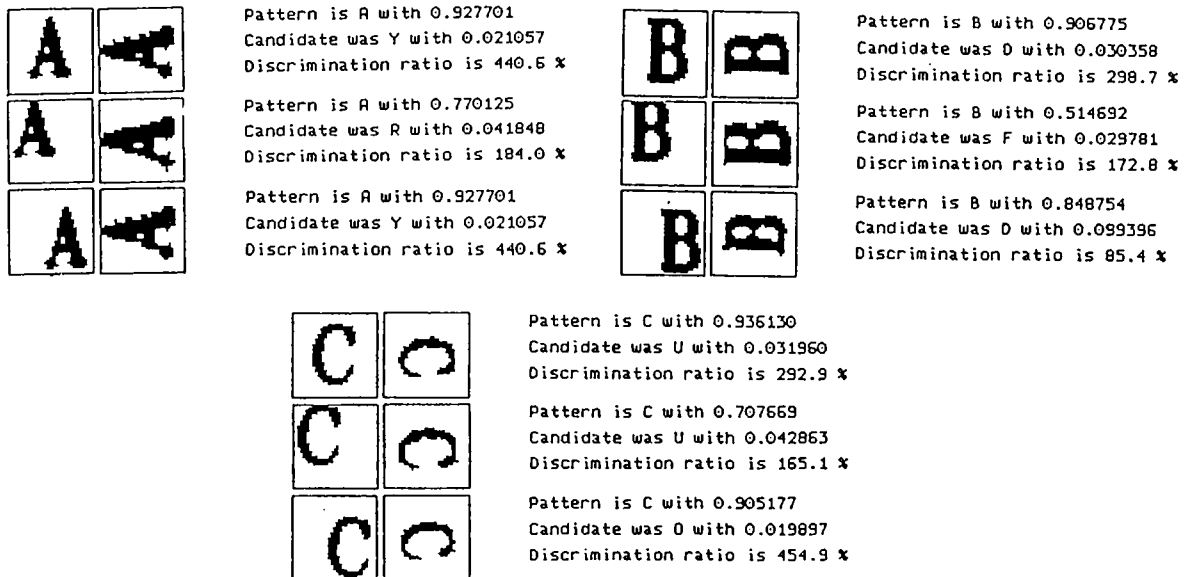


Figure 4.4: Classification results with PREP1 for letters A, B, and C translated diagonally by 0, 6, and -6 pixels.

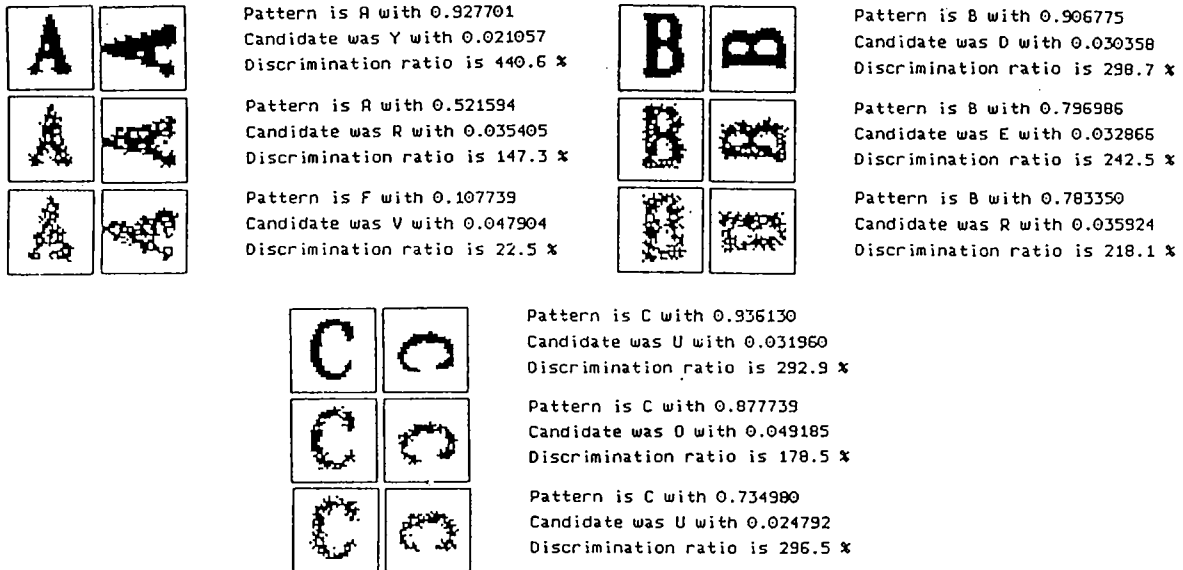


Figure 4.5: Classification results with PREP1 for letters A, B, and C with 0%, 20%, and 40% noise.

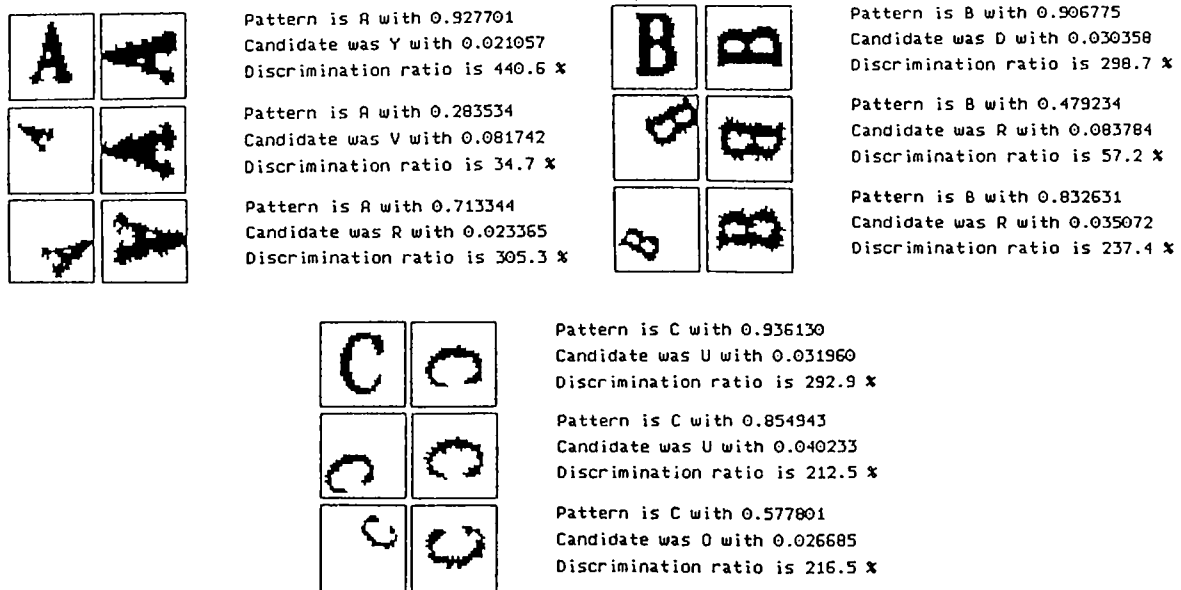


Figure 4.6: Classification results with PREP1 for letters A, B, and C with random translation, scaling, and rotation applied.

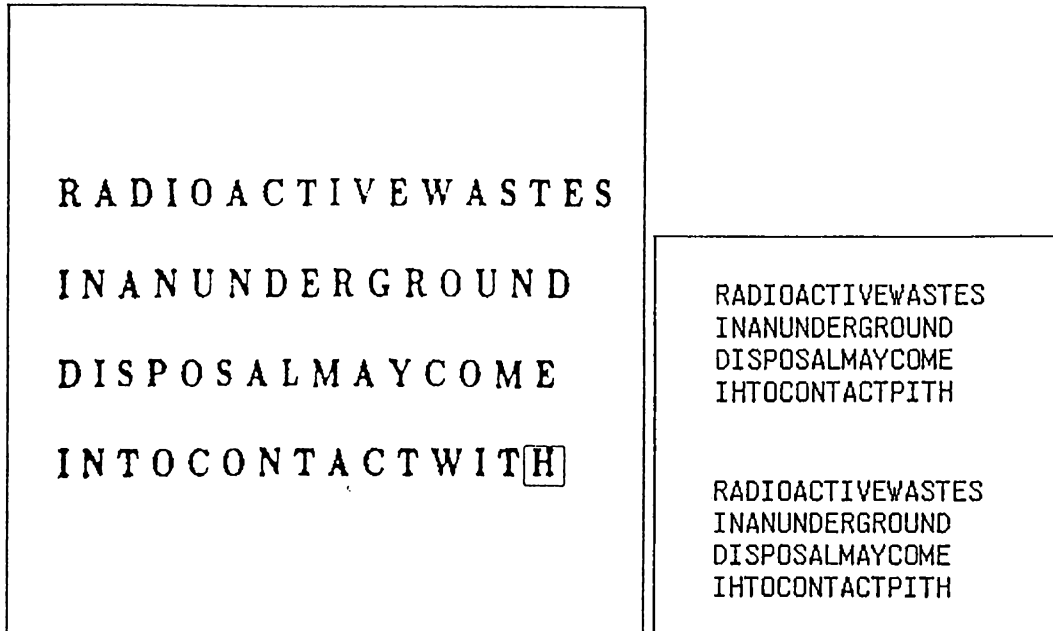


Figure 4.7: Image1, a  $512 \times 512$  pixel image of English text (left), the classification results with PREP1 for  $Th = 0$  (right top), and for  $Th = 2$  (right bottom).

raster images of English text. The preprocessor is PREP1. The images are  $512 \times 512$  pixels. Classification is done by segmenting each letter, then positioning the  $32 \times 32$  grid on the letter, and performing a neural network forward pass. Segmentation process is for boxed-discrete characters. That is, it is assumed that each single pattern can be boxed without interference from the neighboring patterns. For a better evaluation, two separate passes over the image is performed, first when the interpreter threshold set to 0 and second the threshold set to 2. First forces the interpreter to choose one of the classes, while the second enables the interpreter to report *no discrimination* between the classes, represented by “?” in the output. The average success ratio for the given images of English text is 93%, which is considered as *acceptable*, for a general purpose pattern classifier. The number of floating point operations performed during a forward pass is:

$$\text{multiplications} + \text{additions} : 2 \times (1024 \times 20 + 20 \times 26)$$

$$\text{total} : 42000$$



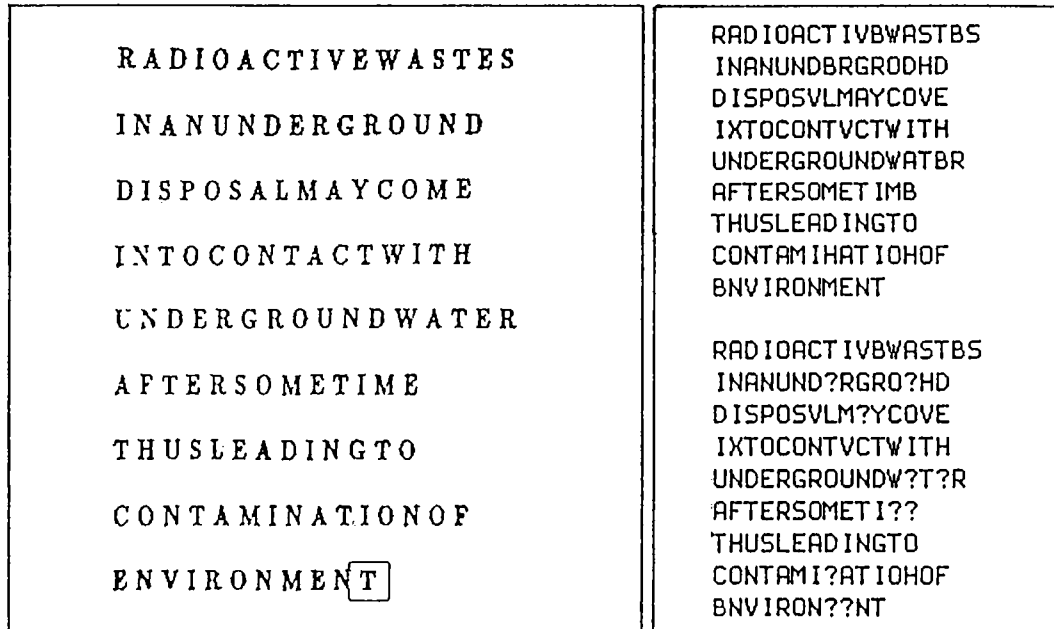


Figure 4.8: Image2, a  $512 \times 512$  pixel image of English text (left), the classification results with PREP1 for  $Th = 0$  (right top), and for  $Th = 2$  (right bottom).

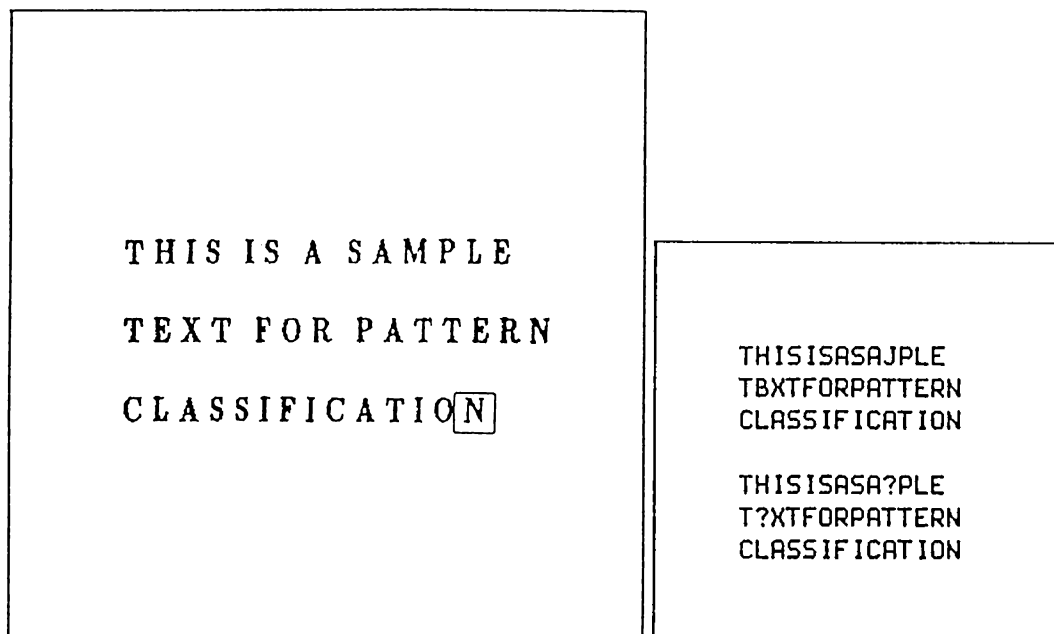


Figure 4.9: Image3, a  $512 \times 512$  pixel image of English text (left), the classification results with PREP1 for  $Th = 0$  (right top), and for  $Th = 2$  (right bottom).

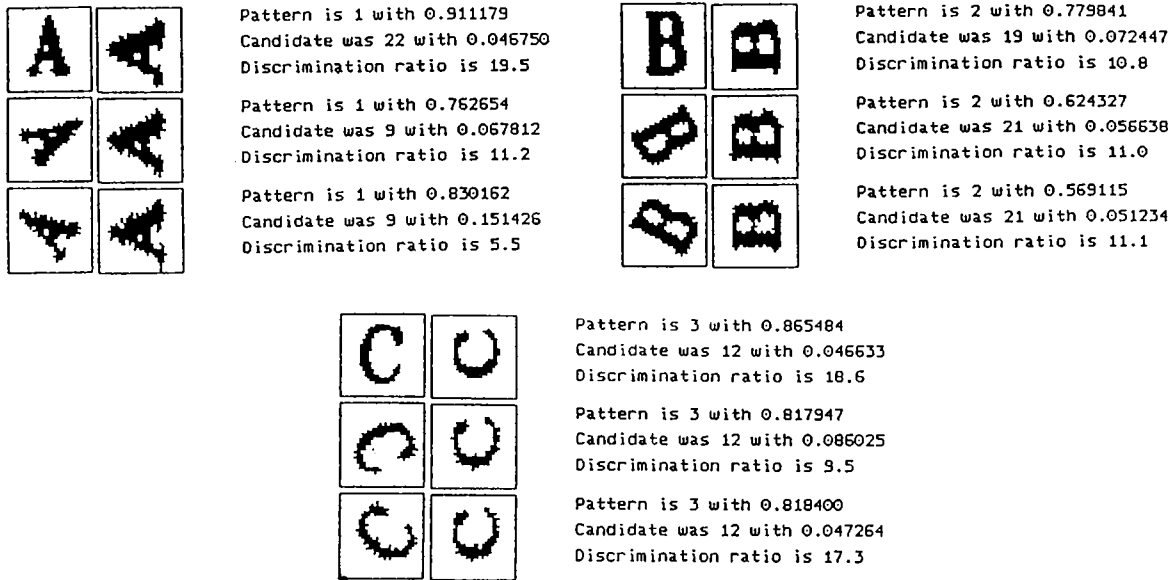


Figure 4.10: Classification results with PREP2 on rotated letters.

On Sun4 Sparc workstations, the average classification speed of the system is 7 patterns per second.

Figure 4.10 through Figure 4.14 give performance on single grid images, where the employed preprocessor is PREP2.

Figure 4.15 through Figure 4.17 give the performance on the previously introduced images of English text. The preprocessor is PREP2, and average success ratio for these images is 91%.

Figure 4.18 compares the classification results for two versions of the system, one with PREP1 and other PREP2. The input patterns are 10 handcrafted English letters.

Figure 4.19 gives the system performance for input images containing English text of different fonts. The seven fonts are: New Century Schoolbook, Bookman, Boston, Courier, Helvetica, Monaco, and Times. Figure 4.20 gives the system

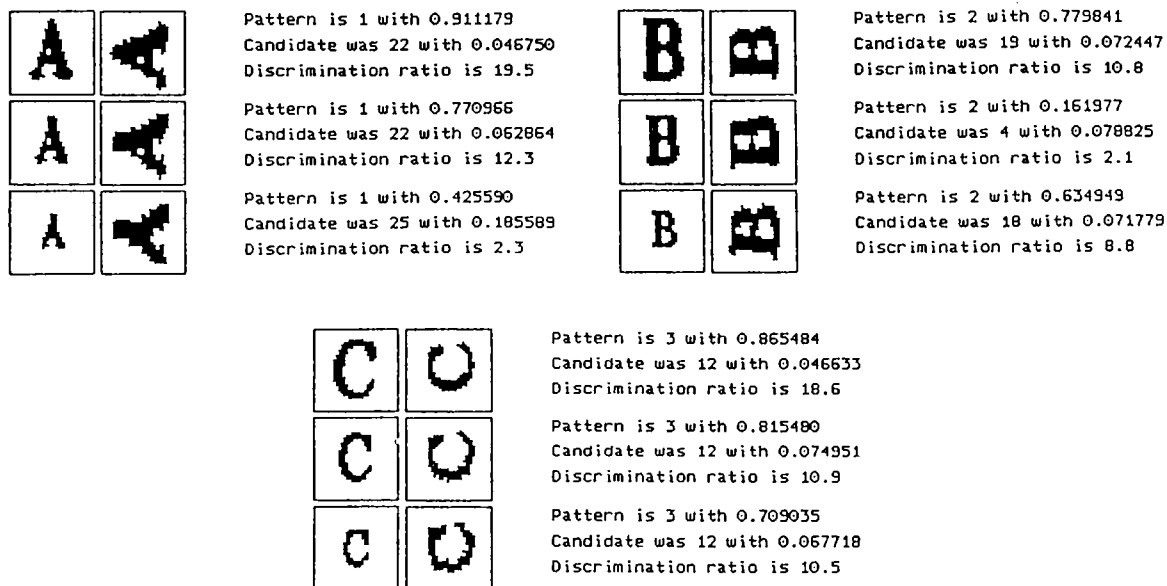


Figure 4.11: Classification results with PREP2 on scaled letters.

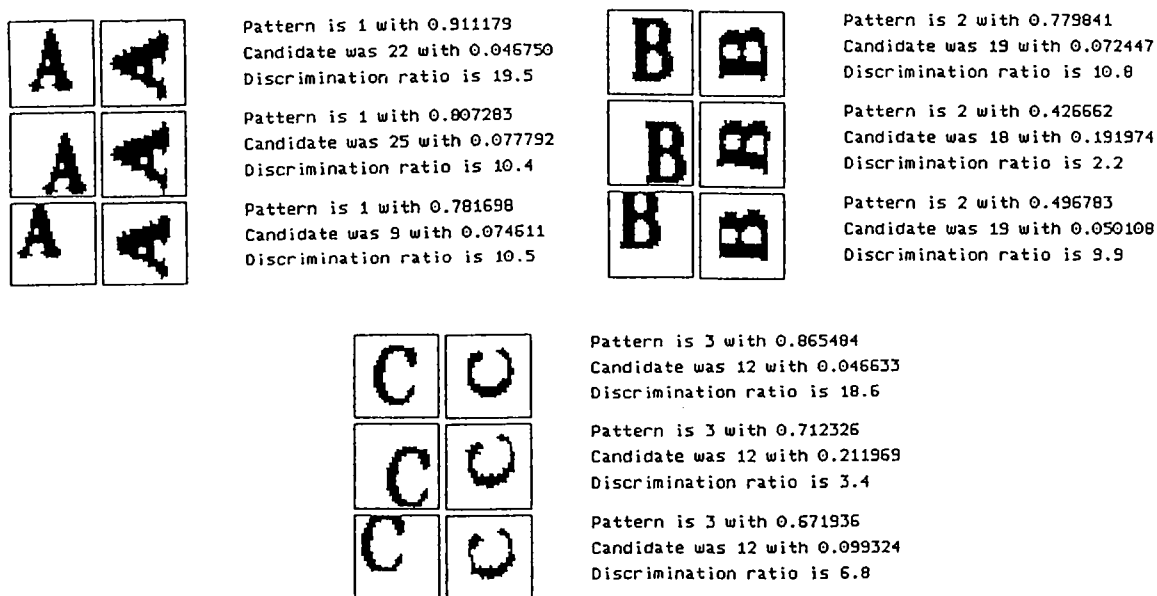


Figure 4.12: Classification results with PREP2 on translated letters.

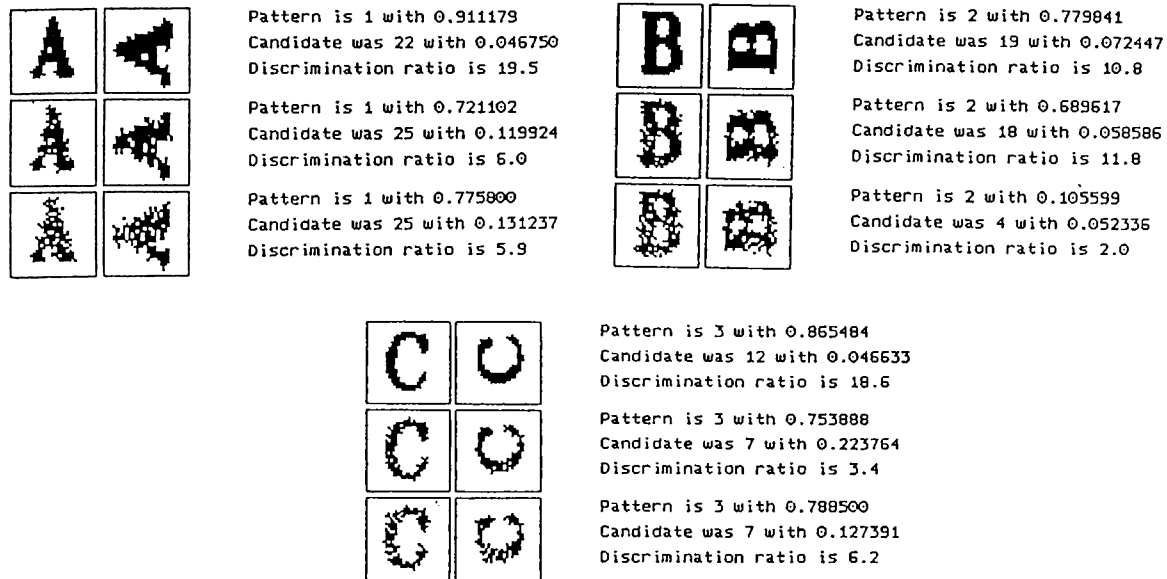


Figure 4.13: Classification results with PREP2 on noisy letters.

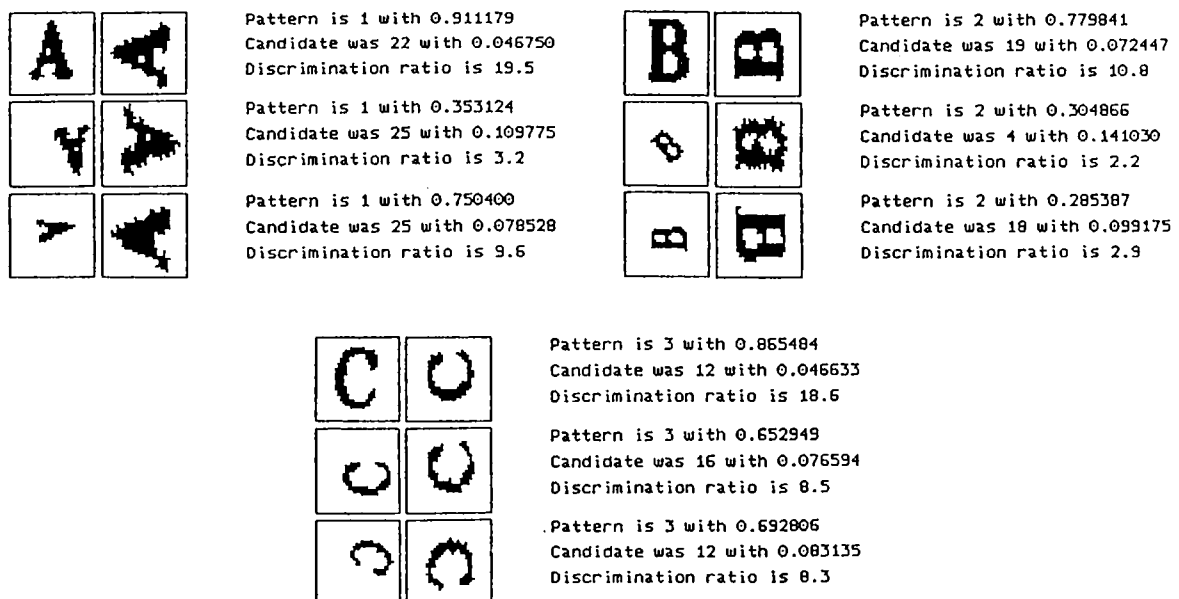


Figure 4.14: Classification results with PREP2 on letters with random translation, scaling, and rotation applied.

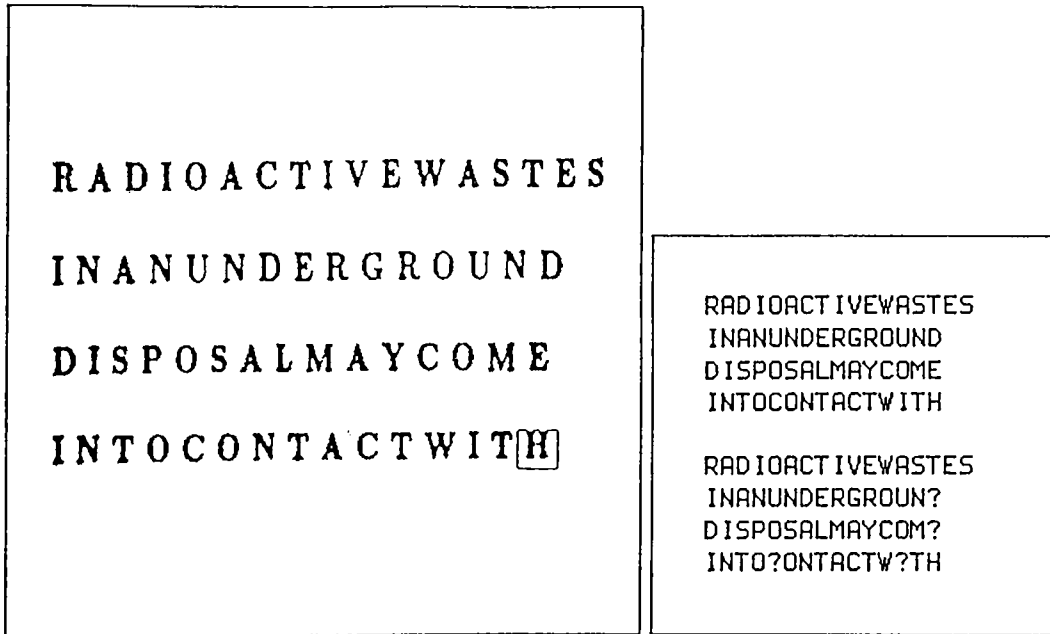


Figure 4.15: Image1 (left), the classification results with PREP2 for  $Th = 0$  (right top), and for  $Th = 2$  (right bottom).

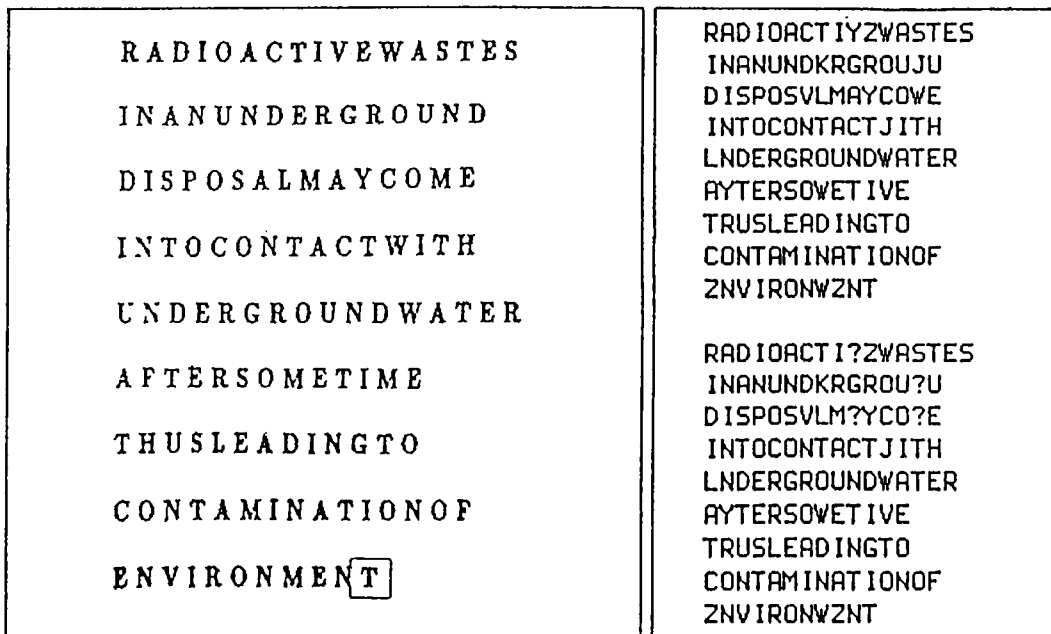


Figure 4.16: Image2 (left), the classification results with PREP2 for  $Th = 0$  (right top), and for  $Th = 2$  (right bottom).

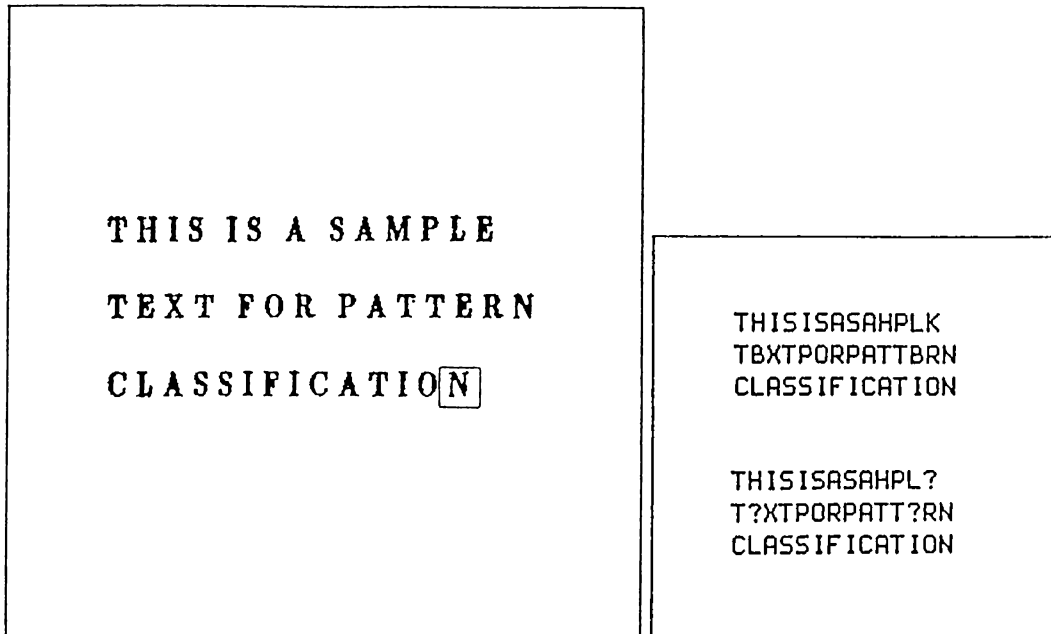


Figure 4.17: Image3 (left), the classification results with PREP2 for  $Th = 0$  (right top), and for  $Th = 2$  (right bottom).

performance for the same images applied to a network trained on letters of *Roman* and *Monaco* font. Table 4.2 summarizes the classification performance of the system with PREP1 and PREP2. The first two columns are results for the network trained on letters of *Roman* font, which is the network used up to this point. Last two columns are results for a network trained on letters of both *Roman* and *Monaco* font. For the *Roman* font network, the average success ratio for PREP1 is 70%, while for PREP2 is 68%. However, for the *Roman* and *Monaco* network the average success ratio for PREP1 rises to 90%, and for PREP2 to 89%. Note that, initially the network was trained on letters of Roman font and these texts of seven different fonts are completely new to the network. Also note that, the 20% increase in the overall success ratio is achieved by simply adding the letters of worst classified font into the training set.

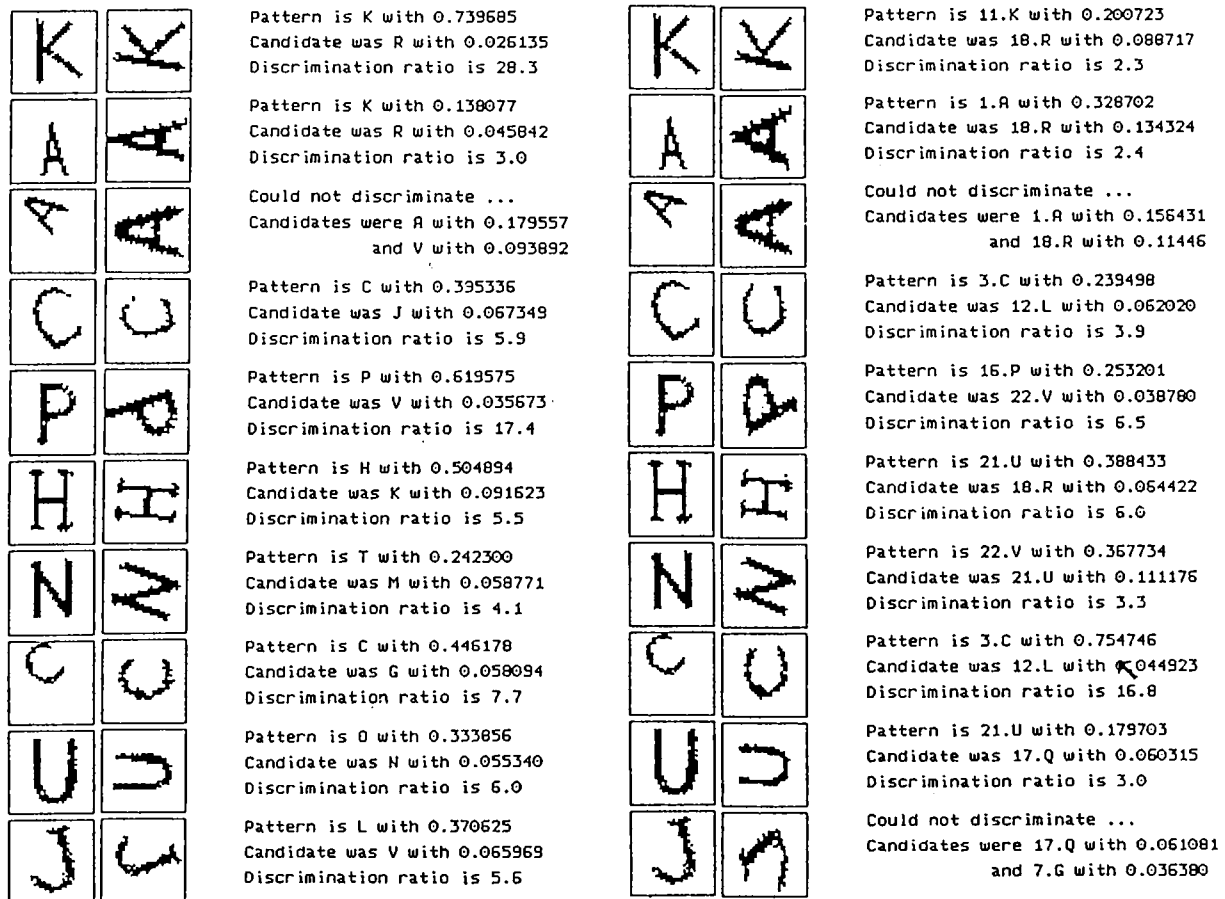


Figure 4.18: Classification results, with PREP1 (left) and PREP2 (right), on 10 handcrafted English letters.

Font		PREP1	PREP2
N.C.S.	ABCDEFGHIJ KLMNOPQ RSTUVWXY[Z]	VBCDEFDHIP KLWNO PQ RSTUVWXYZ	ABCDEFUHC KLKNOPQ RSTUVXYZ
Bookman	ABCDEFGH IJKLMNOP QRSTUVWXY[Z]	VBCDBFGH IJKLWNO P QRSTUVW XFZ	PBCDEFGH IJKLUNOP DRSTUVYXFZ
Boston	ABCDEFGHI JKLMNOPQRST UVWXY[Z]	BBCDEFQBI PELMHOPQRB T BBW XFZ	RBCCEFGHI PKLNNOPQRST DLAKPZ
Courier	ABCDEFGHIJ KLMNOPQRST UVWXY[Z]	RBCDBLGBXC KLWHOPPRBT BVW XEZ	PBCDKFGRLC KLRHOPPRST CPRXXK
Helvetica	ABCDEFGHI JKLMNOPQR STUVWXY[Z]	VBCGELOHI LKJMNOPB STDVW XFZ	PBGDEPGR I PKLHNO PQR STUIAXAL
Monaco	ABCDEFGH IJKLMNOP QRSTUVWXY[Z]	RBCDBFDB IJKTNHOP QRSTBRKXJZ	RDCCEPGH YCKJINDP QRSTUPFKLZ
Times	ABCDEFGHIJ KLMNOPQRST UVWXY[Z]	ABCOEIOHII KLMNOPQRST UVWXYZ	ABCDRPURIJ KLUNOPQRST UVWXYZ

Figure 4.19: Classification results for English text of seven different fonts: New Century Schoolbook, Bookman, Boston, Courier, Helvetica, Monaco, and Times.



Font		PREP1	PREP2
N.C.S.	<p>ABCDEFGHIJ            KLMNOPQ            RSTUVWXY[Z]</p>	<p>VBDDEFUHIJ            KLWNO PQ            RSTUW XYZ</p>	<p>VBJDKFGHIJ            KLMNO PQ            RSTJW XYZ</p>
Bookman	<p>ABCDEFGHIH            IJKLMNOP            QRSTUVWXY[Z]</p>	<p>VBDDEPGH            IJKLMNOP            URSTUW XVZ</p>	<p>VBCDEFGH            IJKLMNOP            BRSTUVJ XYZ</p>
Boston	<p>A B C D E F G H I            J K L M N O P Q R S T            U V W X Y [Z]</p>	<p>KBCDEFGHI            JKLANOPQRST            UVWXYZ</p>	<p>ABCDEFGHI            JKLMNOPQRST            UVMXYZ</p>
Courier	<p>A B C D E F G H I J            K L M N O P Q R S T            U V W X Y [Z]</p>	<p>ABCDEFGHIJ            KLMNOPQRST            UVW XEZ</p>	<p>ABCDELGKIJ            KLWNO PQRST            UVWXYZ</p>
Helvetica	<p>ABCDEFGHI            JKLMNOPQR            STUVWXY[Z]</p>	<p>ABCDEFGHI            JKLMNOPOU            STUVW XFZ</p>	<p>ASCUELGHT            JMLMNOPOR            STUVW XYZ</p>
Monaco	<p>ABCDEFGHIH            IJKLMNOP            QRSTUVWXY[Z]</p>	<p>ABCDEFGHIH            IJKLMNOP            QRSTUW XYZ</p>	<p>ABCDEFGHIH            IJKLMNOP            QRSTUW XYZ</p>
Times	<p>ABCDEFGHIJ            KLMNOPQRST            UVWXY[Z]</p>	<p>ABCDEFGHIJ            KLWNO PQRST            UVWXYZ</p>	<p>ABCDEPGH I            KLMNO PQRST            UVWXYZ</p>

Figure 4.20: Classification results for English text of seven different fonts using a network trained on letters of both *Roman* and *Monaco* fonts.

Font	Roman font		Roman and Monaco fonts	
	Success ratio for PREP1	Success ratio for PREP2	Success ratio for PREP1	Success ratio for PREP2
N.C.S.	85%	85%	85%	85%
Bookman	85%	81%	81%	88%
Boston	62%	65%	92%	96%
Courier	54%	50%	96%	88%
Helvetica	62%	62%	85%	77%
Monaco	54%	50%	100%	100%
Times	85%	81%	92%	92%
Overall	70%	68%	90%	89%

Table 4.2: Performance results for English text of seven different fonts.

## 4.2 Character Recognition on the Japanese Katakana Alphabet

This problem is the classification of symbols in the Japanese Katakana alphabet. The alphabet is shown in Figure 4.21. Since the 111 Katakana symbols are in fact combined forms of 66 unique patterns, the system is trained only on these patterns. Figure 4.22 shows the example patterns 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. From experimentation, a network with only one hidden layer having twenty nodes is chosen.

Figures 4.23 through 4.27 give the performance on single grid images, where the preprocessor is PREP1. Images are  $32 \times 32$  pixels. 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.

Figure 4.28 gives the performance on a camera scanned raster image, using

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	ニユ	ヒユ	ビユ	ピユ	ミユ	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			

Figure 4.21: The 111 symbols of the Japanese Katakana Alphabet.

ア <sub>1</sub>	カ <sub>6</sub>	ガ <sub>11</sub>	サ <sub>16</sub>	ザ <sub>21</sub>	タ <sub>26</sub>	ダ <sub>31</sub>	ナ <sub>36</sub>	ハ <sub>41</sub>	バ <sub>46</sub>	マ <sub>51</sub>	ラ <sub>56</sub>	ワ <sub>61</sub>		ン <sub>62</sub>	ヤ <sub>64</sub>
a	ka	ga	sa	za	ta	da	na	ha	ba	ma	ra	wa		n	ya
イ <sub>2</sub>	キ <sub>7</sub>	ギ <sub>12</sub>	シ <sub>17</sub>	ジ <sub>22</sub>	チ <sub>27</sub>	ヂ <sub>32</sub>	ニ <sub>37</sub>	ヒ <sub>42</sub>	ビ <sub>47</sub>	ミ <sub>52</sub>	リ <sub>57</sub>				ユ <sub>65</sub>
i	ki	gi	shi	ji	chi	ji	ni	hi	bi	mi	ri				yu
ウ <sub>3</sub>	ク <sub>8</sub>	グ <sub>13</sub>	ス <sub>18</sub>	ズ <sub>23</sub>	ツ <sub>28</sub>	ヅ <sub>33</sub>	ヌ <sub>38</sub>	フ <sub>43</sub>	ブ <sub>48</sub>	ム <sub>53</sub>	ル <sub>58</sub>				ヨ <sub>66</sub>
u	ku	gu	su	zu	tsu	zu	nu	fu	bu	mu	ru				yo
エ <sub>4</sub>	ケ <sub>9</sub>	ゲ <sub>14</sub>	セ <sub>19</sub>	ゼ <sub>24</sub>	テ <sub>29</sub>	デ <sub>34</sub>	ネ <sub>39</sub>	ヘ <sub>44</sub>	ベ <sub>49</sub>	メ <sub>54</sub>	レ <sub>59</sub>				
e	ke	ge	se	ze	te	de	ne	he	be	me	re				
オ <sub>5</sub>	コ <sub>10</sub>	ゴ <sub>15</sub>	ソ <sub>20</sub>	ゾ <sub>25</sub>	ト <sub>30</sub>	ド <sub>35</sub>	ノ <sub>40</sub>	ホ <sub>45</sub>	ボ <sub>50</sub>	モ <sub>55</sub>	ロ <sub>60</sub>			ヲ <sub>63</sub>	
o	ko	go	so	zo	to	do	no	ho	bo	mo	ro			o	

Figure 4.22: The 66 unique patterns and their corresponding class numbers.

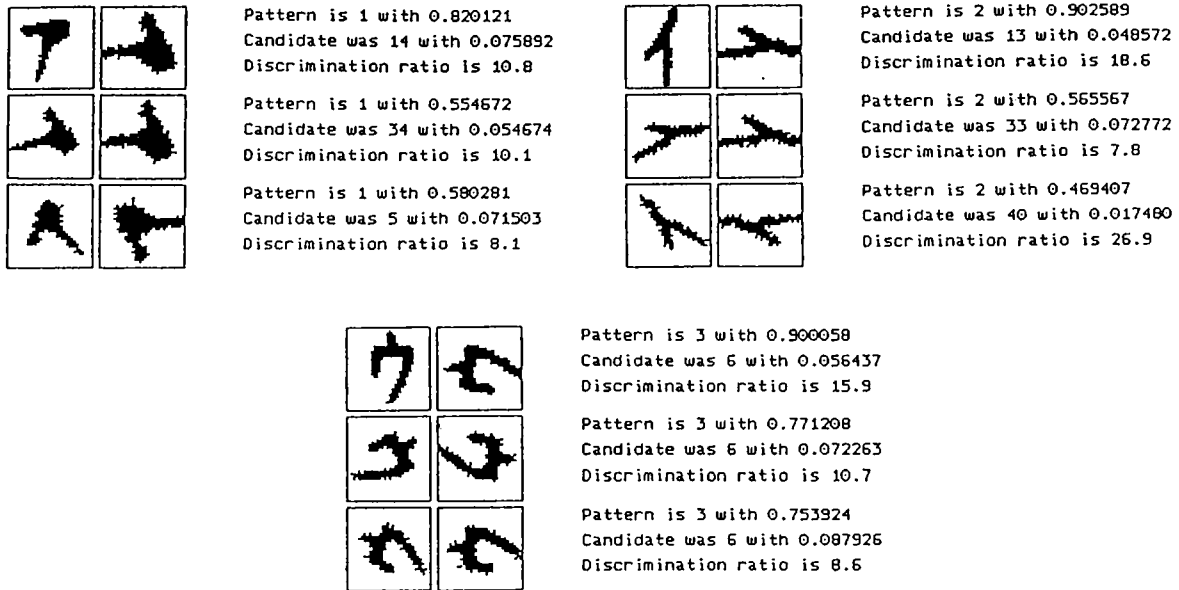


Figure 4.23: Classification results with PREP1 for symbols from Class 1, 2, and 3 rotated by 0, 60, and -60 degrees.

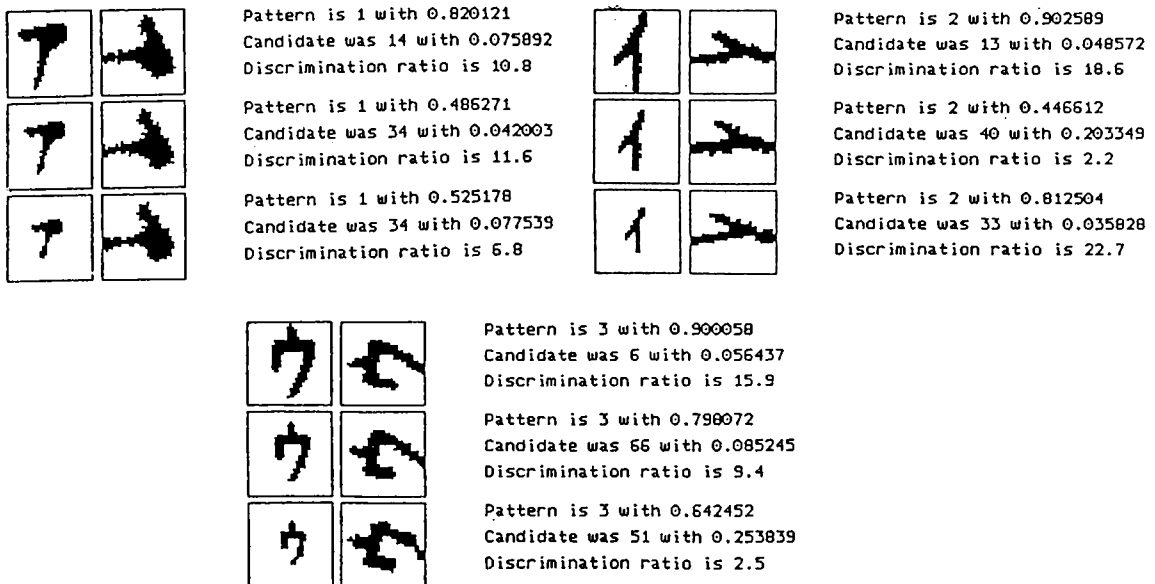


Figure 4.24: Classification results with PREP1 for symbols from Class 1, 2, and 3 scaled by a factor of 1, 0.8, and 0.6.

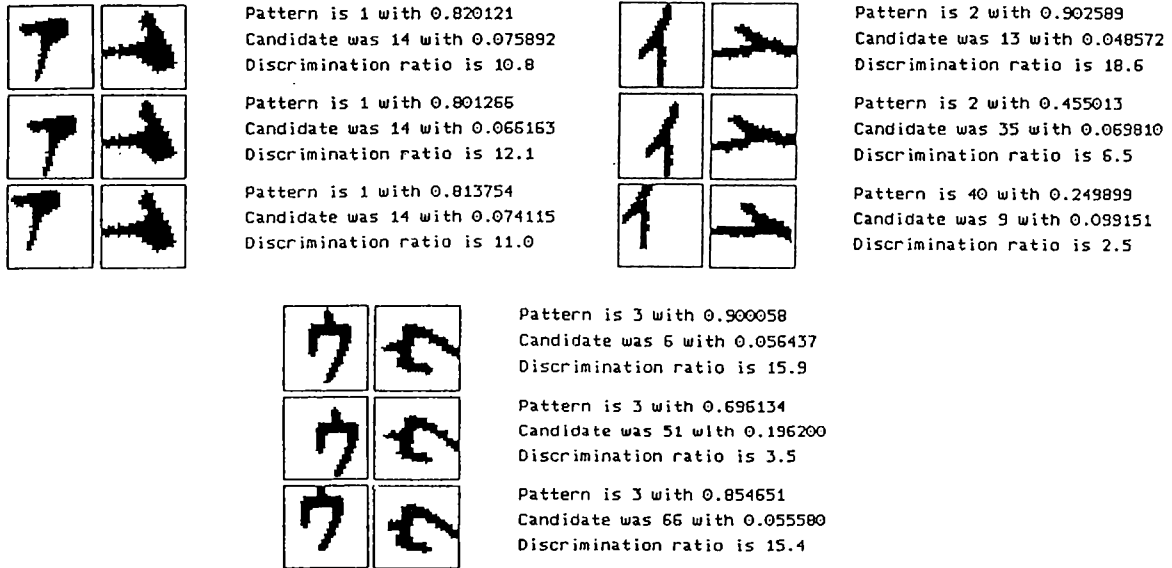


Figure 4.25: Classification results with PREP1 for symbols from Class 1, 2, and 3 translated diagonally by 0, 6, and -6 pixels.

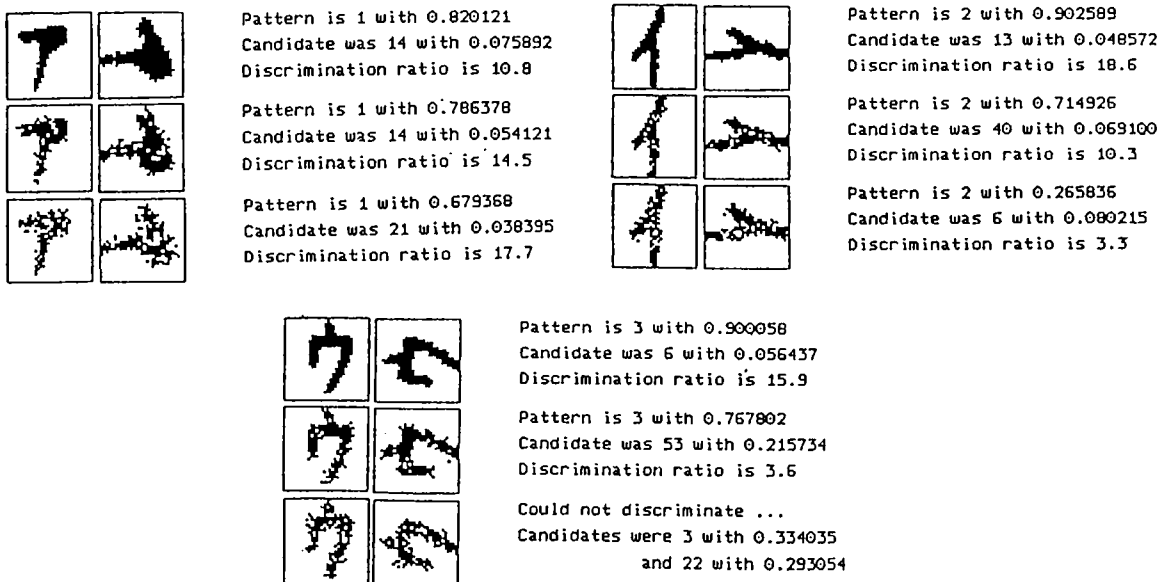


Figure 4.26: Classification results with PREP1 for symbols from Class 1, 2, and 3 with 0%, 20%, and 40% noise.

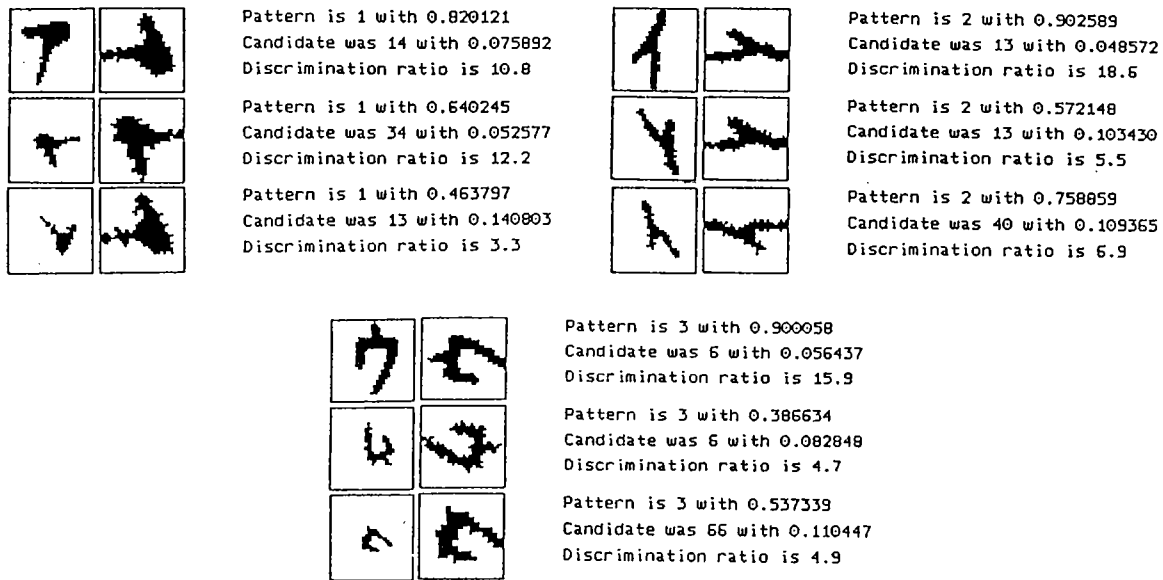


Figure 4.27: Classification results with PREP1 for symbols from Class 1, 2, and 3 with random translation, scaling, and rotation applied.

PREP1. The image is  $512 \times 512$  pixels. Classification is done by positioning the  $32 \times 32$  grid on each pattern. If only correctly classified patterns are considered the success ratio is 40%. However, for most wrong classifications the system has decided on a pattern geometricly similar to the correct pattern. Note that, in the Japanese Katakana alphabet, there are groups of symbols that are broad versions of an original pattern. Only small differences differentiate such patterns. Hence, the success ratio rises up to 82%, if classifying geometricly similar patterns is accepted. This high percentage points out that if more detailed training is applied for the Japanese Katakana symbols the success ratio may rise up to values of 60% - 70%.

### 4.3 Classification of Geometric Symbols

The problem is the classification of five main geometric symbols; circular, cross, line, square-like, and triangular patterns. The original patterns from each class

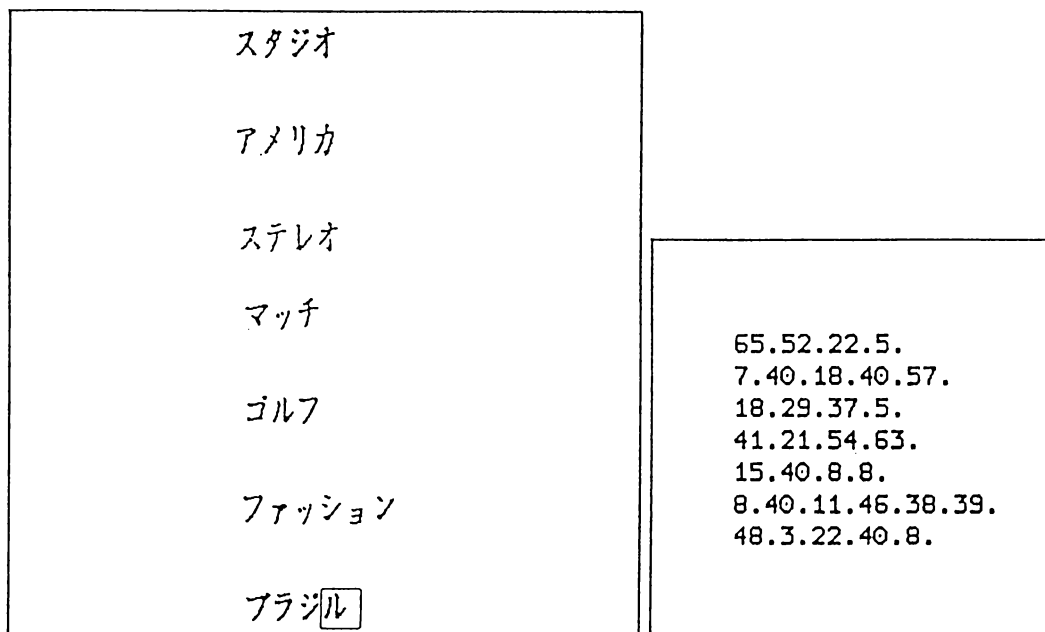


Figure 4.28: Image4, a  $512 \times 512$  pixel image of Japanese text (left) and the classification result with PREP1 (right).

is given in Figure 4.29.

Figures Figure 4.30 through Figure 4.34 gives the performance on single grid images of the geometric symbols, using PREP1. Images are  $32 \times 32$  pixels. 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 followed by its corresponding output neuron's value are given.

○	×	-	1 2 3
□	△		4 5

Figure 4.29: The patterns and class numbers for the five main geometric symbols; circular, cross, line, square-like, and triangular.

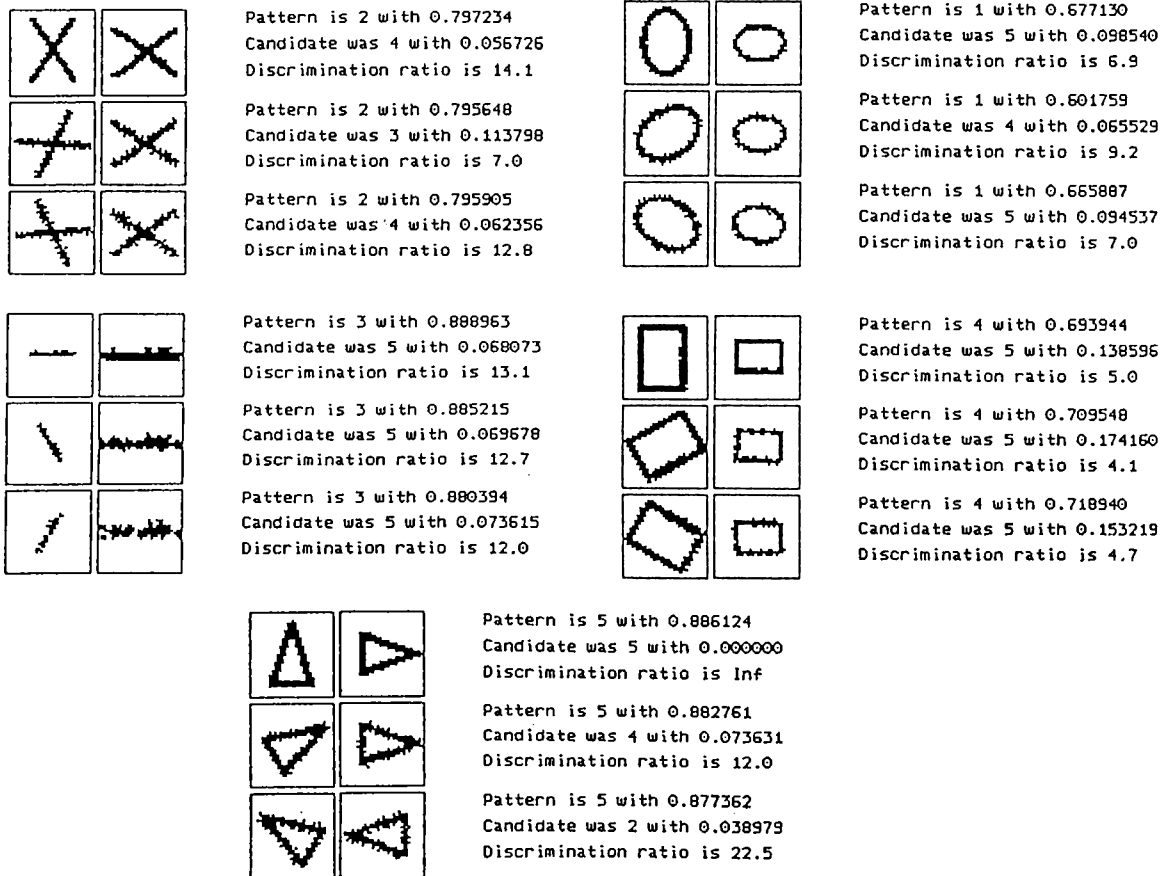


Figure 4.30: Classification results with PREP1 for geometric symbols rotated by 0, 60, and -60 degrees.



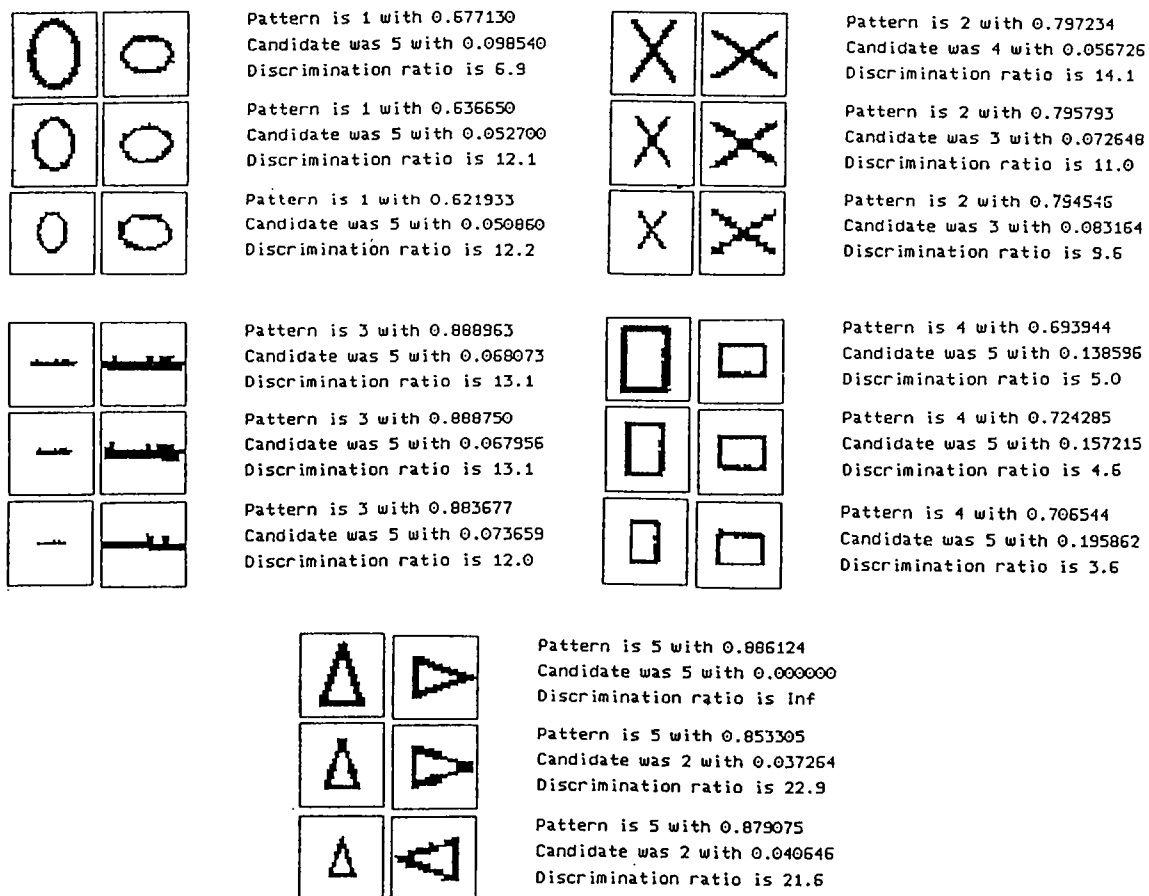


Figure 4.31: Classification results with PREP1 for geometric symbols scaled by a factor of 1, 0.8, and 0.6.

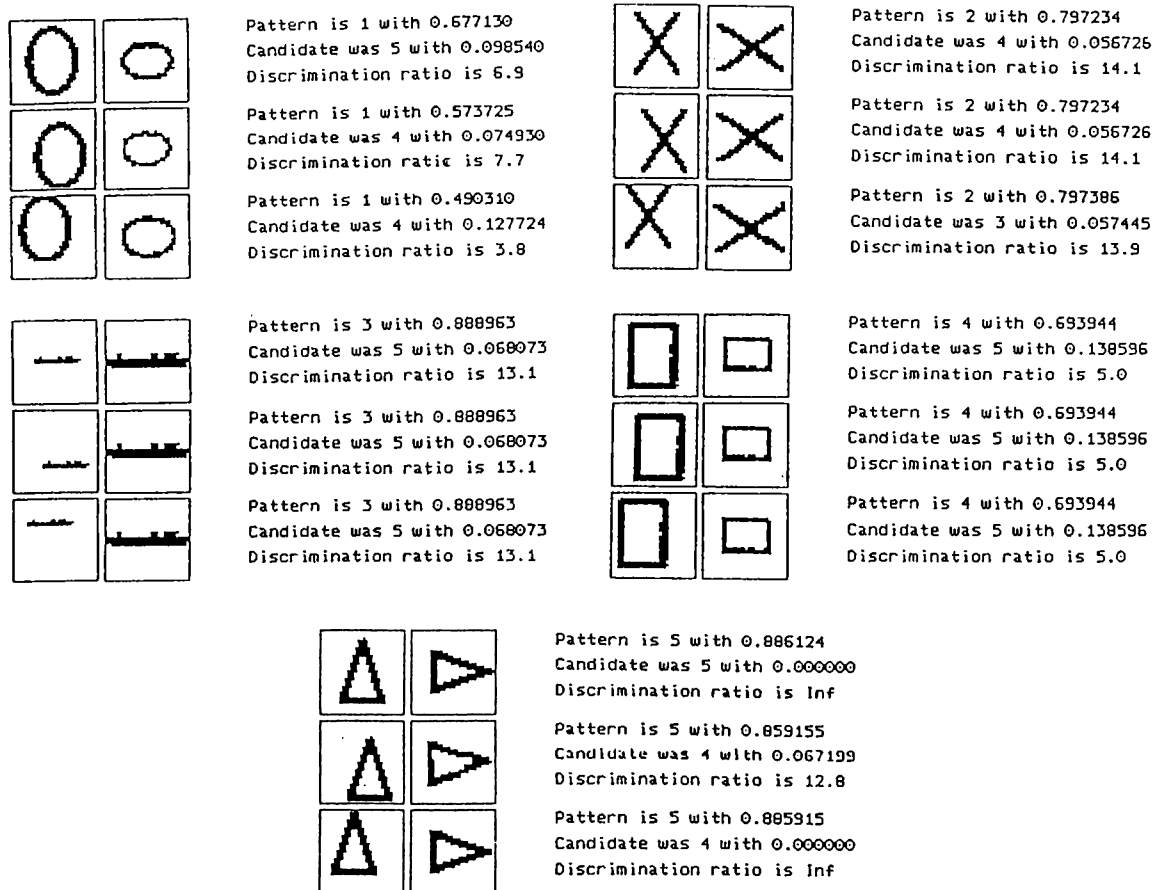


Figure 4.32: Classification results with PREP1 for geometric symbols translated diagonally by 0, 6, and -6 pixels.

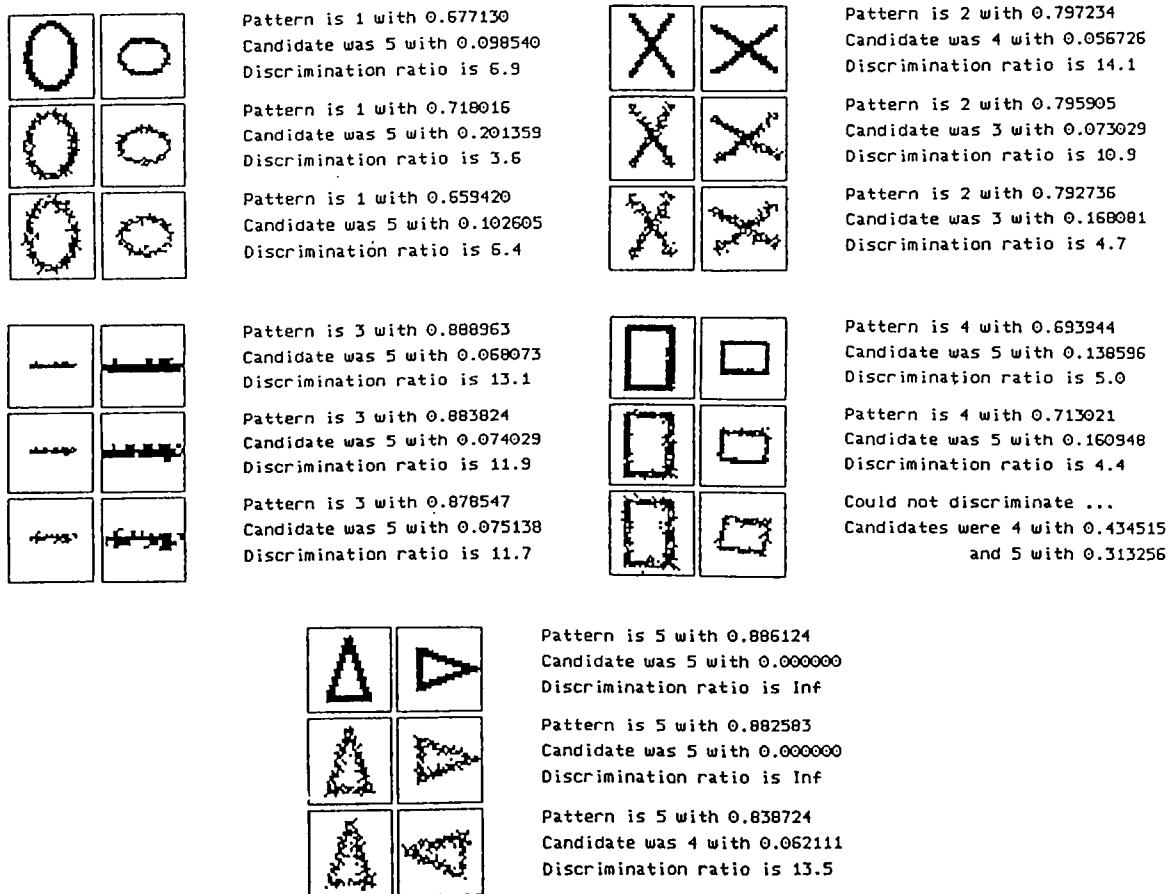


Figure 4.33: Classification results with PREP1 for geometric symbols with 0%, 20%, and 40% noise.

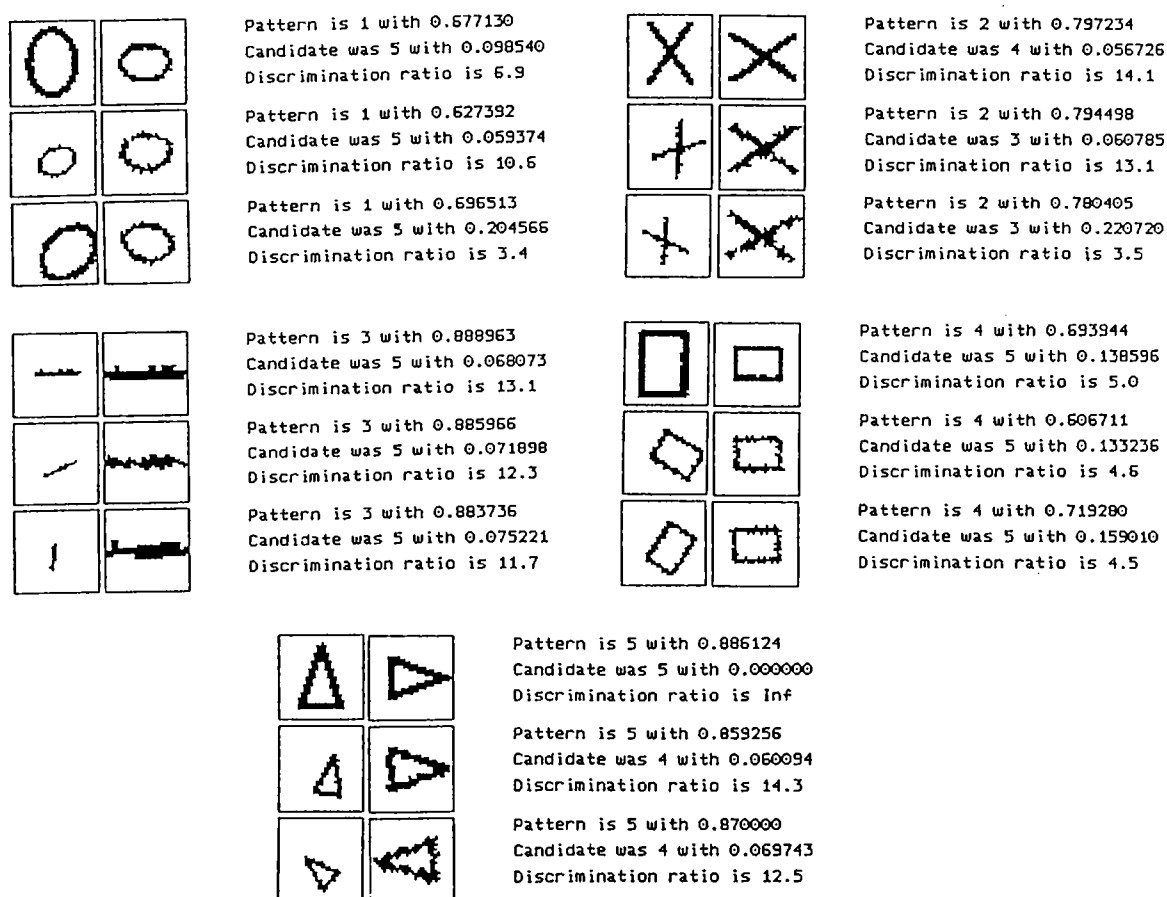


Figure 4.34: Classification results with PREP1 for geometric symbols with random translation, scaling, and rotation applied.

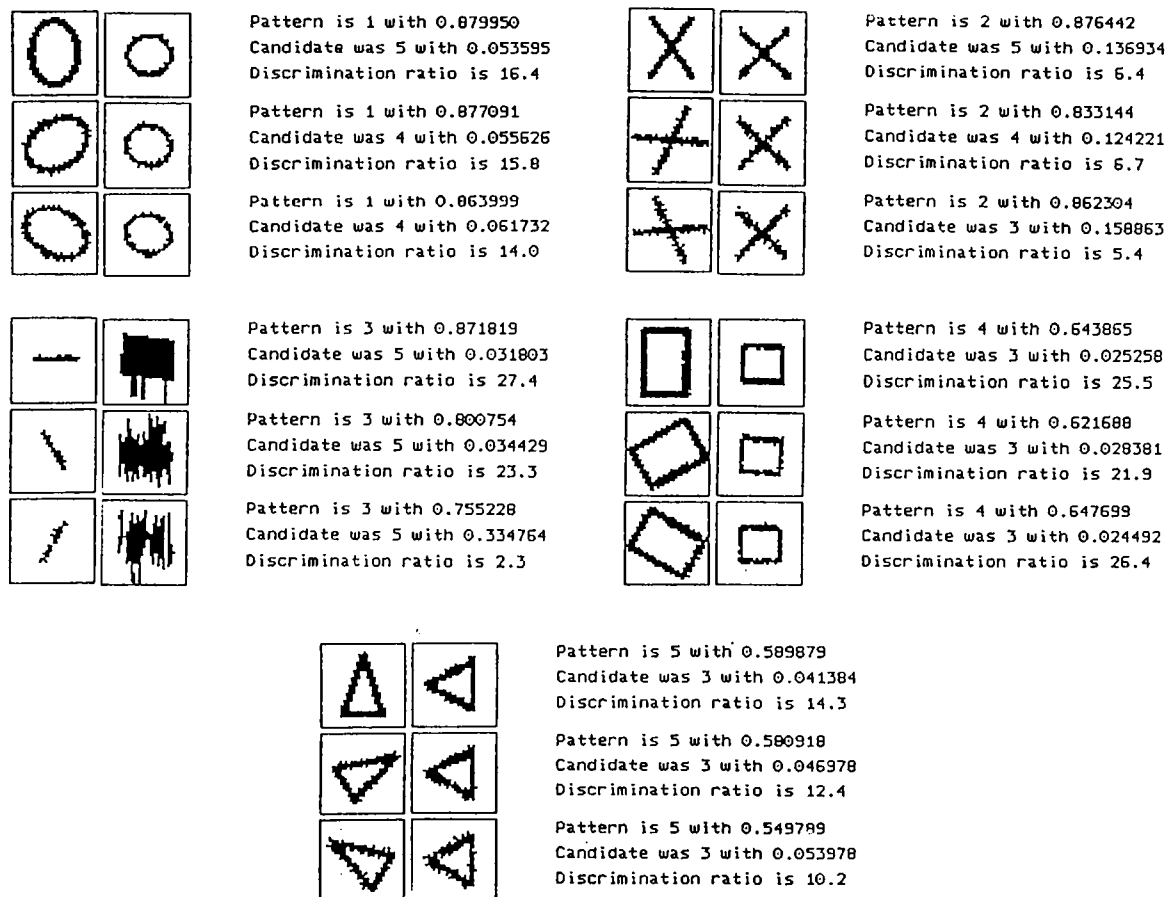


Figure 4.35: Classification results with PREP2 for rotated geometric symbols.

Figure 4.35 through Figure 4.39 give the performance on single grid  $32 \times 32$  pixel images, using PREP2.

In order to test concurrent classification power of the system, patterns formed by merging two of the geometric symbols are used. Concurrent classification is the ability to recognize all of the patterns if multiple patterns exist in a single image. Figure 4.40 gives the classification results for the two versions of the preprocessor on the mentioned patterns of merged symbols. For the preprocessor with radial scaling, PREP1, in three out of ten patterns, the system catches both of the symbols. In six of the remaining seven patterns, the system successfully classifies one of the symbols. In the remaining pattern not the decision but the second candidate is one of the symbols. For the preprocessor with axial scaling, PREP2,

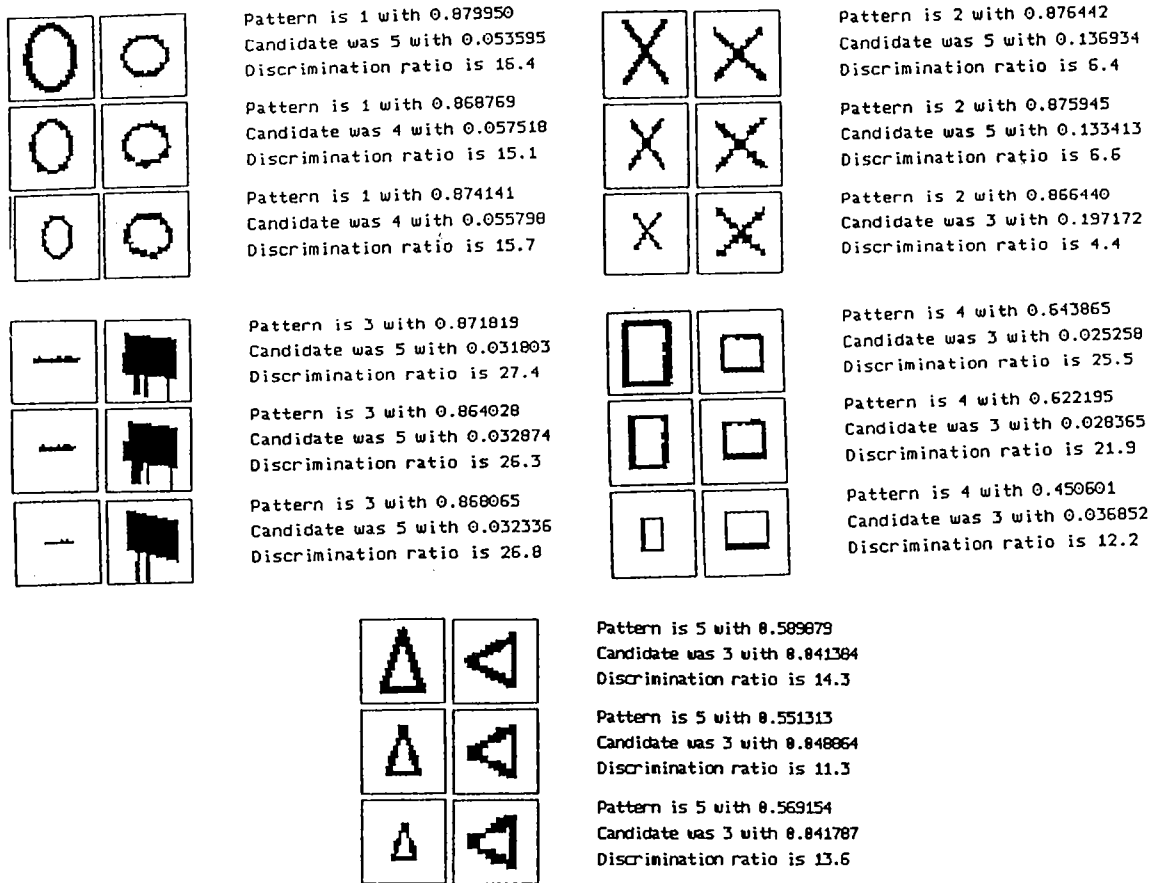


Figure 4.36: Classification results with PREP2 for scaled geometric symbols.

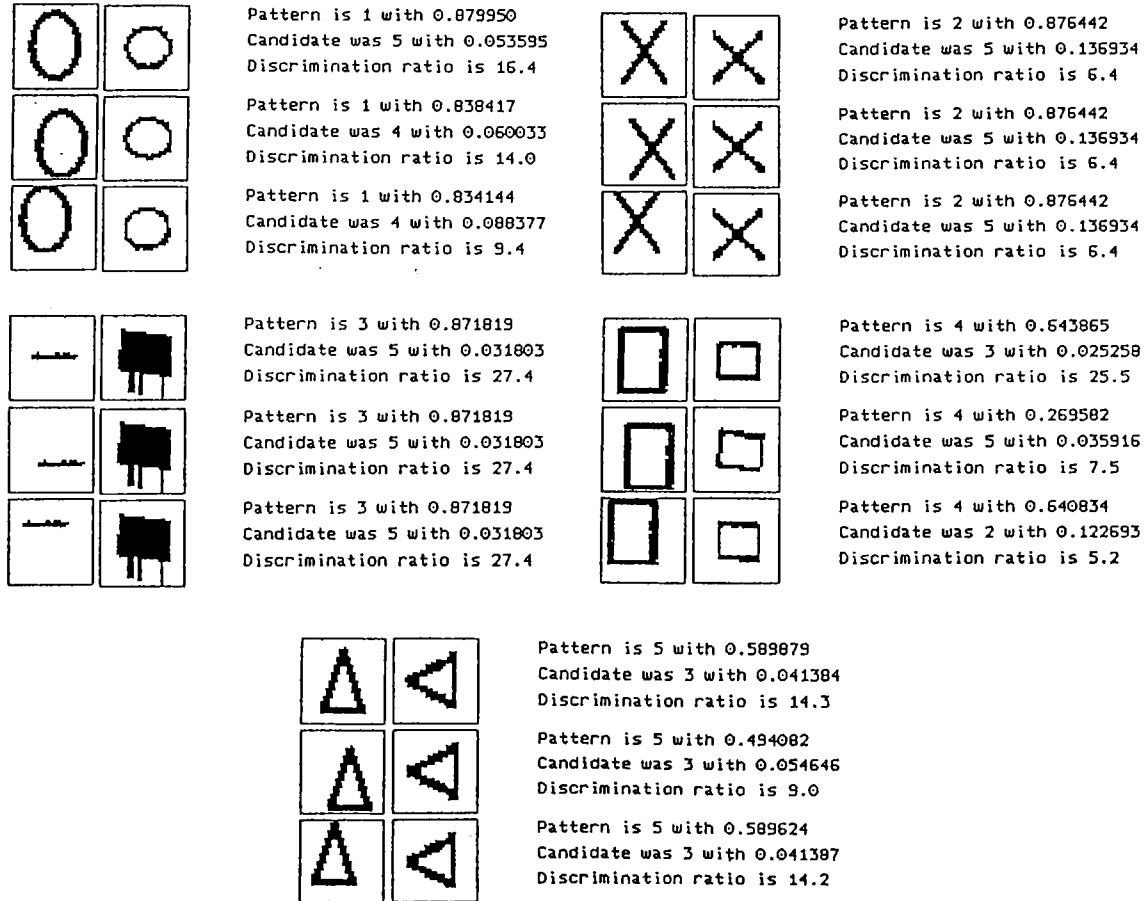


Figure 4.37: Classification results with PREP2 for translated geometric symbols.

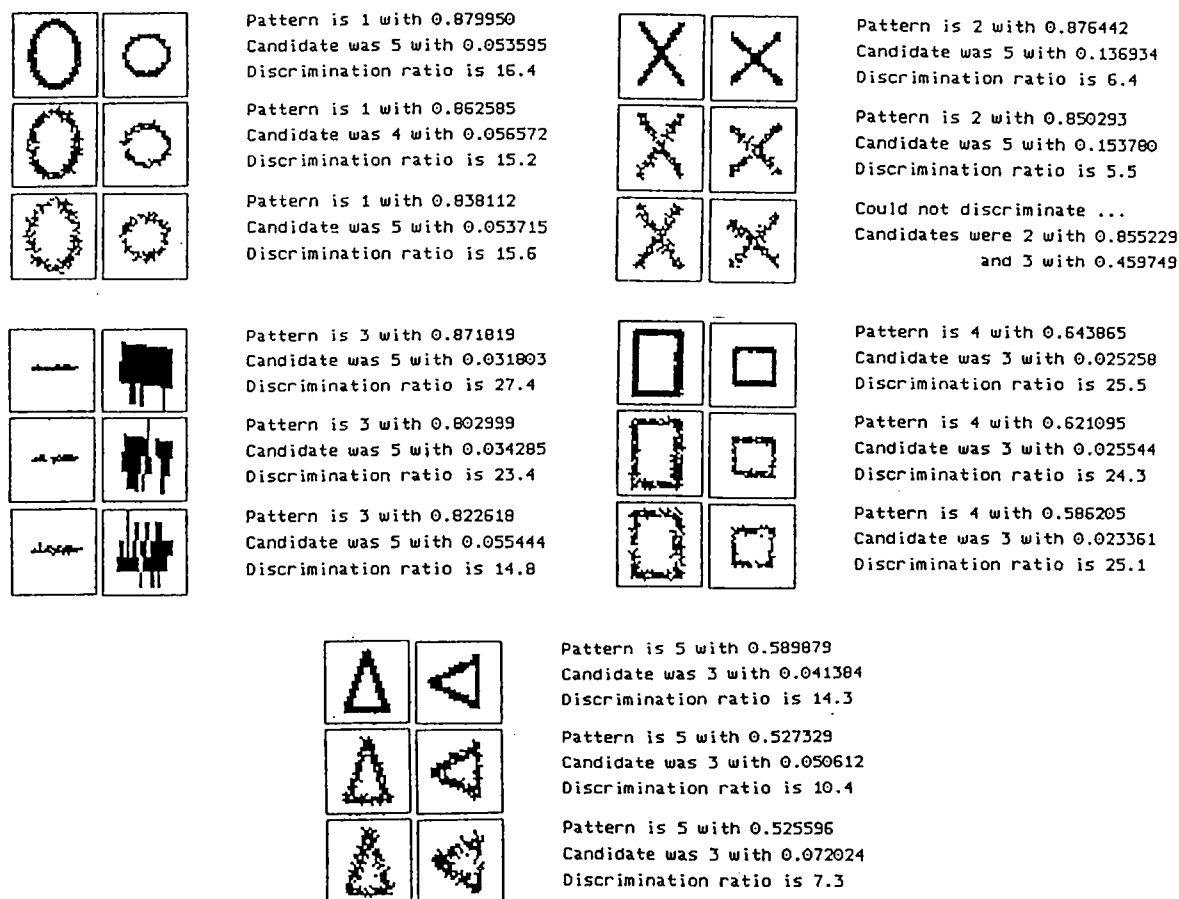


Figure 4.38: Classification results with PREP2 for noisy geometric symbols.



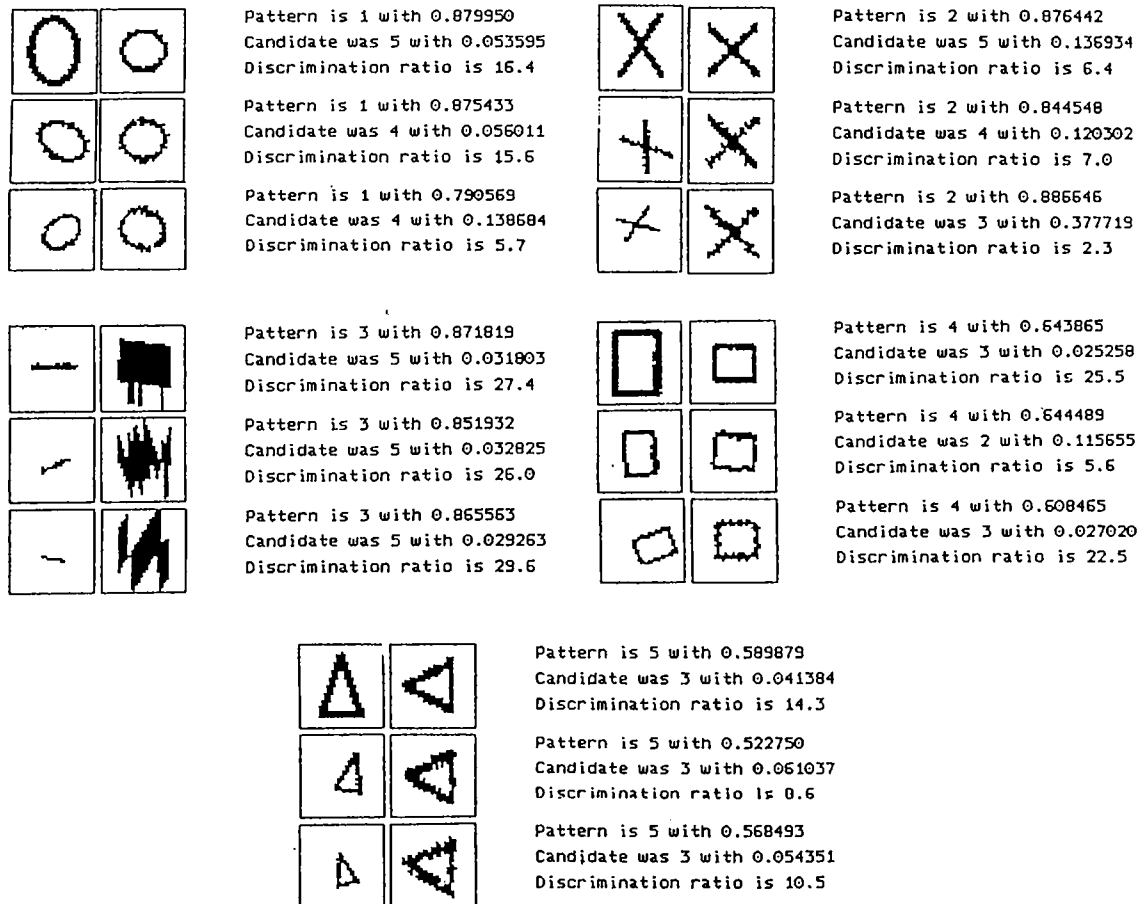


Figure 4.39: Classification results with PREP2 for geometric symbols with random translation, scaling, and rotation applied.

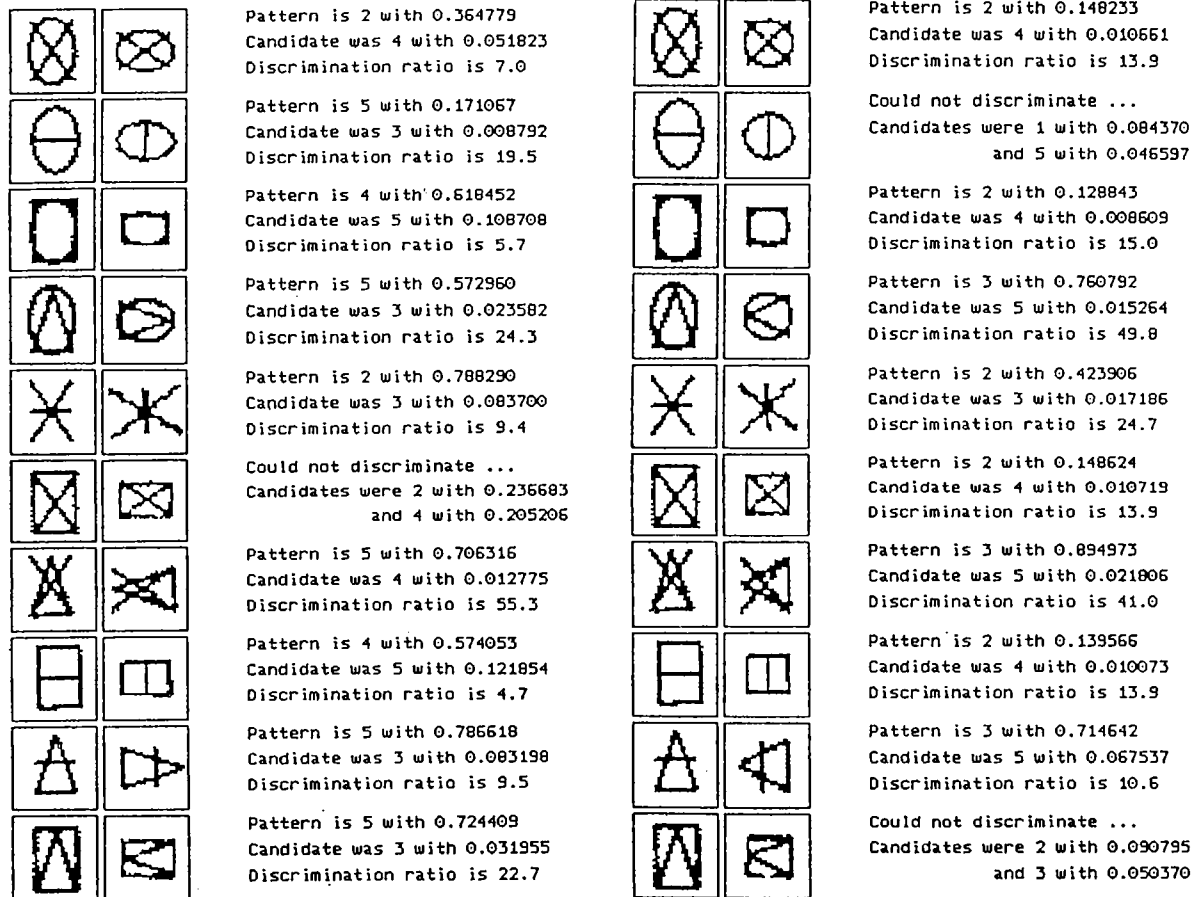


Figure 4.40: Classification results, with PREP1 (left) and PREP2 (right), on patterns formed by merging two geometric symbols.

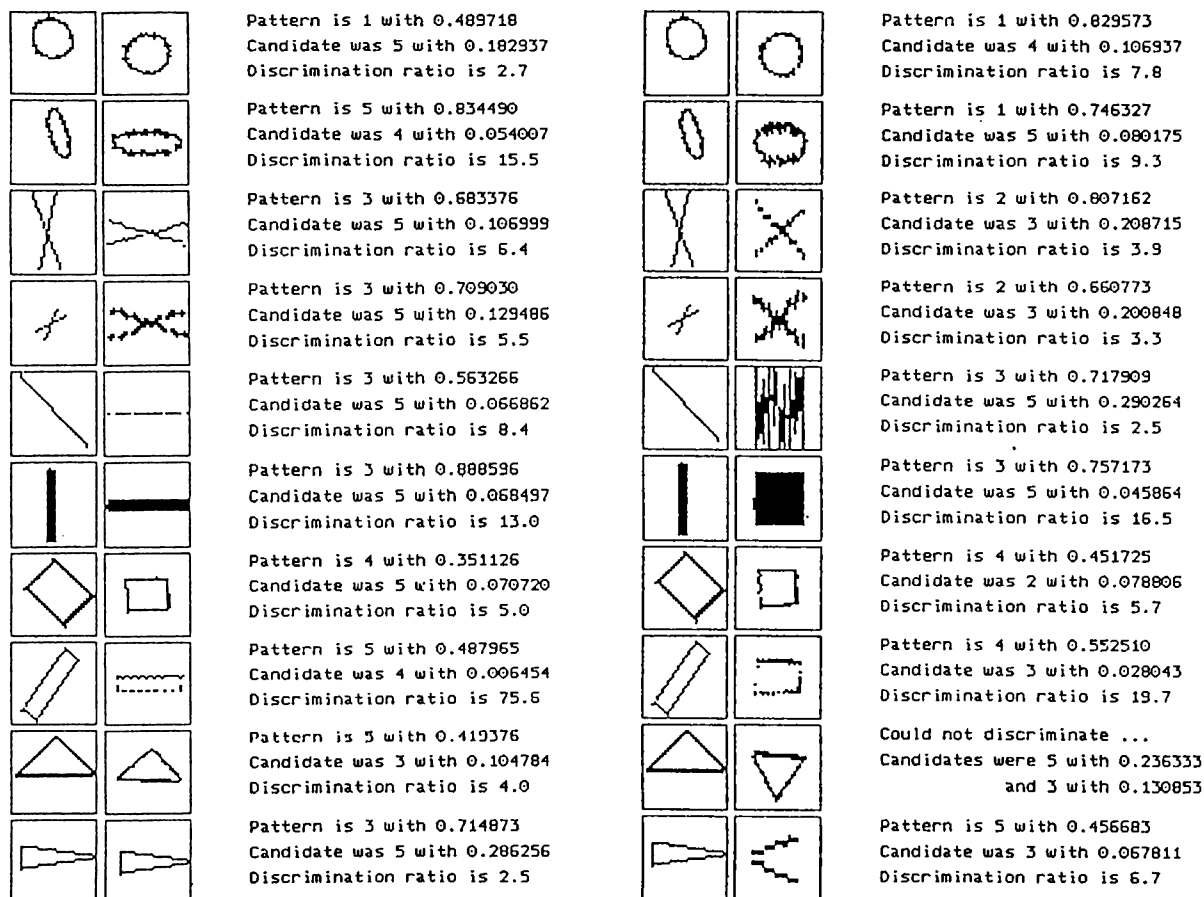


Figure 4.41: Classification results, with PREP1 (left) and PREP2 (right), on distorted patterns of the five main geometric symbols.

in three out of ten patterns, the system catches both of the symbols. In two of the remaining seven patterns, the system correctly classifies one of the symbols. In four of the remaining five patterns not the decision but the second candidate is one of the symbols. The remaining single pattern is completely misclassified.

Figure 4.41 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.

# Chapter 5

## Conclusions

Artificial neural networks have recently been used for pattern classification purposes. Although previous works had successful results there was a need for a general purpose pattern classification system which is rotation, scaling, and translation invariant. The invariances are sought to deal with real life applications, and generality to be able to apply the system without any modifications on any classification problem.

In this work we have proposed a pattern recognition system which is rotation, scaling, and translation invariant. In Chapter 3, a detailed definition of the system is given followed by the formulation. Two preprocessors, one with radial scaling correction and other with axial scaling correction is introduced. The preprocessors are responsible of maintaining the rotational, scaling, and translational invariances. Rotational invariance is maintained by a subblock named R-Block, embedding a Karhunen-Loève transformation based mapping function. In Chapter 4, through experimentation on the English alphabet, the Japanese Katakana alphabet, and some geometric patterns the system's power in maintaining the invariances has been shown. Unless a pattern loses its original structure by extreme translations or scaling, the preprocessor normalizes it into one of the

canonical patterns which are rotation, scaling, and translation invariant. The generalizing property of the system emerges from the artificial neural network employed in the classifier block. The network is trained on the preprocessed versions of the example patterns from each class using the supervised learning algorithm *backpropagation*. The final weights are used thereon to perform classifications on patterns of unknown class. The activation values of the output nodes are then used by the interpreter. The ratio of the maximum output to the next highest output should exceed a predetermined threshold. If true, then the class represented by the neuron having the maximum output is reported. If false, then the interpreter reports *no discrimination*.

Experiments with the mentioned classification problems showed that the artificial neural network performed successfully although it was formed of simple and basic functioning units. The average percentage of correct classification, or the average success ratio, for three  $512 \times 512$  pixel images of English text is 93% which is an *acceptable* value considering the system is a general purpose pattern classifier – not a special purpose optical character recognition system –. During a forward pass 42000 floating point operations are performed. On Sun4 Sparc workstations, the average classification speed of the system is 7 patterns per second. The system is also been tested on English text of previously unseen seven new fonts, which are New Century Schoolbook, Bookman, Boston, Courier, Helvetica, Monaco, and Times. Two networks have been used, first is the network trained on letters of *Roman* font – the network used up to this point – and second a network trained on letters of both *Roman* and *Monaco* fonts. For the *Roman* network the average success ratio with the preprocessor with radial scaling correction, PREP1, is 70% while with the preprocessor with axial scaling correction, PREP2, is 68%. However for the *Roman* and *Monaco* network the average success ratio for PREP1 rises to 90%, and for PREP2 to 89%. Note that, initially the neural network was trained on letters of *Roman* font and these texts of seven different fonts are completely new for the network. For the Japanese Katakana alphabet the correct success ratio for a  $512 \times 512$  pixel image of Japanese text

is only 40%. Careful analysis yields that the system had decided on patterns that are geometricly similar to the correct pattern for most of the wrong decision cases. Note that, several groups of Japanese Katakana symbols are in fact broad versions of an original symbol, and only small differences differentiate such patterns. Hence, the success ratio rises up to 82% if classifying geometricly similar patterns is accepted. This high percentage points out that if more detailed training is applied for the Japanese Katakana symbols the success ratio may rise up to values of 60% - 70%. Finally, for the geometric symbols two tests have been performed. First, on patterns formed by merging two geometric symbols where concurrent classification power of the system was tested. Concurrent classification is the ability to recognize all of the patterns if multiple patterns exist in a single image. The system has managed to capture both of the symbols in the image by 30%, and managed to detect at least one of the symbols by 70%. Second test has been performed for distorted versions of the geometric symbols. The preprocessor with radial scaling correction, PREP1, could detect only 50% of the distorted patterns, while the preprocessor with axial scaling correction, PREP2, managed to successfully classify 90%. The performance difference emerges from the axial scaling correction ability of this preprocessor.

# Appendix A

## Derivation of the R-Block Functions

Define,

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

$$P = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \quad (\text{A.2})$$

$$T_{xx} = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \cdot x_i^2 \quad T_{yy} = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \cdot y_j^2 \quad (\text{A.3})$$

$$T_{xy} = \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \cdot x_i \cdot y_j \quad (\text{A.4})$$

The covariance matrix defined as:

$$C = \frac{1}{P} \left( \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix} \right) \left( \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix} \right)^T \quad (\text{A.5})$$

can be simplified to:

$$C = \left( \frac{1}{\sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j)} \sum_{i=1}^N \sum_{j=1}^N f_{TS}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix} \begin{bmatrix} x_i \\ y_j \end{bmatrix}^T \right)$$

$$- \begin{bmatrix} m_x \\ m_y \end{bmatrix} \begin{bmatrix} m_x \\ m_y \end{bmatrix}^T \quad (\text{A.6})$$

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:

$$C = \begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix} \quad (\text{A.7})$$

The eigenvalues of this matrix are computed from:

$$(\lambda I - C) = \begin{bmatrix} (\lambda - T_{xx}) & -T_{xy} \\ -T_{xy} & (\lambda - T_{yy}) \end{bmatrix} \quad (\text{A.8})$$

$$(\lambda - T_{xx})(\lambda - T_{yy}) - T_{xy}^2 = 0 \quad (\text{A.9})$$

$$\lambda_{1,2} = \frac{T_{yy} + T_{xx} \pm \sqrt{(T_{yy} + T_{xx})^2 - 4(T_{xx} \cdot T_{yy} - T_{xy}^2)}}{2} \quad (\text{A.10})$$

then the corresponding eigenvectors can be derived as:

$$\begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix} \quad (\text{A.11})$$

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

$$T_{xy} \cdot x + T_{yy} \cdot y = \lambda \cdot y \quad (\text{A.12})$$

$$\frac{y}{x} = \frac{\lambda - T_{xx}}{T_{xy}} \quad (\text{A.13})$$

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

$$\frac{y}{x} = \frac{T_{yy} - T_{xx} + \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}{2 \cdot T_{xy}} \quad (\text{A.14})$$

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

$$\sin \theta = \frac{y}{\sqrt{x^2 + y^2}} \quad (\text{A.15})$$



$$= \frac{(T_{yy} - T_{xx}) + \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}{\sqrt{2} \cdot \left[ \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx}) \right] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}} \quad (\text{A.16})$$

$$\cos \theta = \frac{x}{\sqrt{x^2 + y^2}} \quad (\text{A.17})$$

$$= \frac{2 \cdot T_{xy}}{\sqrt{2} \cdot \left[ \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx}) \right] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}} \quad (\text{A.18})$$

The mapping function of R-Block is:

$$f_{TSR}(x_i, y_j) = f_{TS}(\cos \theta \cdot x_i + \sin \theta \cdot y_j, -\sin \theta \cdot x_i + \cos \theta \cdot y_j) \quad (\text{A.19})$$

# Appendix B

## Derivation of the PREP2 Functions

Define:

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

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

$$y_{av} = \frac{\sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot y_j}{P} \quad (\text{B.3})$$

Recalling from Equation 3.10, the formula for  $T_{xx}$  which was:

$$T_{xx} = \sum_{i=1}^N \sum_{j=1}^N f_T(x_i, y_j) \cdot x_i^2 \quad (\text{B.4})$$

can be written as:

$$T_{xx} = \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) (x_i - x_{av})^2 \quad (\text{B.5})$$

$$= \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i^2 - 2 \cdot x_{av} \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i + x_{av}^2 \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \quad (\text{B.6})$$

$$= \left( \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i^2 \right) - P \cdot x_{av}^2 \quad (\text{B.7})$$

Similarly:

$$T_{yy} = \left( \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot y_j^2 \right) - P \cdot y_{av}^2 \quad (\text{B.8})$$

$$T_{xy} = \left( \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) \cdot x_i \cdot y_j \right) - P \cdot x_{av} \cdot y_{av} \quad (\text{B.9})$$

The *sine* and *cosine* of the rotation angle remain unchanged:

$$\sin \theta = \frac{(T_{yy} - T_{xx}) + \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}{\sqrt{2 \cdot [\sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx})] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}} \quad (\text{B.10})$$

$$\cos \theta = \frac{2 \cdot T_{xy}}{\sqrt{2 \cdot [\sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2} + (T_{yy} - T_{xx})] \sqrt{(T_{yy} - T_{xx})^2 + 4 \cdot T_{xy}^2}}} \quad (\text{B.11})$$

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

$$s_x = \sqrt{\frac{R_x \cdot P}{\sum_{i=1}^N \sum_{j=1}^N f_{TR}(x_i, y_j) \cdot x_i^2}} \quad (\text{B.12})$$

$$s_y = \sqrt{\frac{R_y \cdot P}{\sum_{i=1}^N \sum_{j=1}^N f_{TR}(x_i, y_j) \cdot y_j^2}} \quad (\text{B.13})$$

The denominator term can be written as:

$$\sum_{i=1}^N \sum_{j=1}^N f_{TR}(x_i, y_j) \cdot x_i^2 = \sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) [(x_i - x_{av}) \cos \theta + (y_i - y_{av}) \sin \theta]^2 \quad (\text{B.14})$$

which simplifies into:

$$\sum_{i=1}^N \sum_{j=1}^N f(x_i, y_j) [(x_i^2 - x_{av}^2) \cos^2 \theta + 2(x_i y_j - x_{av} y_{av}) \cos \theta \sin \theta + (y_i^2 - y_{av}^2) \sin^2 \theta] \quad (\text{B.15})$$

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

$$s_x = \sqrt{\frac{R_x \cdot P}{T_{xx}(\cos \theta)^2 + 2 \cdot T_{xy} \cdot \cos \theta \cdot \sin \theta + T_{yy}(\sin \theta)^2}} \quad (\text{B.16})$$

similarly scaling factor  $s_y$  can be derived as:

$$s_y = \sqrt{\frac{R_y \cdot P}{T_{xx}(\sin \theta)^2 - 2 \cdot T_{xy} \cdot \cos \theta \cdot \sin \theta + T_{yy}(\cos \theta)^2}} \quad (\text{B.17})$$

# Bibliography

- [1] E. Barnard and D. Casasent, "Shift invariance and the neocognitron," *Neural Networks*, vol. 3, pp. 403–410, 1990.
- [2] D. Casasent and D. Psaltis, "Position, rotation, and scale invariant correlation," *Applied Optics*, vol. 15, pp. 1795–1799, 1976.
- [3] Y. Le Cun et al., "Handwritten digit recognition: Applications of neural network chips and automatic learning," *IEEE Communications Magazine*, pp. 41–46, November 1989.
- [4] Y. Le Cun et al., "Handwritten zip code recognition with multilayer networks," in *Proceedings of 10th International Conference on Pattern Recognition, Atlantic City, NJ USA*, pp. 35–40. IEEE Computer Society Press, Los Alamitos, 1990.
- [5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. John Wiley and Sons, NY, 1973.
- [6] K. S. Fu, *Syntactic Pattern Recognition and Application*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, NY, 1972.

- [8] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, pp. 826–834, 1983.
- [9] R. C. Gonzales and P. Wintz, *Digital Image Processing*, pp. 122–130. Addison-Wesley, MA, 1987.
- [10] Y. N. Hsu, H. H. Arsenault, and G. April, "Rotational invariant digital pattern recognition using circular harmonic expansion," *Applied Optics*, vol. 21, pp. 4012–4015, 1982.
- [11] R. L. Kashyap and R. Chellappa, "Stochastic models for closed boundary analysis: Representation and reconstruction," *IEEE Transactions on Information Theory*, vol. IT-27, pp. 627–637, September 1981.
- [12] A. Khotanzad and J. Lu, "Classification of invariant image representations using a neural network," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 1028–1038, June 1990.
- [13] M. Kirby and L. Sirovich, "Application of the karhunen-loève procedures for the characterization of human faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 103–108, January 1990.
- [14] S. Kollias, A. Tirakis, and T. Millios, "An efficient approach to invariant recognition of images using higher-order neural networks," in T. Kohonen, K. Makisara, O. Simula, and J. Kangas, eds., *Artificial Neural Networks*, pp. 87–92. Elsevier Science Publishers B.V., North-Holland, 1991.
- [15] A. Kryzak, S. Y. Leung, and C. Y. Suen, "Reconstruction of two-dimensional patterns by fourier descriptors," in *Proceedings of 9th ICPR*, pp. 555–558, Rome, Italy, November 1988.
- [16] H. A. Malki and A. Moghaddamjoo, "Using the karhunen-loève transformation in the back-propagation training algorithm," *IEEE Transactions on Neural Networks*, vol. 2, pp. 162–165, January 1991.

- [17] G. L. Martin and J. A. Pittman, "Recognizing hand-printed letters and digits using backpropagation learning," *Neural Computation*, vol. , pp. 258–267, 1991.
- [18] K. Oflazer and D. Ercoşkun, "Implementation of backpropagation learning algorithm on a hypercube parallel processor system," in *Proceedings of IS-CIS V, The Fifth International Symposium on Computer and Information Science, Türkiye*, 1990.
- [19] E. Persoon and K. S. Fu, "Shape discrimination using fourier descriptors," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, pp. 170–179, March 1977.
- [20] A. P. Reeves, R. J. Prokop, S. E. Andrews, and F. Kuhl, "Three-dimensional shape analysis using moments and fourier descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 937–943, November 1988.
- [21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing*, volume 1, pp. 318–362. MIT Press, Bradford Books, Cambridge MA, 1986.
- [22] C. Yüceer and K. Oflazer, "A rotation, scaling, and translation invariant pattern classification system," in Mehmet Baray and Bülent Özgüç, eds., *Proceedings of IS-CIS VI, The Sixth International Symposium on Computer and Information Science, Side, Türkiye*, pp. 859–869, 1991.
- [23] C. T. Zhan and C. T. Roskies, "Fourier descriptors for plane closed curves," *IEEE Transactions on Computers*, vol. C-21, pp. 269–281, March 1972.