Fall 1989

Optical Machine Recognition of Lower-Case Greek Characters of Any Size

Ivan X. D. D'Cunha
Old Dominion University

OPTICAL MACHINE RECOGNITION OF LOWER-CASE GREEK CHARACTERS OF ANY SIZE

by

Ivan X. D. D'Cunha
B.E. June 1987, Osmania University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
November, 1989

Approved by :

Nicolas Alvertos (Director)

S. A. Zahorian

S. M. Park

ABSTRACT

OPTICAL MACHINE RECOGNITION OF LOWER-CASE GREEK CHARACTERS OF ANY SIZE

Ivan X. D. D'Cunha

Old Dominion University, 1989

Director: Dr. Nicolas Alvertos

An algorithm utilizing a syntactic approach and a criterion based on normalized moments is defined for the reliable, automatic, machine recognition of handwritten and printed Greek characters of any size and font. In this approach a binary image of the character in question is obtained initially; its skeleton is then produced by utilizing a standard thinning algorithm. The classification process then incorporates the topological features of the characters such as existence of closed curves, number of intersections, number and location of free ends, axial symmetry, and the criteria derived from normalized moments to uniquely identify each pattern. Experiments conducted demonstrated recognition rate of 100% with this approach. An alternate approach involving a mathematical (geometrical) modeling of each character is also proposed.

ACKNOWLEDGEMENTS

At the outset I would like to express my gratitude to Dr. Nicolas Alvertos for his guidance, encouragement and the insight he provided throughout the course of this research. I thank him for the endless hours of helpful and lively discussions and above all the patience which successfully got me through this research.

I would like to thank Dr. S. A. Zahorian and Dr. S. M. Park for being in my committee and reviewing this thesis.

I would also like to thank Gursel Serpen and Jagadeesh Gullapalli for their help whenever needed.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

In the world of industry and commerce, the number of characters that have to be recognized each year is astronomically large. Though exact figures are not available, but it is safe to say that even today only a small percentage of all characters that have to be recognized are in fact recognized by machine [1]. During the last 15 years optical character recognition, a new field then in technology, has come into being and has been nurtured intensively.

Devices now exist for reading conventional typescript and specially constructed fonts at thousands of character per second with relatively small reject rates. Hand written letters and numbers have been automatically recognized with high accuracy, and carefully handwritten cursive script has been successfully read by machines. As powerful as some of the automatic techniques are, the machine capabilities are yet not extremely flexible; they only begin to match human capability, except for speed. Really potent or deeply sophisticated algorithms for machine recognition of print and script still lies in the future.

The use of automatic magnetic-ink character recognition for sorting bank checks has been well established for many years because the number of checks per day has been far too large for manual sorting. In the U.S and Japan, the colossal volume of mail has driven post offices to use optical character recognition (OCR) equipment for postal sorting. This equipment is cost effective even though it rejects a considerable percentage of mail pieces as unreadable.

Any character-reading device, whether it uses magnetic, optical or any other systems, contains the basic components shown in Figure 1.1. Traditionally, expensive equipment has been required to transport documents physically past a scanner. If characters are read while the document is actually in motion, the mechanical equipment is simpler than in the case where characters are read while the document is stationary [2]. Mechanical document handling has been the most costly part of character recognition equipment. Recently, hand-held "wand" scanners have been developed. A human operator simply wipes the pen-like reading head along a line read [3]. This is slower than the fully automatic methods for transporting documents past a reading head, but much less expensive.

As mentioned before, there are four different aspects to a practical character recognition system. The first is to feed and position the character with respect to some scanning device. The second is scanning in order to provide signals which represent the optical image. Thirdly, these signals must be assembled into a coder which represents the image effectively. Finally, a decision-making process operates on this coder to complete classification[4].

OBJECTIVE AND ORGANIZATION OF THESIS

The objective of this thesis is to develop an algorithm for the automatic recognition of handwritten and printed Greek characters. Recognition of printed Greek characters has been investigated in [21] where only topological features were incorporated. This thesis puts forward two schemes for the successful recognition of handwritten and printed Greek characters independent of font and size. Chapter II discusses in brief the two approaches in general. A review of existing character recognition techniques is also presented in this chapter. In chapter III, the general system description is presented. A detailed explanation of the various features utilized in the recognition schemes is also described. Chapter IV presents two different classification processes.

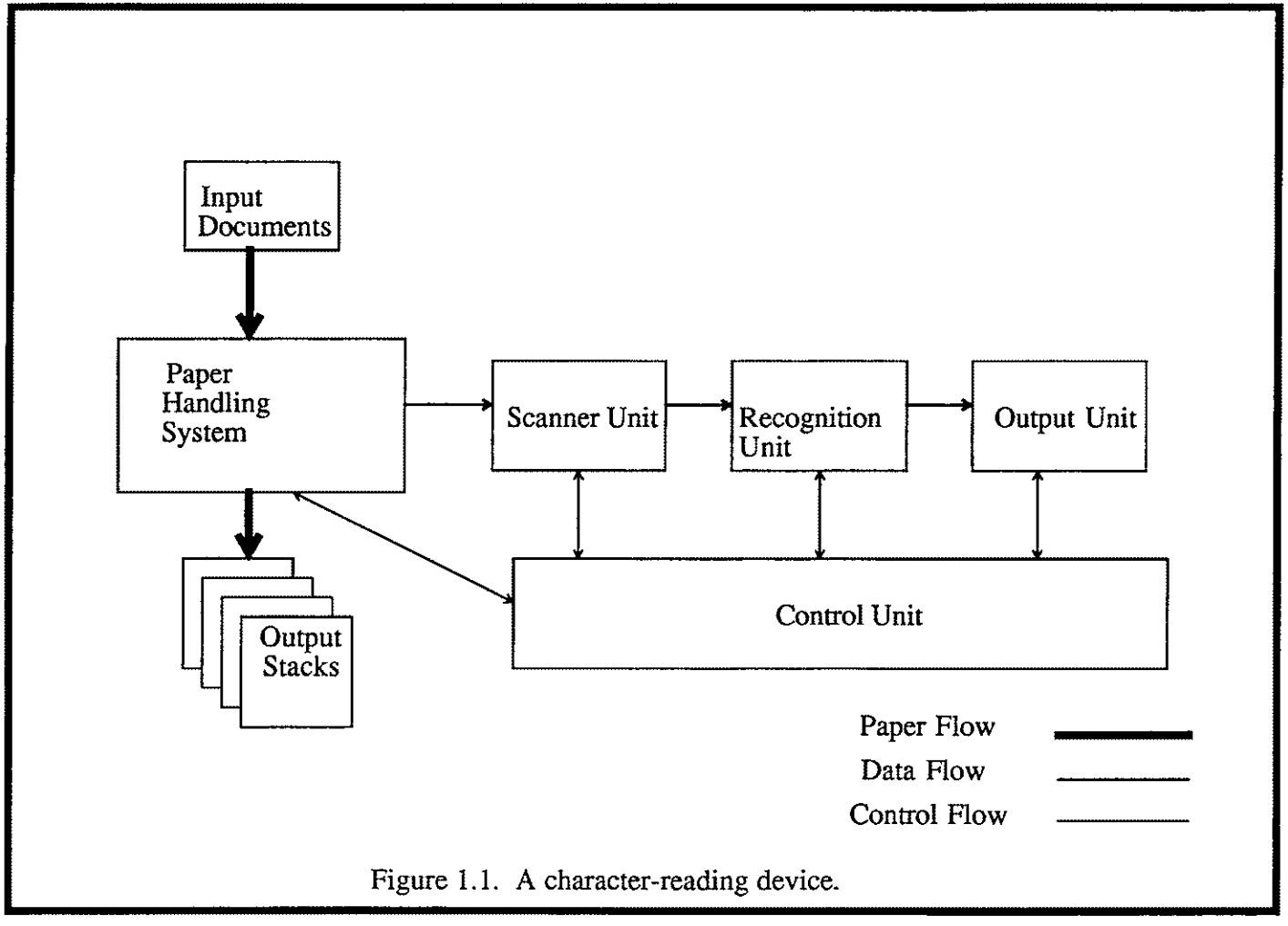


Figure 1.1. A character-reading device.

The first classification process recognizes only printed Greek characters, while the second classification process which is an improvization of the first one, recognizes both printed and handwritten Greek characters. The recognition process based on the second approach is also discussed in this chapter. Chapter V summarizes and discusses the results and finally chapter VI presents the conclusions from this research. A few pictures of captured images and their thinned images are shown in appendix A, while the complete program listing is given in appendix B.

CHAPTER II

BACKGROUND

Several character recognition techniques [5,6,7,8,9] that have been proposed the past few years can be broadly classified into four categories: the feature extraction approach, the statistical- decision theoretic approach, and the syntactic or linguistic approach.

Although a significant amount of research has been devoted to handwritten character recognition, most research has been concentrated on the automatic recognition of printed characters. Automatic character recognition has been accomplished to some extent for Chinese, Latin, Japanese, Hebrew, and several Asian-Indian Languages. In this work we present an algorithm for recognizing handwritten lower case characters of the Greek language, which are also used in engineering and mathematics, regardless of their font and size. In relation to font and size, Kahan *et al.* [10] describe a method that removes the font, size and tilt restrictions of characters while maintaining overall rates close to those of the limited-font commercial machines. No effort is made to find text lines in advance; they begin by detecting and recognizing blobs after which they proceed on with line formation. The Line Adjacency Graph [11] is constructed from a scanned and run length encoded image of the document to be read. Thinning is used as a basic feature extractor yielding a set of strokes approximating the skeleton, resulting in the following features: (i) Number of Holes, (ii) Approximate hole location and size from subgraph matching on the Line Adjacency Graph, (iii) Concavities in the skeletal structure, (iv) Crossings of strokes, and (v) Endpoints in the vertical direction.

Another paper [12] demonstrates the use of polygonal boundary approximation for shape recognition and handwritten character recognition in general. The basic feature of this approach is the use of significant numerical preprocessing of the data without the use of semantic information, followed by an analysis of the reduced data which is based heavily on semantics. First, the boundary points are identified by a tracing algorithm. During this process the number of holes in the figure is determined. After an ordered list of the boundary is obtained, a piecewise linear approximation of only the external boundary is performed. Again the presence of concave arcs, the number and location of holes were significant classifying features of the algorithm. Though it was demonstrated that polygonal approximation generates quite powerful features for classification the results obtained were not as good as when extensive use of semantic information was made.

Baptista and Kulkarni [13] used segmentation and the position of endpoints of the resulting segments as the basic features to describe a character. A syntactic approach is used to parse the character and segment it, while a deterministic approach is used to identify the segment features. Again a syntactic approach is used to provide a complete structural description of the character, and finally the deterministic approach is used to identify the character from a previously stored look up table.

In another paper [14] the combination of a statistical and a structural technique is employed for recognizing numerals. The algorithm is divided into two stages. It first uses concavity measurements to characterize the pattern and a linear discrimination technique which assigns one character to one cluster. The second stage controls the results of the first stage and makes new decisions if necessary. The pre-recognition algorithm is a set of simple rules that performs a rough estimate of the shape of the different "forms" before and after any treatment. The principal criteria are statistical for different characteristics such as: (i) Local height and width of characters with

correction according to slant, (ii) Vertical and horizontal transition between characters and background, (iii) Number of pixels enclosed within characters.

Jeng [15] has developed a new scheme for recognition of chinese characters, where the features selected show the relative position of strokes and the relation between the object and the background. The use of accumulated stroke features that reflect the geometrical and topological properties of a character can account for the pattern variation in multifonts and constrained handwritten chinese characters.

Recently several algorithms have been developed demonstrating the role of neural networks in the field of character recognition [16,17,18,19,20]. Though much of the preprocessing is the same as in pattern recognition, the training and matching process is performed using a large network of neurons.

PROPOSED METHODS

Two algorithms are presented here for the reliable, automatic machine recognition of handwritten and printed Greek characters regardless of size and font. In the first algorithm, which is syntactic in nature, initially a binary image of the character in question is obtained; its skeleton is then produced by utilizing a standard thinning algorithm. To uniquely identify each pattern, the classification process incorporates features of the characters such as existence of a closed curve, number of intersections, number and location of free ends, axial symmetry, as well as criteria derived from normalized moments. The second approach is based on the mathematical modeling of each character with a standard geometrical shape. Considering the morphological and topological features of the Greek characters, each character can be modeled with a single or a combination of standard curves such as, lemniscates, cardioids, ellipses, circles and straight lines. Once the skeleton of the image is obtained, during the recognition process, a best fit to a particular test pattern is obtained from the several equations characterizing each of the twenty-four characters. To demonstrate this approach

characters alpha, beta, gamma, omicron, theta, epsilon, omega, and rho have been selected as sample models.

Since the ultimate goal of this research is to incorporate the Greek character recognizer into a reading machine for the blind, a brief mention of one such machine known as the Kurzweil Reading Machine follows next.

The Kurzweil Reading Machine, or KRM for short, was developed by Raymond Kurzweil. It is a special scanning device which converts several hundred types and styles of print into speech for use by the blind and visually impaired. Although its primary application is as a reading machine, it has several applications as a scanning device, in general KRMs come in various modes and are currently being used in schools, libraries, employment agencies of and for the blind.

OPERATING THE MACHINE

The reader places the book on the glass surface of the machine, presses the top of page button and the automatic page scanner then scans the book, finds the first line and begins to read aloud. As one reads one can stop the machine to spell out a particular word, or one can make the machine read one line at a time and stop or read a paragraph and stop. The machine also remembers the last three hundred characters it has read and hence by pressing the appropriate button, one can go back into its memory and read any number of words or lines again. The beauty of the KRM is its unique shape recognition system, which allows the system to "learn" new characters and type styles and can read, therefore, many books and typeset materials. Since the KRM can learn new characters, its reading improves and becomes accurate with use. If the machine has difficulty with a type style the first time around, it may become more accurate by running the pages through a second time. This Machine unlike most OCRs- recognizes shape, not type faces, a process which more closely approximates

human reading. Other than the price, the KRM has some difficulty with some types of print, although far less than other OCRs. The variety of type fonts, proportional spacing, pictures, and graphs makes it difficult for the computer to analyze the data correctly. The quality of paper is another factor. Thin paper may not work if the camera lights see through it and the print bleeds through from the reverse side. Pictures present obstacles for the OCR when it tries to convert the light and dark images into letters. A third problem with the KRM is that it cannot read handwriting, nor can any other OCR, and here is where the algorithm proposed in this study may have a direct application. The recognition system configuration and the recognition features are dealt with in the next chapter.

CHAPTER III

GENERAL SYSTEM DESCRIPTION

As shown in Figure 3.1, the optical character recognition system used consists of an optical scanner, a digitizer, a preprocessor, and finally the classifier. Isolated characters which are considered for input are scanned by a CCD camera and then digitized. Before the first level of the decision tree is applied, the following preprocessing steps are considered: (i) Thresholding, (ii) Noise Removal, (iii) Character enclosure and thinning.

THRESHOLDING

Since isolated characters are being considered, segmentation is not necessary for it is used for partioning text into individual characters prior to recognition. Thresholding results into the binary image where only two grey levels exist, black and white. This process is also known as binarization. A particular threshold value in the process of creating the binary image, results, to some extent in the thinning of the character. It should be noted that the black pixels refer to the character and the white pixels refer to the background. Also, sparse noise in the form of neighborless pixels is removed.

CHARACTER ENCLOSURE AND THINNING

The noise-free binary image of the character is then bounded by a rectangular array which we shall refer to as the character array: The first black pixel encountered in the character array, if the image is scanned horizontally, belongs to the top row of the array and the last black pixel encountered belongs to the bottom row of the character array. Similarly, if the character array is scanned vertically, the first black pixel

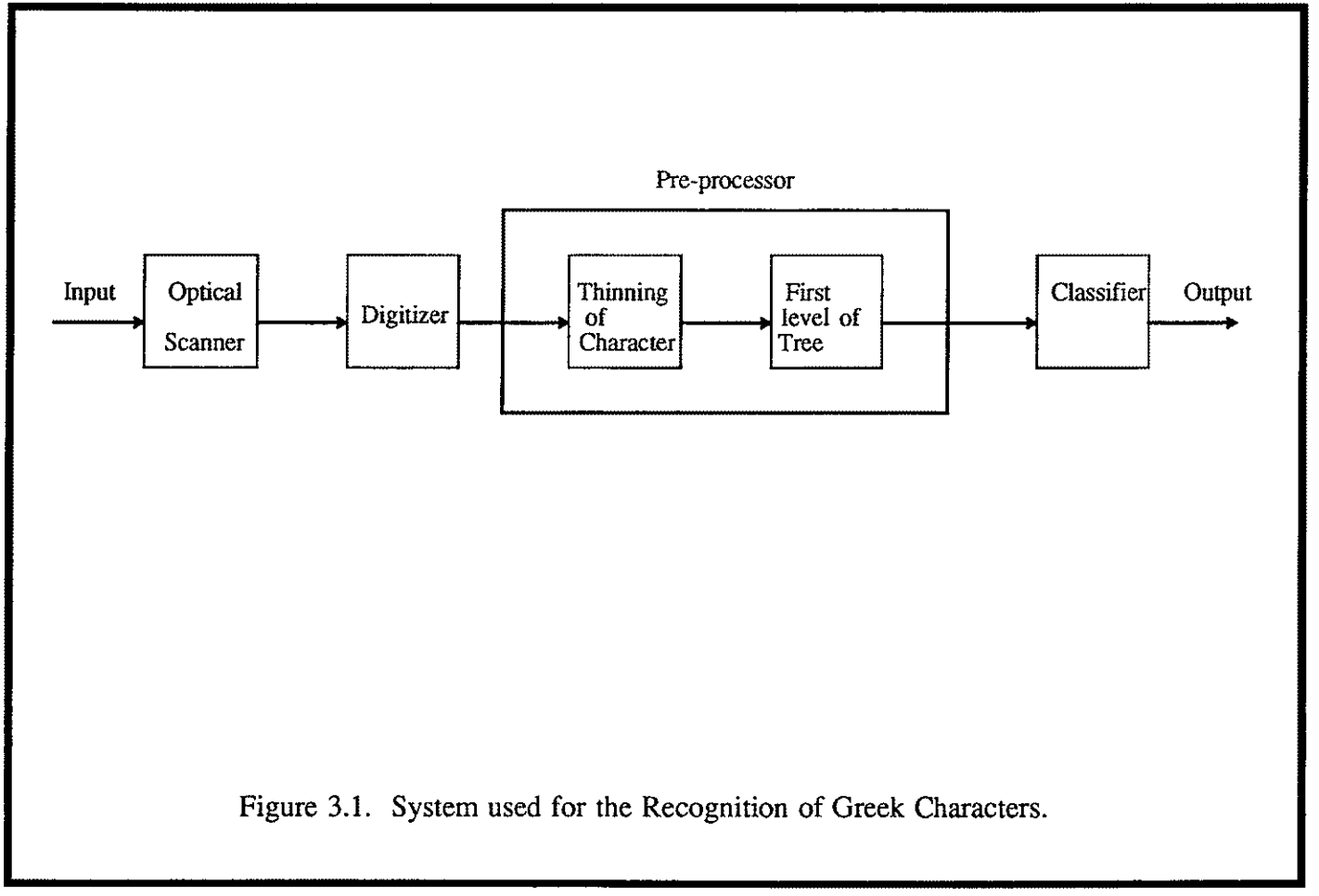


Figure 3.1. System used for the Recognition of Greek Characters.

encountered belongs to the first (left) column of the array and the last black pixel encountered belongs to the last column of the character array. This operation results in a rectangular character array of size N x M.

The character array is then thinned using a standard thinning algorithm [11,22]. Since no normalizing techniques such as averaging have been used, no noise or distortions are introduced in the image. A description of the classification process, which is represented by the final block of the system (Figure 3.1), follows next.

RECOGNITION FEATURES

The unique classification of each of the characters is based on the following features: (i) Existence of closed curve, (ii) Number of intersections, (iii) Number of free ends, (iv) Location of free ends with respect to the horizontal and the vertical reference axis, (v) Horizontal symmetry, (vi) Vertical symmetry, and (vii) Comparisons of the seven invariant moments.

EXISTENCE OF CLOSED CURVE

This feature is utilized at the first level of the classification tree, where the complete set of twenty-four characters is separated into two subsets : one with closed curve characters and the other without. Characters having closed curves are those characters which when written enclose an area. Alpha, Beta, Gamma, Theta, Phi, Omicron, Rho, Delta, and Sigma are the only characters which have some form of area enclosed within them. The remaining characters fall into the other category. The procedure used to detect the existence and the number of closed areas is as follows.

The N x M character array is augmented by two columns (one to the left and one to the right) and two rows (one to the top and the other to the bottom) all filled with white pixels.

The white pixels of the first row are all assigned an arbitrary value ν. Beginning with the second row, whenever a white pixel is encountered the minimum assigned value of the two neighboring points, as shown in Figure 3.2a, is assigned to this pixel. All the black pixels encountered are assigned the value of NM + ν, where NM is the total number of all the pixels (black and white) in the character array and ν is the initial value assigned to the white pixels of the first row. This value of NM + ν never changes. Whenever a white pixel has two neighboring black pixels as in Figure 3.2a, the value of ν is incremented and the new value is assigned to this white pixel. The second step of the process involves the reassignment of all the white pixels with the minimum assigned value of the four neighboring sites as shown in Figure 3.2b. The assignment remains the same (NM + ν) for black pixels. The number of enclosed areas existing in that character is the absolute difference between the maximum assigned value among all white pixels in the character array and the initial value ν.

This procedure can be summarized as follows. Let the pixels of the character array be denoted as

$p(i,j) \quad 1 \le i \le N+2, \; 1 \le j \le M+2.$

p(i,j)= 0 refers to a white pixel and

p(i,j)= 1 refers to a black pixel.

For all p(1,j), $1 \le j \le M+2$, we assign the value $n_{1,j} = \nu$.

Beginning from the second row, for $2 \le i \le N+2, \; 1 \le j \le M+2$

If $p(i,j) = 1, \quad assign \quad n_{i,j} = NM + \nu$

If $p(i,j) = 0$, *then*

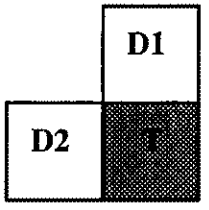


Figure 3.2a

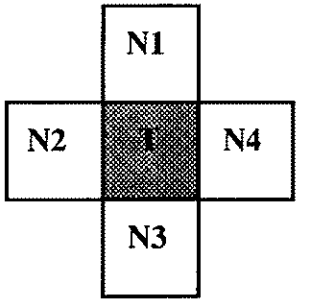


Figure 3.2b

Figure 3.2a. Minimum assigned value of D1 and D2 (D1 and D2 can be both white pixels or one white and the other black pixel) is assigned to T.

Figure 3.2b. The ν of the T pixel is replaced with the minimum assigned ν of N1, N2, N3, N4, and T.

if $p(i-1,j) = p(i,j-1) = 1$, then $n_{i,j} = \max(n_{l,k}) + 1$,

for *all* $l \le i, k < j$ *where* $n_{l,k} \neq NM + \nu$

otherwise *assign* $n_{i,j} = \min(n_{i-1,j}, n_{i,j-1})$

During the second step, for

$3 \le i \le N+2$, $1 \le j \le M+2$,

for each pixel p(i-1,j), with assignment $n_{i-1,j} \neq NM + \nu$

assign $n_{i-1,j} = \min(n_{i-2,j}, n_{i-1,j-1}, n_{i-1,j}, n_{i-1,j+1}, n_{i,j})$

Finally, the number of enclosed areas is

$$max(n_{i,j}) - \nu$$

NUMBER OF INTERSECTIONS

The efficiency in utilizing this particular feature depends on the performance of the thinning algorithm. The character array is scanned horizontally. Whenever a black pixel is encountered, the sum of all the black pixels at the eight neighboring sites, as shown in Figure 3.3, is obtained. It has been determined that if this sum is equal to or greater than four, then an intersection has occured.

That is,

For $1 \le i \le N$, $1 \le j \le M$,

If $p(i,j) = 1$, then evaluate

$$SUM = p(i-1,j-1) + p(i-1,j) + p(i-1,j+1) + p(i,j-1) + p(i,j+1) + p(i+1,j-1) + p(i+1,j) + p(i+1,j+1)$$

If $SUM \ge 4$, *then Number of Intersections* $I = I + 1$,where I is initially set at zero.

After all the pixels have been scanned, the final value of I equals the number of Intersections in the character.

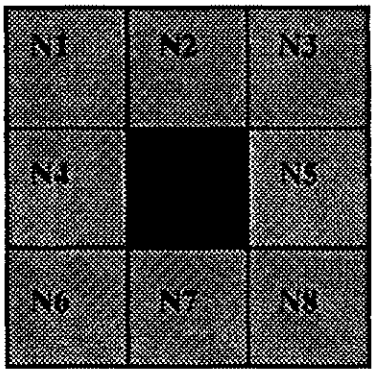


Figure 3.3. N1, N2, N3, N4, N5, N6, N7, and N8 are the eight neighbors considered to determine the existence of an intersection at the black pixel located at the center.

NUMBER OF FREE ENDS

As before, the character array is scanned horizontally. Whenever a black pixel is encountered, the sum total of black pixels at the eight neighboring sites, as shown in Figure 3.3, is evaluated. If this sum is equal to one (implying that the pixel under question has a single neighbor), then a free end exists.

That is,

$For \quad 1 \le i \le N,\ 1 \le j \le M,$

If $p(i,j) = 1$, then evaluate

$$SUM = p(i-1,j-1) + p(i-1,j) + p(i-1,j+1) + p(i,j-1) + p(i,j+1) + p(i+1,j-1) + p(i+1,j) + p(i+1,j+1)$$

$If \quad SUM = 1, \quad then \quad Number\ of\ Free\ ends \quad F = F + 1,$

where F is initially set to zero. The final value of F gives the number of Free ends after all the pixels of the character array are scanned.

LOCATION OF FREE ENDS

The $\frac{N}{2}$ row of the N x M character array is considered as the reference horizontal axis. Using the procedure described above, the number of free ends above and below the horizontal reference axis is determined.

Similarly, the $\frac{M}{2}$ column of the N x M character array is considered as the reference vertical axis. Using the same procedure discussed above, the number of free ends to the left and right of the vertical reference axis is determined.

HORIZONTAL SYMMETRY

This feature is not required for the recognition of handwritten characters, however, when it is known that the character is printed it can be used instead of the feature regarding the location of free ends.

Let $R_1, R_2, R_3, \ldots\ldots R_N$, be the N rows of the character array.

R_1 *has elements* $p_{11}, p_{12}, \cdots p_{1M}$ and similarly R_N *has elements* $p_{N1}, p_{N2}, \cdots p_{NM}$.

A sequential row comparison of the pixels is initialized, i.e., the first row is compared with the last one, the second with the second last and so on. If $R_1 = R_N, R_2 = R_{N-1}, \ldots R_k = R_{N-k}$ for all $k \leq N/2$ then the character under test is horizontally symmetric.

VERTICAL SYMMETRY

The approach to determine vertical symmetry in a character is similar to that for determining horizontal symmetry. This feature, also, is not required for recognition of handwritten characters.

Let $C_1, C_2, C_3, \ldots\ldots C_M$, be the M columns of the character array.

C_1 *has elements* $p_{11}, p_{21}, \cdots p_{N1}$ and similarly C_M *has elements* $p_{1M}, p_{2M}, \cdots p_{NM}$.

A sequential column comparison of the pixels is initialized, i.e., the first column is compared with the last one, the second with the second last and so on. If $C_1 = C_M, C_2 = C_{M-1}, \ldots C_k = C_{M-k}$ for all $k \leq M/2$ then the character under test has vertical symmetry.

CRITERION BASED ON NORMALIZED MOMENTS

Given any two dimensional continuous function f(x, y), a moment of order (p+q) is defined by,

$$m_{pq} = \int\int x^p \; y^q \; f(x,y)dxdy$$

where p, q= 0,1,2..

The central moment for the same function f(x, y) is given as

$$U_{pq} = \int\int (x - \bar{x})^p \; (y - \bar{y})^q \; f(x,y)dxdy$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \; \bar{y} = \frac{m_{01}}{m_{00.}}$$

For a digital image, the above equations become,

$$M_{pq} = \sum_i \sum_j i^p \; j^q \; f(i,j)$$

$$U_{pq} = \sum_i \sum_j (i - \bar{i})^p \; (j - \bar{j})^q \; f(i,j)$$

where

$$\bar{i} = \frac{M_{10}}{M_{00}}, \; \bar{j} = \frac{M_{01}}{M_{00.}}$$

The central moments have been shown [22] to be invariant to translation of an image.

Another set of moments have been derived [22] from the central moments which are invariant to the size of the image. These normalized central moments are denoted by η_{pq} and are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma}$$

where

$$\gamma = \frac{p+q}{2} + 1$$

for p+q = 2,3,...

Hu [23,24] derived the following set of seven moments invariant to size and orientation.

$$MM_1 = \eta_{20} + \eta_{02}$$

$$MM_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$MM_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$MM_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$MM_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$$
$$+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$MM_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$MM_7 = (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$$
$$+ (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12}^2) - (\eta_{21} + \eta_{03})^2]$$

where η_{pq} is the normalized central moment.

Let MM_i^j, (i = 1,. . .,7, and j = 1,2,. . ,24) be the invariant moments of each character with index j evaluated and stored in a database before the recognition process.

Let N_i x M_i be the size of the character array in the database, where i= 1,...,24 (total number of characters). Let the test character be of size n x 1.

If m_1, m_2, m_3, m_4, m_5, m_6, and m_7 are the seven moments evaluated for the test character while in the recognition process, then

$$MT_i = \frac{m_i * N_i * M_i}{MM_i * n * l}$$

MT_i are the normalized test moments, where i= 1,....,7.

If MT_i^k, (i = 1,2,. .,7) are the invariant moments obtained for the character in question, with index value k, in the recognition process,

then

$$MSE(j) = \sum_{i=1}^{7} [MM_i^j - MT_i^k]^2$$

where j= 1,2,3,....., 24 (the total number of characters), represents the mean-square error between the moments of the test pattern and every set of moments stored in the database for each character.

If the minimum of all MSE(j) occurs when $j = k$ then the character in question is identified as the one from the database with index value k.

The Classification process of the classifier which recognizes just the printed characters and the second classifier which recognizes both handwritten and printed Greek characters simultaneously is dealt with next. A separate section on the second approach referring to the mathematical modeling aspect follows at the end of the classification processes.

CHAPTER IV

CLASSIFICATION PROCESS

CLASSIFIER FOR PRINTED GREEK CHARACTERS

The tree structure of the classifier for recognizing printed Greek characters is shown in Figure 4.1.

The first level of the tree which determines the existence of closed curve in a character divides the character set {α, β, χ, δ, ε, η, γ, ι, κ, λ, μ, ν, ω, ο, φ, π, ψ, ρ, σ, τ, θ, υ, ξ, ζ} into two groups, one with closed curves {α, β, γ, δ, θ, φ, ρ, σ, ο}, and the other without {χ, ε, η, ι, κ, λ, μ, ν, π, ω, ξ, ψ, ζ, τ, υ}.

The character set {α, β, γ, δ, θ, φ, ρ , σ, ο}, upon application of the second level of the tree, which determines the number of enclosed areas in a character, is divided into two groups, the first consists of characters with one enclosed area {α, γ, ρ, σ, δ , ο}, and the other of characters with two enclosed areas {β, θ, φ}. The third level of the tree which utilizes the feature determining the number of free ends in a character classifies the set {β, θ, φ} into{θ}, {β}, and {φ}, with the free ends being zero, one, and two respectively. The set {α, γ , ρ, σ, δ, ο}, on the other hand, upon application of the feature regarding horizontal and vertical symmetry is divided into {ο} and {α, γ , ρ, σ, δ}. The fourth level of the tree which tests for horizontal symmetry in a character results in {α} and { γ , ρ, σ, δ}. The fifth level of the tree which determines vertical symmetry in a character subdivides { γ , ρ, σ, δ} into {γ} and { ρ, σ, δ}. The set { ρ, σ, δ} is next classified utilizing the moment invariance principle.

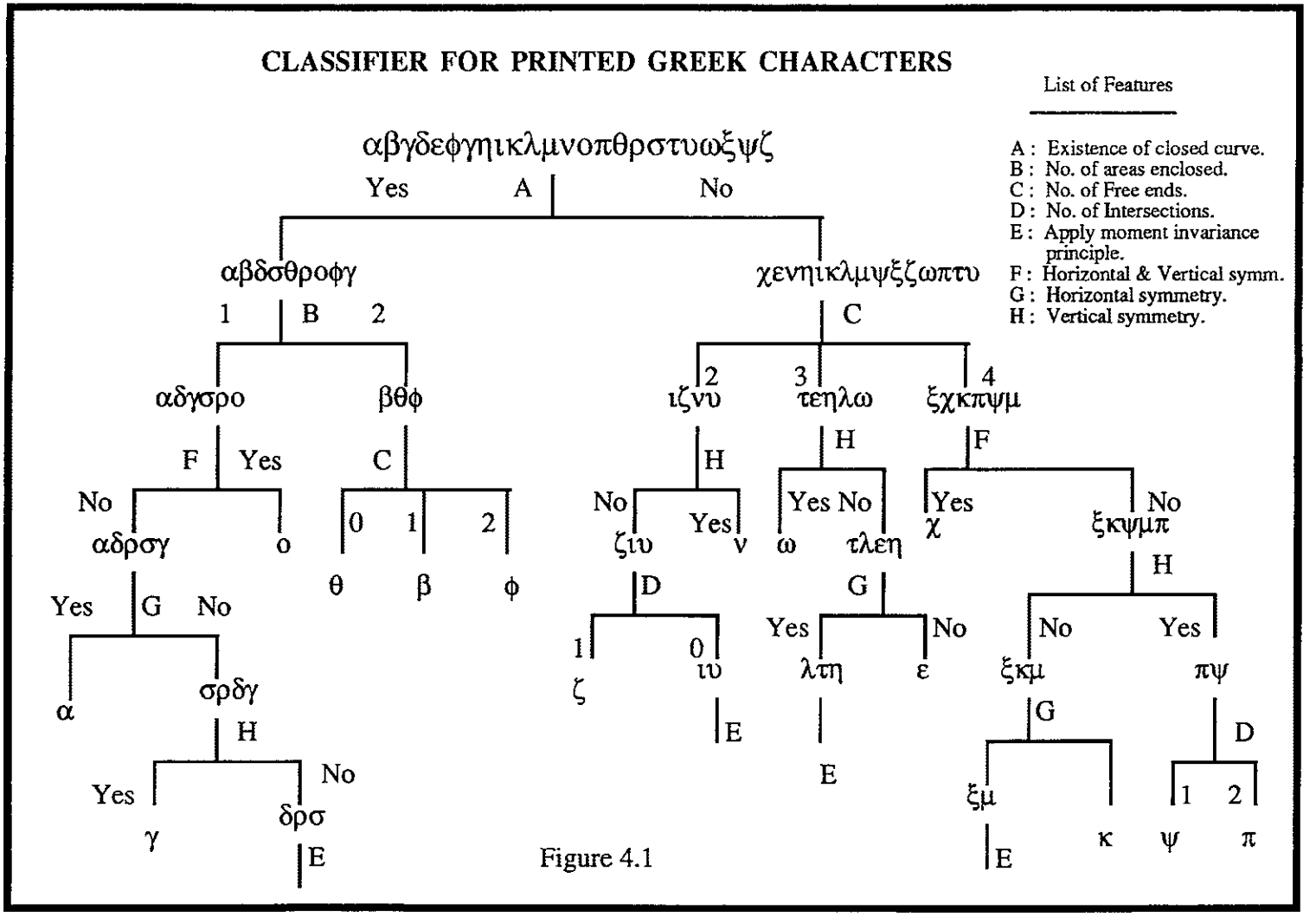

CLASSIFIER FOR PRINTED GREEK CHARACTERS
List of Features
A : Existence of closed curve.
B : No. of areas enclosed.
C : No. of Free ends.
D : No. of Intersections.
E : Apply moment invariance principle.
F : Horizontal & Vertical symm.
G : Horizontal symmetry.
H : Vertical symmetry.
αβγδεφγηικλμνοπθρστυωξψζ
Yes
A
No
αβδσθροφγ
χενηικλμψξζωπτυ
1
B
2
C
αδγσρο
βθφ
ιζνυ
τεηλω
ξχκπψμ
2
3
4
F
Yes
C
H
H
F
No
0
1
2
No
Yes
Yes No
Yes
No
αδρσγ
ο
ζιυ
ν
ω
τλεη
χ
ξκψμπ
θ
β
φ
Yes
G
No
D
G
H
1
0
Yes
No
No
Yes
α
σρδγ
ζ
ιυ
λτη
ε
ξκμ
πψ
H
G
D
Yes
No
E
E
ξμ
1
2
δρσ
γ
κ
ψ
π
E
E

Figure 4.1

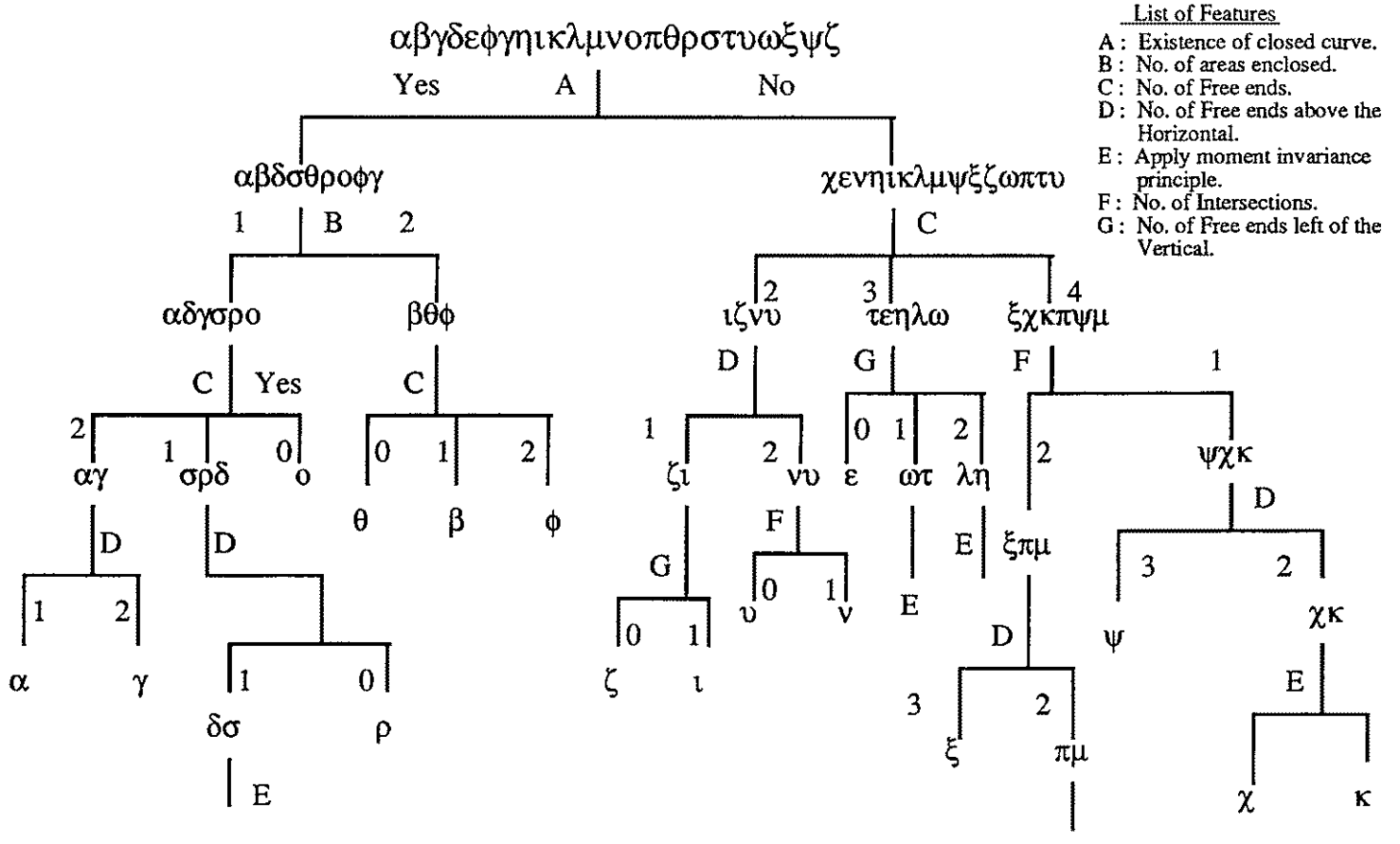
The character set {χ, ε, η, ι, κ, λ, μ, ν, π, ω, ξ, ψ, ζ, τ, υ} which does not consist of closed curves, during the second level of the tree, utilizes the feature determining the number of free ends to result in three sets, {ι, ζ, ν, υ}, {τ, η, ε, λ, ω} and {ξ, χ, κ, π, ψ, μ}, with free ends two, three, and four respectively. In the third level of the tree the feature determining vertical symmetry in a character separates the set {ι, ζ, ν, υ} into {ν} and {ι, ζ, υ}, whereas the set {τ, η, ε, λ, ω} is divided into {ω} and {τ, β, ε, λ}. At the same level the set {ξ, χ, κ, π, ψ, μ} is subdivided into the set {χ}, having both horizontal and vertical symmetry, and the other {ξ, κ, π, ψ, μ} without this feature. Existence of horizontal symmetry isolates {ε} from the set {τ, η, ε, λ} in the fourth level of the tree, whereas the rest {τ, η, λ} are classified distinctly utilizing the criteria of moment invariance during the fifth level of the tree. The set {ι, ζ, υ} upon application of the feature determining the number of intersections in a character is separated into {ι, υ} with zero intersections and {ζ} with one intersection. The principle of moment invariance subsequently classifies the set {ι, υ} into {ι} and {υ}. The set {ξ, κ, π, ψ, μ} on the other hand utilizes the feature involving the existence of vertical symmetry, to be separated into two sets, {π, ξ} and {ψ, κ, μ}. In the fifth level of the tree, character {κ} with horizontal symmetry is isolated from the set {ψ, κ, μ}, leaving {ψ, μ} to be separated with the moment invariance principle. The set {π, ξ} meanwhile, upon application of the feature determining the number of intersections in a character is separated into {ξ} and {π} with one and two intersections respectively.

CLASSIFIER FOR HANDWRITTEN AND PRINTED GREEK CHARACTERS

The tree structure of the classifier for recognizing handwritten and printed Greek characters is shown in Figure 4.2.

The first level of the tree which determines the existence of closed curve in a character divides the character set {α, β, χ, δ, ε, η, γ, ι, κ, λ, μ, ν, ω, ο, φ, π, ψ, ρ, σ, τ, θ,

CLASSIFIER FOR HANDWRITTEN AND PRINTED GREEK CHARACTERS



List of Features
A : Existence of closed curve.
B : No. of areas enclosed.
C : No. of Free ends.
D : No. of Free ends above the Horizontal.
E : Apply moment invariance principle.
F : No. of Intersections.
G : No. of Free ends left of the Vertical.

Figure 4.2

υ, ξ, ζ} into two groups, one with closed curves {α, β, γ, δ, θ, φ, ρ, σ, o}, and the other without {χ, ε, η, ι, κ, λ, μ, ν, π, ω, ξ, ψ, ζ, τ, υ}.

The character set {α, β, γ, δ, θ, φ, ρ , σ, o}, upon application of the second level of the tree, which determines the number of enclosed areas in a character, is divided into two groups, the first with one enclosed area {α, γ, ρ, σ, δ , o}, and the second with two enclosed areas {β, θ, φ}. The third level of the tree which utilizes the feature determining the number of free ends in a character classifies the set {β, θ, φ} into {θ}, {β}, and {φ}, with the free ends being zero, one, and two respectively. The set {α, γ , ρ, σ, δ, o}, however, upon application of this feature, is divided into three subsets {o}, {ρ, σ, δ}, and {α, γ}, with the free ends being zero, one and two respectively. The fourth level of the tree, which determines the number of free ends above the horizontal axis, breaks down the set {ρ, σ, δ} into two groups, {δ, σ} with one free end above the horizontal axis, and {ρ} with zero free ends above the horizontal axis. The set {α, γ} is divided into {α} and {γ} with the free ends above the horizontal axis being one and two respectively. The fifth level of the tree utilizes the moment invariance principle to classify {δ} and { σ} distinctly.

The character set {χ, ε, η, ι, κ, λ, μ, ν, π, ω, ξ, ψ, ζ, τ, υ} which does not consist of closed curves, during the second level of the tree, utilizes the feature determining the number of free ends to result in {ι, ζ, ν, υ}, {τ, η, ε, λ, ω} and {ξ, χ, κ, π, ψ, μ}, with free ends two, three, and four respectively. The third level of the tree utilizes the feature evaluating the number of free ends above the horizontal to decompose the set {ι, ζ, ν, υ} into {ι, ζ} and {ν, υ}, with the feature values being one and two respectively. The set {τ, η, ε, λ, ω} on the other hand utilizes the feature determining the free ends to the left and right of the vertical to result in three subsequent sets {ε}, {ω, τ} and {λ, η} with feature values being zero, one, and two respectively. The third set which has resulted after the application of the second level, namely {ξ, χ, κ, π, ψ, μ} is separated by the feature determining the number of intersections resulting in {ξ, π, μ}

and {ψ, χ, κ} with the intersections being two and one respectively. The fourth level of the tree incorporating the feature determining the number of free ends to the left and right of the vertical axis, separates the set {ι, ζ} into {ι} and {ζ} distinctly. The set {ν, υ} is divided by the feature determining the number of intersections resulting in {ν}, and {υ}, with the number of intersections being one and zero respectively each of the cases. Character sets {ω, τ} and {λ, η} are to be classified using the moment invariance principle. Also, the sets {ξ, π, μ} and {ψ, χ, κ} use the feature involving the detection of the number of intersections to result in {ξ}, {π, μ} with feature values three and two, and {ψ}, {χ, κ} with feature values three and two again, respectively. The fifth level of the tree utilizes the principle of moment invariance to separate the sets {π, μ} and {χ, κ} into {π}, {μ}, {χ}, and {κ}.

The features involving the vertical and horizontal symmetry of a character are not used for handwritten characters, however, they may be used as optional features, replacing the features determining the number of free ends, when recognizing printed characters.

GEOMETRICAL MODELING APPROACH

Let us now consider the second approach attempted for the Recognition of handwritten Greek characters. Among the characters successfully modeled are alpha, beta, gamma, omicron, theta, rho, epsilon, and gamma. Mathematical modeling refers to the process of representing each character with a single or a combination of standard geometrical shapes. The geometrical shapes considered in our case are Lemniscates, Cardioids, Ellipses , Circles, and finally Straight lines. Table 4.1., gives the equations [25] describing each of the curves mentioned above, and Figure 4.3 shows some of the characters successfully modeled and recognized as the geometrical shapes shown next to them.

TABLE 4.1

FIGURE	EQUATION
LEMNISCATE	$r^2 = 2a^2\cos 2\theta$
CARDIOID	$r = a(1 - \cos\theta)$
CIRCLE	$x^2 + y^2 = r^2$
ELLIPSE	$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
STRAIGHT LINE	$y = mx + c$

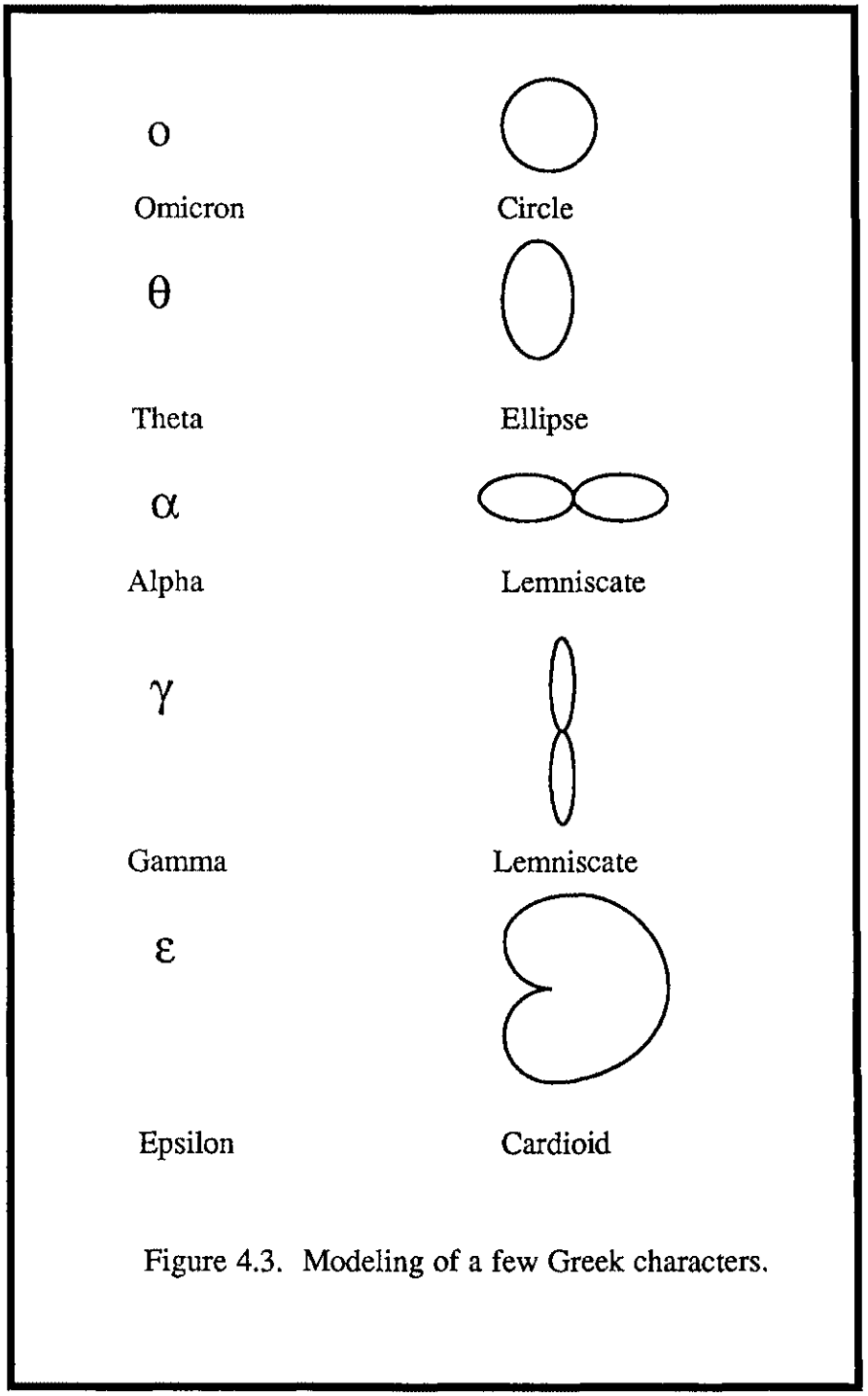


Figure 4.3. Modeling of a few Greek characters.

Theory

Since most of the characters can be represented by a single or a combination of geometrical shapes, each has a unique equation describing it. Once the character array is obtained, the one among the twenty four equations which best fits the test character maps the test character to the correct one in the database. The least square error approach is utilized to obtain the best fit.

Let

$$(x_1,y_1),(x_2,y_2),.....,(x_n,y_n)$$

be the coordinates of the pixels comprising of the test character.

If

$$y = f(x)$$

is a relation satisfying any of the five geometrical shapes discussed above, then the particular equation for which

$$\sum_{i=1}^{N}[y_i - f(x_i)]^2$$

is minimum resembles to that particular character. Scaling is considered next.

Let 'k' be a scaling factor, which means the equation is of the form

$$y = kf(x) \qquad (4.1)$$

Now mean square error for (4.1) is obtained by minimizing

$$\frac{1}{[\sum_{j=1}^{N}|f(x_j)|]^2}\sum_{i=1}^{N}[\frac{y_i}{k} - f(x_i)]^2 \qquad (4.2)$$

Upon substitution of (4.1), minimizing (4.2), is the same as minimizing

$$\sum_{i=1}^{N}[\frac{y_i}{\sum_{j=1}^{N}y_j} - \frac{f(x_i)}{\sum_{j=1}^{N}f(x_j)}]^2 \qquad (4.3)$$

For any j,

$$y_j - f(x_j) = \delta_j \qquad (4.4)$$

where $\delta_j \Rightarrow 0$ for the closest match.

Under this condition , i.e., for $\sum_{j=1}^{N} y_j \approx \sum_{j=1}^{N} kf(x_j)$, minimizing (4.3) is the same as minimizing

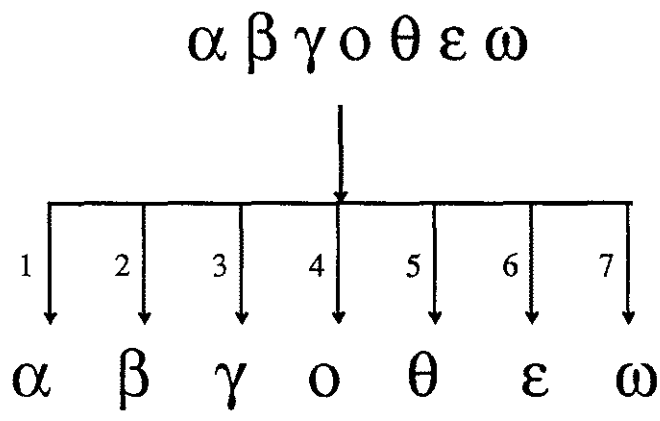
$$\sum_{i=1}^{N} \left[\frac{y_i}{\sum_{j=1}^{N} kf(x_j)} - \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)} \right]^2 \qquad (4.5)$$

which then leads to

$$\frac{1}{\left[\sum_{j=1}^{N} |f(x_j)|\right]^2} \sum_{i=1}^{N} \left[\frac{y_i}{k} - f(x_i) \right]^2$$

which is (4.2) with which we started with.

Hence, to summarize, minimizing (4.2), is the same as minimizing (4.3) which does not contain the scaling factor k. The objective of eliminating k is thus achieved, thereby keeping the above procedure size invariant. It has been shown in figure 4.4 that each of the sample characters have a unique representation using the approach of geometrical modeling. Experimental results are discussed in the next chapter.



1. Resembles Lemniscate.
2. Resembles Cardioid in the region $0<\theta<\pi$.
3. Resembles Lemniscate rotated $\pi/2$ degrees along the clockwise x-axis.
4. Resembles Circle.
5. Resembles Ellipse with major y-axis.
6. Resembles Cardioid rotated by π degrees along the y-axis.
7. Resembles Cardioid rotated by $\pi/2$ degrees along the anticlockwise x-axis.

Figure 4.4. Character recognition based on the Geometrical Modeling approach.

CHAPTER V

EXPERIMENTAL RESULTS

The algorithms described in this thesis were implemented in FORTRAN on a DEC VAX 11/750. A 256 x 256 image of the character was captured and subsequently digitized. The preprocessing steps, thresholding, character enclosure and thinning, generate the character array of the image.

The database library consists of a single set of feature vectors, where each vector has the following features: number of enclosed areas, number of intersections, total number of free ends, number of free ends above the reference horizontal axis and finally the number of free ends to the left of the reference vertical axis. Experiments were performed for five different sets of printed Greek characters and seven different sets of lower case handwritten Greek characters. Table 5.1 and Table 5.2 illustrate the topological features for each of the twenty four characters in both printed and handwritten characters.

Figure 5.1 shows the original captured image of the character phi (ϕ). The thinned character array of the same image is illustrated in Figure 5.2. The rectangular grid in which phi (ϕ) is enclosed is shown in figure 5.3a. The number of enclosed areas in phi (ϕ) is shown evaluated in Figure 5.3b. It is seen in the figure that the initial assignment to v is one, and by the time the entire character array is scanned, this assignment changes to three. As described in chapter III, from the figure, the number of enclosed areas is $max(n_{i,j})$-v, i.e., 3-1 as in this case, resulting in two enclosed areas.

TABLE 5.1

Feature values of the lower case Printed Greek characters					
character	# of Areas	# of Intersections	# of Free ends	Horizontal symmetry	Vertical symmetry
alpha, α	1	1	2	Yes	No
beta, β	2	2	1	No	No
chi, χ	0	1	4	Yes	Yes
delta, δ	1	1	1	No	No
epsilon, ε	0	1	3	Yes	No
eta, η	0	1	3	No	No
gamma, γ	1	1	2	No	Yes
iota, ι	0	0	2	No	No
kappa, κ	0	1	4	Yes	No
lambda, λ	0	1	3	No	No
mu, μ	0	2	4	No	No
nu, ν	0	1	2	No	Yes
omega, ω	0	1	3	No	Yes
omicron, ο	1	0	0	Yes	Yes
phi, φ	2	2	2	Yes	Yes
pi, π	0	2	4	No	Yes
psi, ψ	0	1	4	No	Yes
rho, ρ	1	1	1	No	No
sigma, σ	1	1	1	No	No
tau, τ	0	1	3	No	No
theta, θ	2	2	0	Yes	Yes
upsilon, υ	0	0	2	No	No
xi, ξ	0	2	4	No	No
zeta, ζ	0	1	3	No	No

TABLE 5.2

Feature values of the lower case Greek characters					
character	# of Areas	# of Intersections	# of Free ends (total)	#of Free ends above the horizontal axis	# of free ends to the left of the vertical axis
alpha, α	1	1	2	1	0
beta, β	2	2	1	0	1
chi, χ	0	1	4	2	2
delta, δ	1	1	1	1	0
epsilon, ε	0	1	3	1	0
eta, η	0	1	3	1	2
gamma, γ	1	1	2	2	1
iota, ι	0	0	2	1	1
kappa, κ	0	1	4	2	2
lambda, λ	0	1	3	1	2
mu, μ	0	2	4	2	2
nu, ν	0	1	2	2	1
omega, ω	0	1	3	2	1
omicron, o	1	0	0	0	0
phi, φ	2	2	2	1	0
pi, π	0	2	4	2	2
psi, ψ	0	1	4	3	2
rho, ρ	1	1	1	0	1
sigma, σ	1	1	1	1	0
tau, τ	0	1	3	2	1
theta, θ	2	2	0	0	0
upsilon, υ	0	0	2	2	1
xi, ξ	0	2	4	3	1
zeta, ζ	0	1	3	2	1

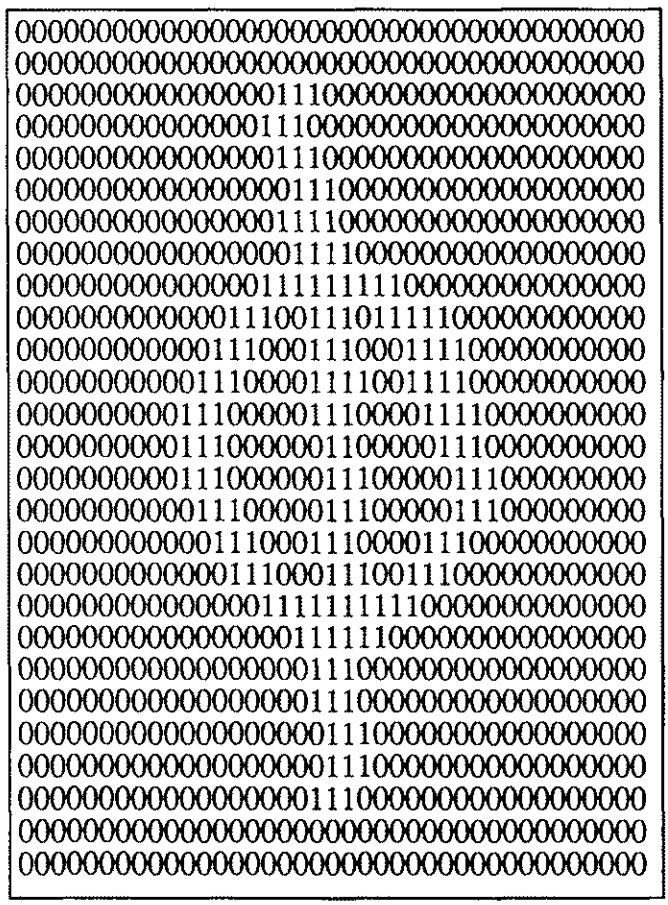


Figure 5.1. Captured image of character phi (ϕ).

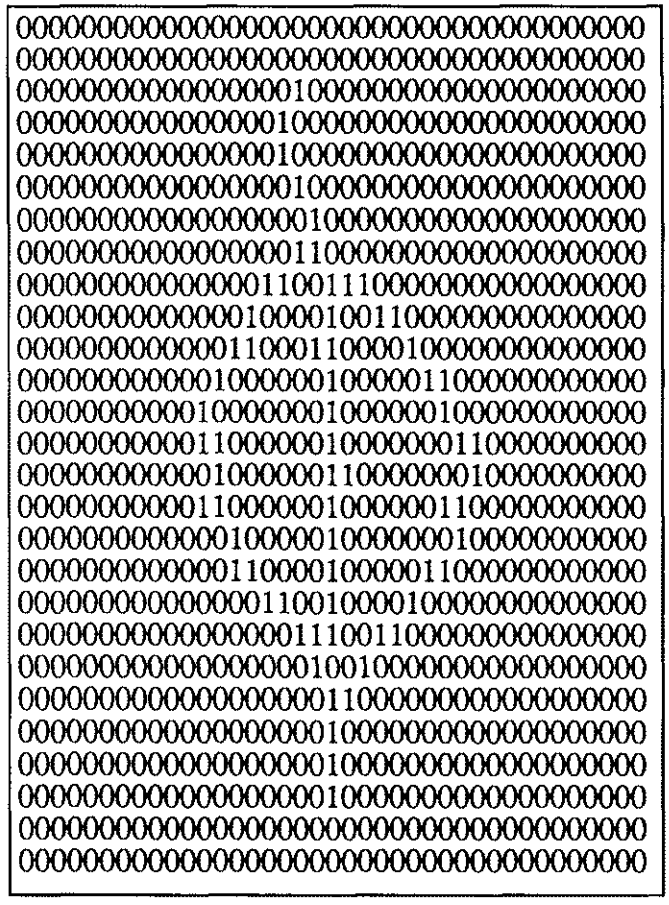


Figure 5.2. Thinned image of character phi (ϕ).

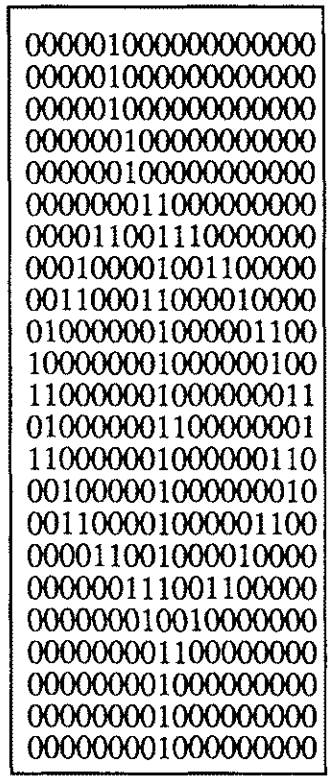


Figure 5.3a. Character array of phi (ϕ) obtained after the completion of the pre-processing steps. The array is next utilized to detect the existence of closed areas in the character phi (ϕ).

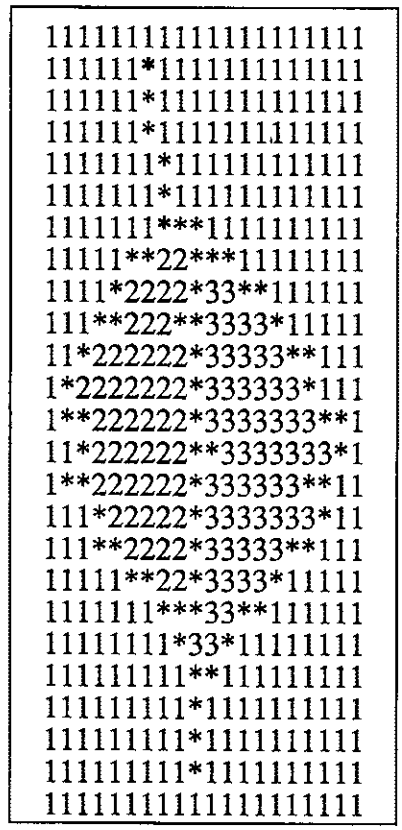


Figure 5.3b. The procedure to detect the number of closed areas in the character phi (ϕ) results in two such areas. "*" refers to the black pixels in the character array.

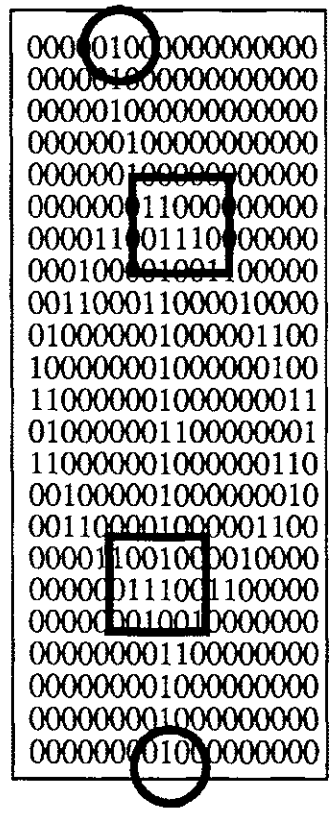


Figure 5.4. The circles signify the presence of the two free ends and the squares represent the existence of the two intersections in the character phi (ϕ).

The intersections and the free ends in the character phi (ϕ) are marked as shown in Figure 5.4. As it can be seen from the figure, the number of intersections is two and the number of free ends is also two.

The principle of moment invariance is mostly utilized to classify a few sets of characters belonging to the sub-group having zero enclosed areas. Table 5.3 lists the seven invariant moments for the characters that are uniquely recognized only when the moment criterion is applied. Each set of experiments resulted in a 100% recognition rate.

TABLE 5.3

Comparison of the Seven Invariant Moments(MM_i)							
Character	MM1	MM2	MM3	MM4	MM5	MM6	MM7
delta, δ	1.0986	2.302	3.178	2.39	5.739	4.304	4.6051
sigma, σ	10.052	9.1519	7.778	5.89	3.09	4.6051	3.583
kappa, κ	6.459	6.008	7.001	4.454	4.43	4.6051	3.496
pi, π	7.196	5.94	7.329	5.176	2.639	4.094	4.6051
upsilon, υ	5.45	4.077	4.744	4.454	3.044	3.091	4.6051
xi, ξ	6.652	6.770	6.073	5.424	5.446	5.924	4.6051
omega, ω	2.944	3.828	4.234	2.564	4.6051	3.806	2.079
tau, τ	2.564	3.891	4.043	2.944	5.416	4.779	4.6051
lambda, λ	1.791	2.197	4.043	2.079	4.248	3.218	4.6051
eta, η	2.890	2.484	4.043	3.526	5.54	3.951	4.6051

CHAPTER VI

CONCLUSIONS

In this thesis we have presented two approaches for the reliable, automatic machine recognition of handwritten and printed Greek characters invariant to the size and font.

Most of the previous works in the field of character recognition are based on feature extraction techniques and a syntactic approach for classification. Moments as such have been utilized [26] for character recognition whereby the experimentations were carried out on only printed latin characters. Large sets of apriori data in the library were required and above all the recognition rate was not very satisfactory.

The first approach presented in this thesis is a combination of the syntactic and the moment invariant properties. In this approach, the binary image of the character is obtained, enclosed in a rectangular grid and finally thinned to one pixel thick skeleton utilizing a standard thinning algorithm. Topological features of the characters such as the existence of closed curves, number of intersections, number and location of free ends, axial symmetry and principle of moment invariance are incorporated in the process to correctly classify each of the characters.

Unlike most of the techniques and approaches which have been utilized for character recognition, the second approach presented in this thesis is an entirely different and new technique for character recognition. In this approach, considering the morphological and topological features of the Greek characters, each character is to be mathematically modeled with a standard geometrical shape. After having undergone similar preprocessing as the first approach, each pattern is modeled with a single or a

combination of standard geometrical shapes such as lemniscates, cardioids, ellipses, circles and straight lines. During the recognition process, a best fit to a particular pattern is obtained from each equation characterizing each of the twenty-four characters. The mean-square-appraoch is utilized for determing the best fit. This approach would be more time efficient than the syntactic one. Research will be conducted on the utilization of additional geometrical shapes other than the ones already being used. In the future, work will be performed to come up with a certain set of invariant moments other than the ones already in existence, which will be sufficient to uniquely classify the characters without considering the other topological features.

Both of the greek character recognition processes can be incorporated in the reading machine for the blind.

LIST OF REFERENCES

[1] K. S. Fu, *Applications of Pattern Recognition,* CRC press, 1983.

[2] J. D Erwin, D. R Duvall, and R. K Habitzreiter, "Single Read station Acquisition for Character Recognition," U.S Patent 4,013,999, 1977.

[3] S. C Requa, "Hand-operated Optical Character Reader Wand," U.S patent 3,947,817, 1976.

[4] L. D Harmon, "Automatic Recognition of Print and Script," *Proc. of the IEEE,* vol. 60, no. 10, October 1972.

[5] C. Y. Suen, M. Berthod, and S. Mori, "Automatic Recognition of Hand-Printed Characters- The State of the Art," *Proc. IEEE 68,* pp. 469-487, 1980.

[6] S. L. Xie and M. Suk, "On Machine Recognition of Hand-Printed Chinese Characters by Feature Relaxation," *Pattern Recognition,* vol. 21, no. 1, pp. 1-7, 1988.

[7] I. Sekita, K. Toraichi, R. Mori, K. Yamamoto, and H. Yamada, "Feature Extraction of Handwritten Japanese Characters by Spline Functions for Relaxation Matching," *Pattern Recognition,* vol. 21, pp. 9-17, 1988.

[8] P. D. Deighton, "A Statistical Approach to Optical Character Recognition," Research Department Report 559, Post Office Research Center, Martlesham Heath, Ipswich, England, 1976.

[9] I. K. Sethi and B. Chatterjee, "Machine Recognition of Constraint Hand Printed Devanagari," *Pattern Recognition,* vol. 9, pp. 69-75, 1977.

[10] S. Kahan, T. Pavlidis, and H. S. Baird, "On the Recognition of Printed Characters of Any Font and Size," *IEEE Trans. PAMI,* vol. PAMI-9, no. 2, pp. 274-288, March 1987.

[11] T. Pavlidis, *Algorithms for Graphics and Image Processing,* Rockville, MD, Computer Science Press, 1982.

[12] T. Pavlidis and F. Ali, "Computer Recognition of Handwritten Numerals by Polygonal Approximation," *IEEE Trans. on Systems, Man and Cybernetics,* vol. SMC-5, no. 6, Nov. 1975.

[13] G. Baptista and K. M. Kulkarni, "A High Accuracy Algorithm for Recognition of Handwritten Numerals," *Pattern Recognition,* vol. 21, no. 4, pp. 287-291, 1988.

[14] E. Lecolinet and Jean-Vincent Moreau, "A New System for Automatic Segmentation and Recognition of Unconstrained Handwritten Zip Code," *The 6th Scandinavian Conference on Image Analysis,* Oulu, Finland, pp. 585-592, June 19-22, 1989.

[15] Bor-Shen Jeng, "Optical Chinese Character Recognition using Accumulated Stroke Features," *Optical Engineering,* vol. 28, no. 7, pp. 793-799, July 1989.

[16] W. L. Reber and J. Lyman, "An Artificial Neural System Design for Rotation and Scale Invariant Pattern Recognition," *IEEE First Int. Conf. on Neural Networks,* San Diego, CA, vol. iv, pp. 277-283, June 1987.

[17] D. Mehr and S. Richfield, "Neural Net Application to Optical Character Recognition," *IEEE First Int. Conf. on Neural Networks,* San Diego, CA, vol. iv, pp. 771-777, June 1987.

[18] T. F. Pawlicki, Dar-Shyang Lee, J. J. Hull, and S. N. Srihari, "Neural Network Models and their Application to Handwritten Digit Recognition," *IEEE First Int. Conf. on Neural Networks,* San Diego, CA, vol. ii, pp. 63- 70, June 1987.

[19] A. Khotanzad and J. H. Lu, "Distortion Invariant Character Recognition by a Multi-Layer Perceptron and Back Propagation Learning," *IEEE Int. Conf. on Neural Networks,* San Diego, CA, vol. 2, pp. 625-632, July 1988.

[20] L. D. Jackel, H. P. Graf, W. Hubbard, J. S. Denker, D. Henderson, and I. Guyon, "An Application of Neural Net Chips: Handwritten Digit Recognition," *IEEE Int. Conf. on Neural Networks,* San Diego, CA, vol. 2, pp. 107- 115, July 1988.

[21] N. Alvertos and I. D' Cunha, "Optical Machine Recognition of Greek Characters of Any Size," *IEEE Proc. Southeastcon,* vol. 2, pp. 623-626, 1989.

[22] R. C. Gonzalez and P. Wintz, *Digital Image Processing,* Addison Wesley, Reading, MA, 1977.

[23] M. K. Hu, "Pattern Recognition by Moment Invariants," *Proc. IRE 49,* pp. 1428, 1961.

[24] M. K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Inf. Theory IT-8,* pp. 179-187, 1962.

[25] G. B. Thomas Jr, *Calculus and Analytic Geometry,* Addison-Wesley Publishing Company, Inc., 1972.

[26] G. L. Cash and M. Hatamian, "Optical Character Recognition by Method of Moments," *Proc. Computer vision, Graphics, and Image Processing 39,* pp. 291-310, 1987.

APPENDIX A

The figures that follow next are the captured and thinned images of characters ALPHA, BETA, MU, KSI, and PSI respectively.

The thinned arrays are obtained succeeding the preprocessing step which determines the rectangular enclosure of the characters.

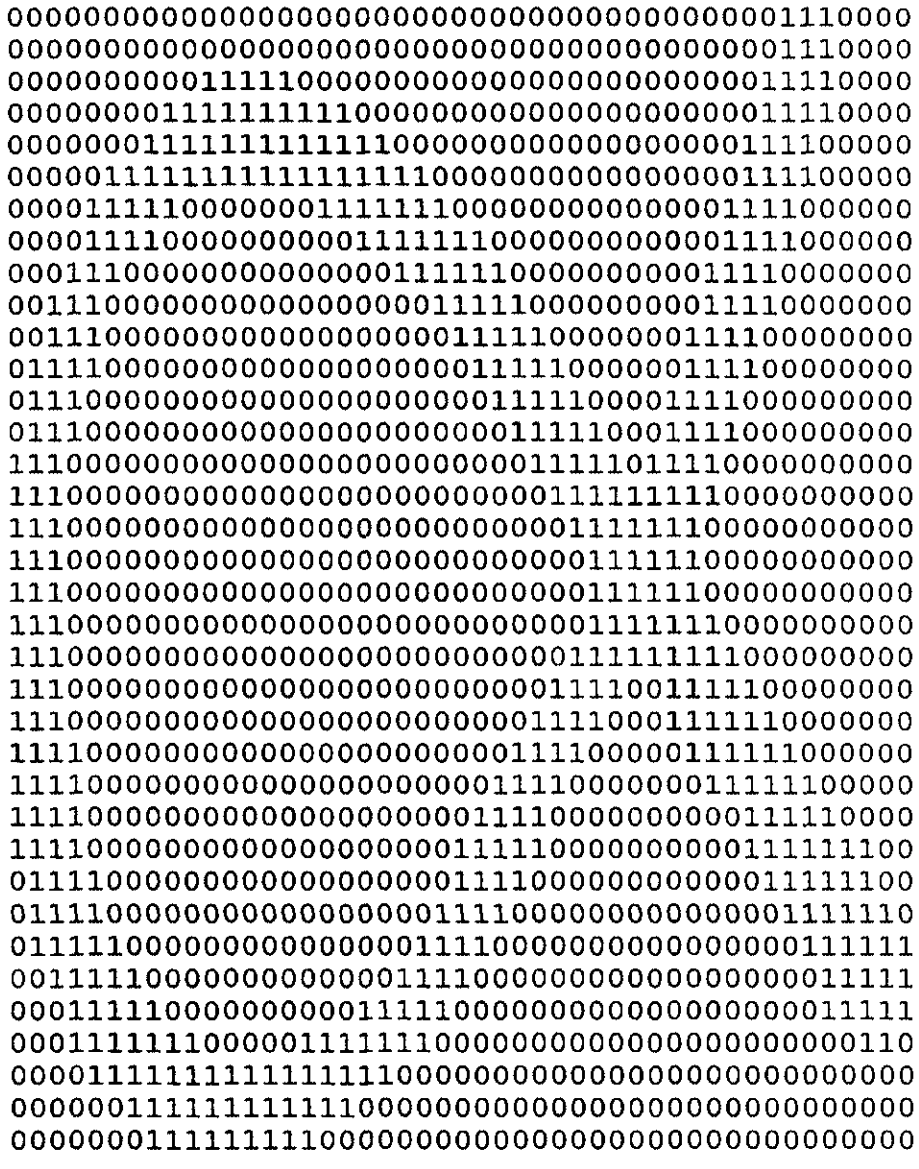


Figure A.1. Captured image of character ALPHA.

```
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000001000000
00000000000000000000000000000000000000000000010000000
00000000001111111000000000000000000000000010000000
00000000110000001100000000000000000000000100000000
00000000110000000001100000000000000000000100000000
00000011000000000000010000000000000000001000000000
00000110000000000000011000000000000000001000000000
00000100000000000000000100000000000000010000000000
00001000000000000000000100000000000000100000000000
00001000000000000000000010000000000001000000000000
00010000000000000000000001000000000010000000000000
00010000000000000000000000100000001000000000000000
00100000000000000000000000010000010000000000000000
00100000000000000000000000001000010000000000000000
00100000000000000000000000000111110000000000000000
01000000000000000000000000000011000000000000000000
01000000000000000000000000000111000000000000000000
01000000000000000000000000000100100000000000000000
01000000000000000000000000000100010000000000000000
01000000000000000000000000000100011000000000000000
01000000000000000000000000000110000110000000000000
01000000000000000000000000000110000011000000000000
01000000000000000000000000000110000000010000000000
00100000000000000000000000001100000000001100000000
00100000000000000000000000001000000000000110000000
00100000000000000000000000001000000000000011000000
00110000000000000000000001000000000000000000100000
00010000000000000000000010000000000000000000100000
00001000000000000000001100000000000000000000011000
00001100000000000000011000000000000000000000000000
00000110000000000000111000000000000000000000000000
00000011000000000011100000000000000000000000000000
00000001111111111100000000000000000000000000000000
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
```

Figure A.2. Thinned image array of character ALPHA.

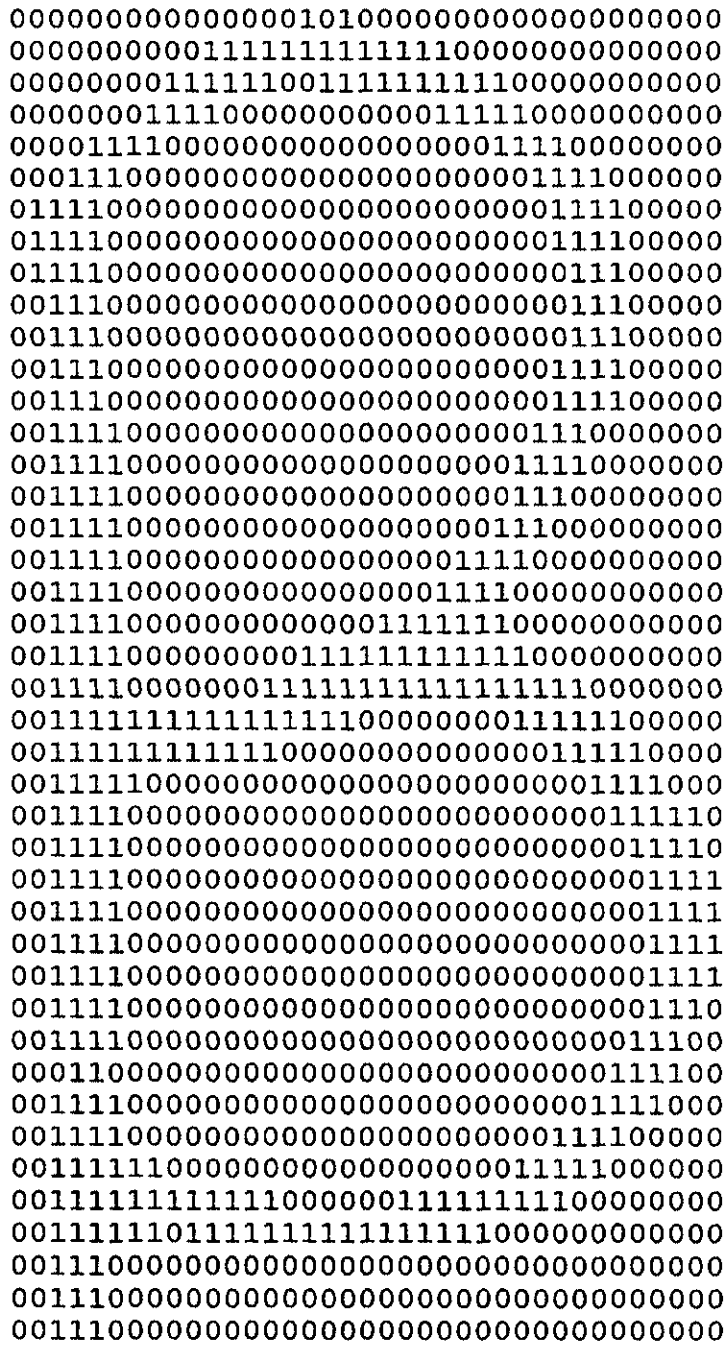


Figure A.3. Captured image of character BETA.

```
00000000000000000000000000000000000000
00000000000000000000000000000000000000
00000000000011111111111000000000000000
00000000001100000000000111000000000000
00000000100000000000000011000000000000
00000001100000000000000001100000000000
00000011000000000000000000110000000000
00000110000000000000000000011000000000
00001100000000000000000000001000000000
00001000000000000000000000001000000000
00001000000000000000000000010000000000
00010000000000000000000000010000000000
00010000000000000000000000100000000000
00010000000000000000000001000000000000
00010000000000000000000010000000000000
00010000000000000000000100000000000000
00010000000000000000001100000000000000
00010000000000000000111000000000000000
00010000000000001111000110000000000000
00010000000000111000000001110000000000
00011111111111100000000000001110000000
00010000000000000000000000000011000000
00010000000000000000000000000001100000
00010000000000000000000000000000110000
00010000000000000000000000000000010000
00010000000000000000000000000000010000
00010000000000000000000000000000010000
00010000000000000000000000000000010000
00010000000000000000000000000000010000
00001000000000000000000000000000010000
00001000000000000000000000000000010000
00001000000000000000000000000001000000
00001000000000000000000000000110000000
00001110000000000000000000011000000000
00011001111111000000001111110000000000
00010000000001111111100000000000000000
00010000000000000000000000000000000000
00010000000000000000000000000000000000
```

Figure A.4. Thinned image array of character BETA.

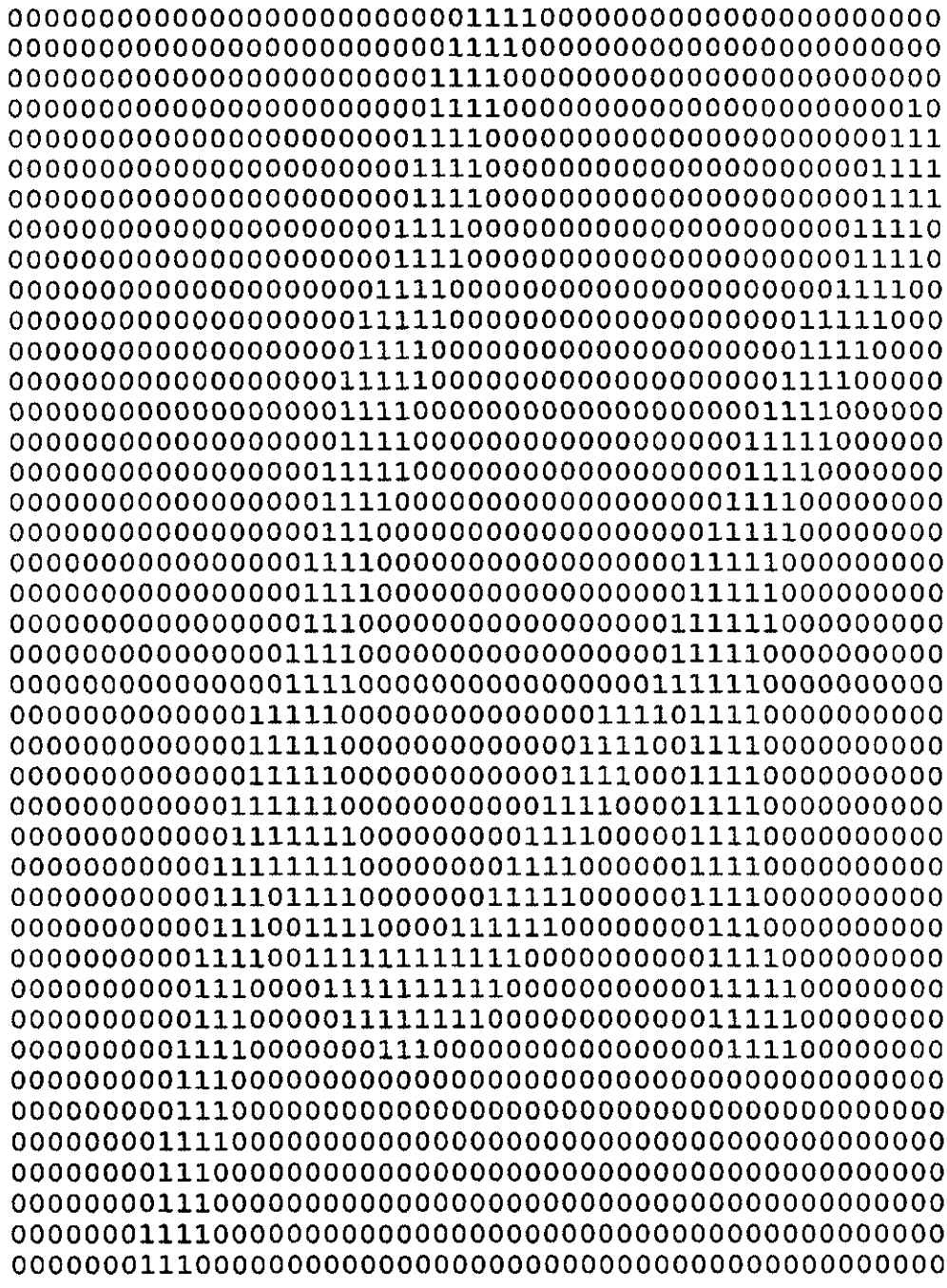


Figure A.5. Captured image of character MU.

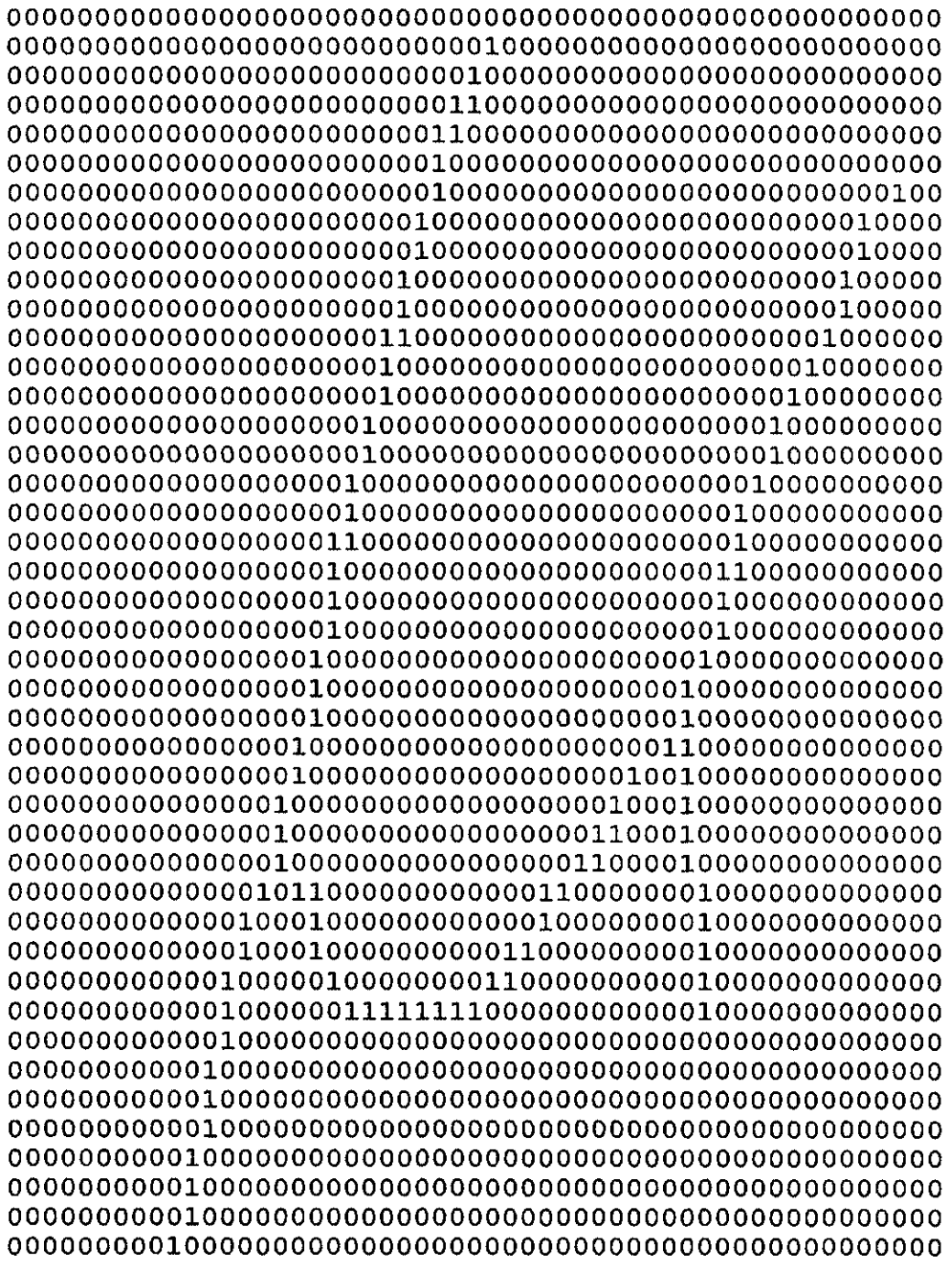


Figure A.6. Thinned image array of character MU.

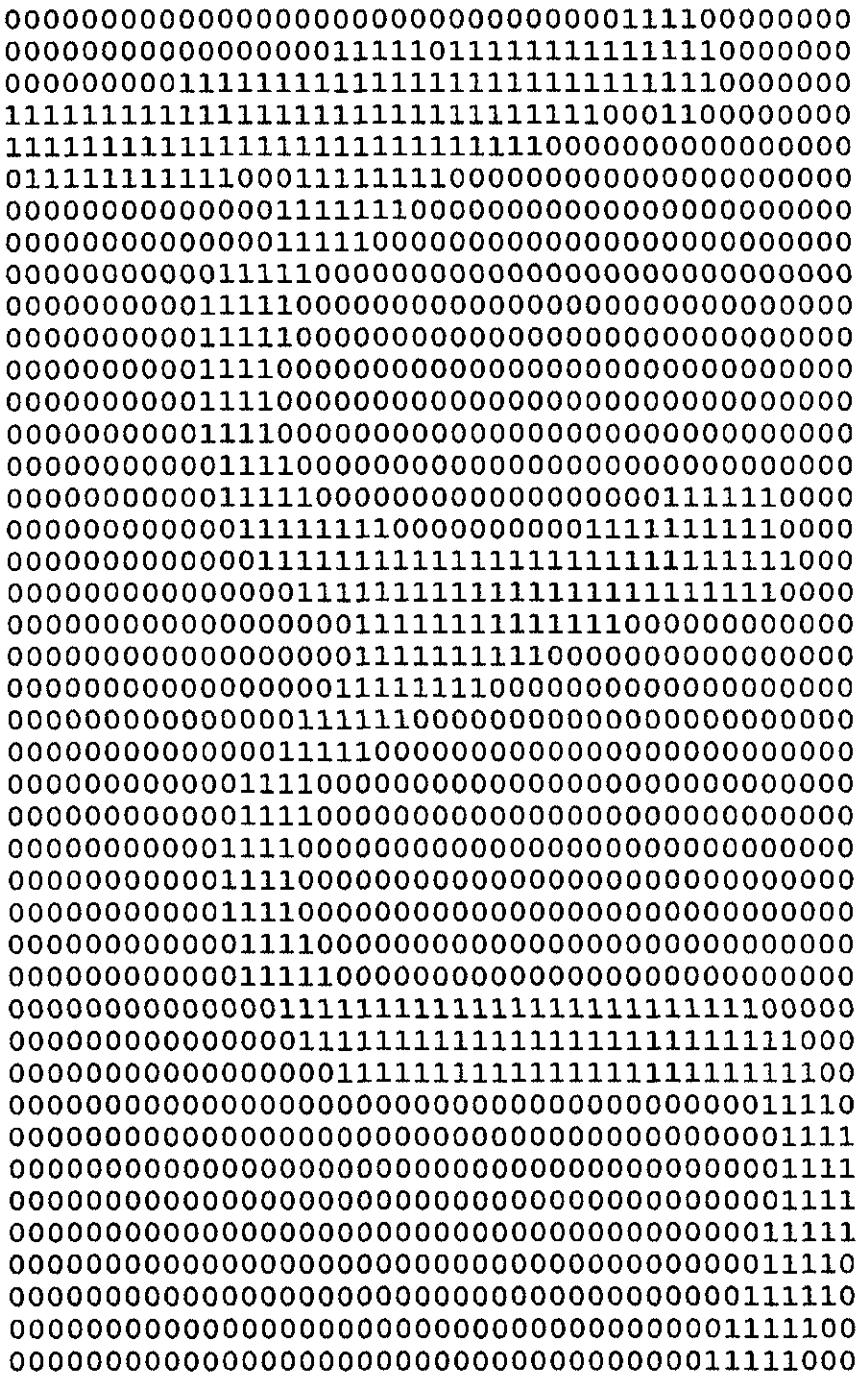


Figure A.7. Captured image of character KSI.

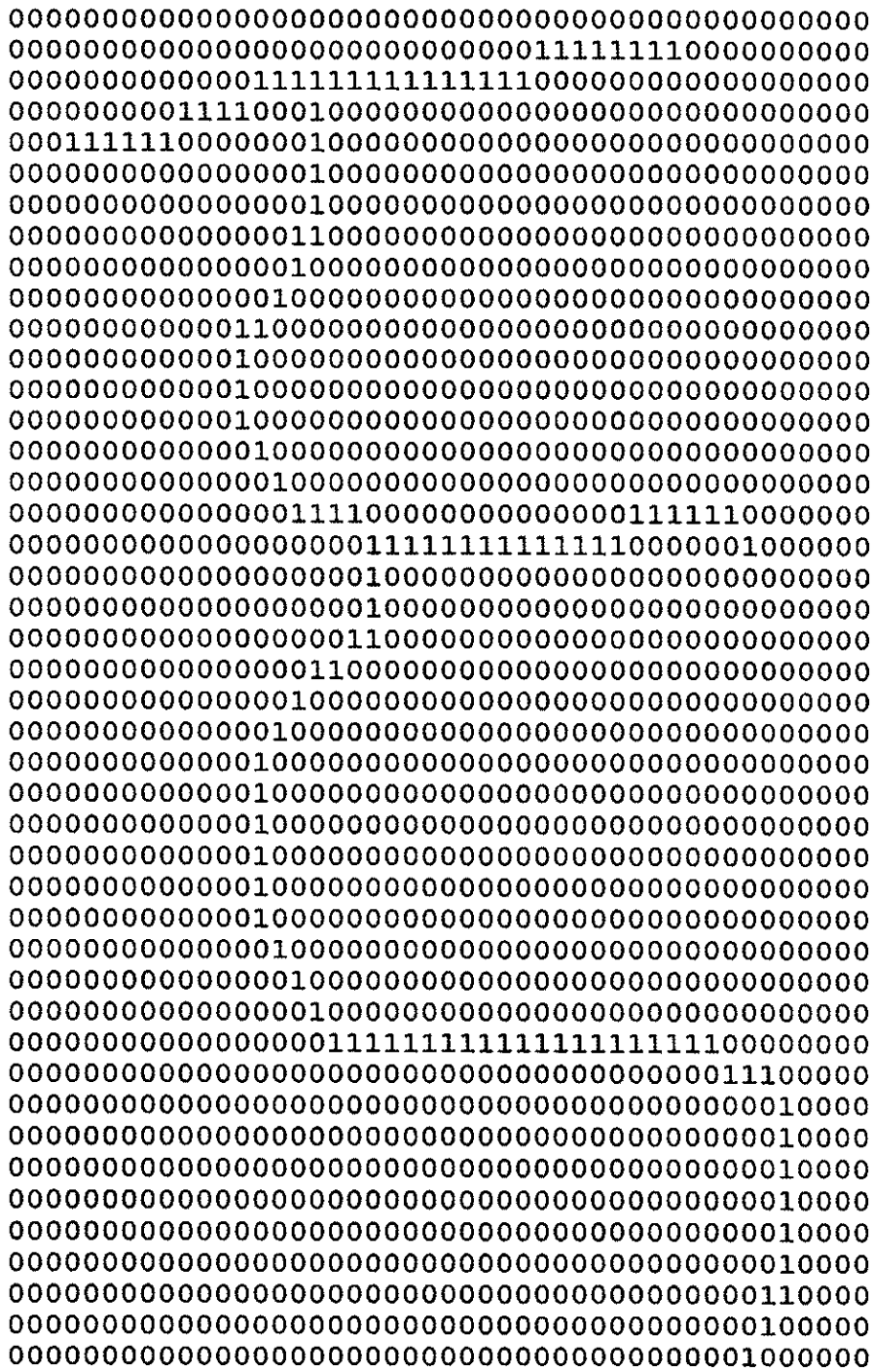


Figure A.8. Thinned image array of character KSI.

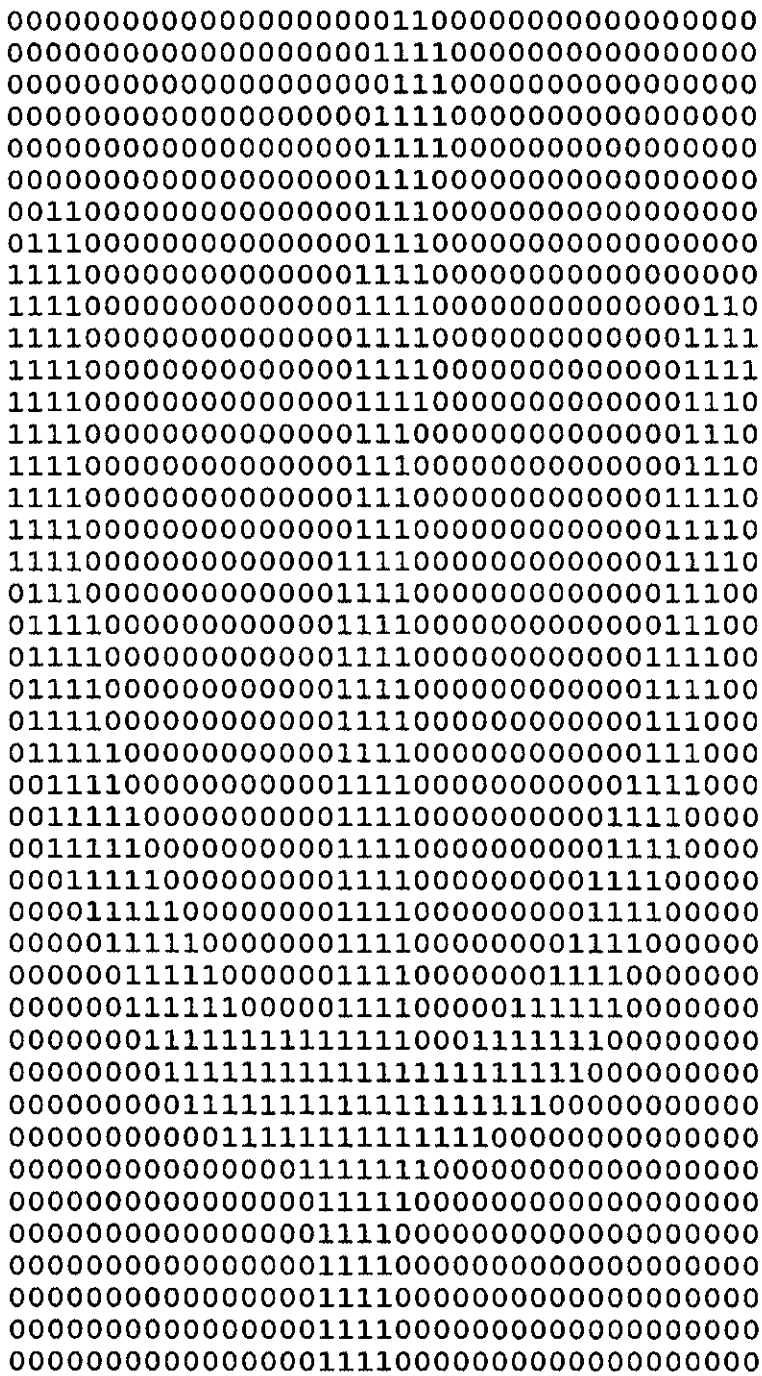


Figure A.9. Captured image of character PSI.

```
00000000000000000000000000000000000000000
00000000000000000000100000000000000000000
00000000000000000000100000000000000000000
00000000000000000000100000000000000000000
00000000000000000000100000000000000000000
00000000000000000000100000000000000000000
00000000000000000000100000000000000000000
00000000000000000000100000000000000000000
00000000000000000000100000000000000000000
00010000000000000000100000000000000000000
00100000000000000000100000000000000000000
00100000000000000000100000000000000000000
01000000000000000000100000000000000000000
01000000000000000000100000000000000010000
01000000000000000000100000000000000010000
01000000000000000000100000000000000010000
00100000000000000000100000000000000010000
00100000000000000000100000000000000010000
00100000000000000000100000000000000100000
00100000000000000000100000000000000100000
00100000000000000000100000000000000100000
00100000000000000000100000000000000100000
00010000000000000000100000000000001100000
00010000000000000000100000000000001000000
00010000000000000000100000000000001000000
00010000000000000000100000000000001000000
00001000000000000000100000000000001000000
00001000000000000000100000000000010000000
00001000000000000000100000000000010000000
00000100000000000000100000000000110000000
00000110000000000001000000000000100000000
00000010000000000001000000000000100000000
00000011000000000001000000000000100000000
00000001100000000010000000000010000000000
00000000110000000100000000000110000000000
00000000011111111100000000110000000000000
00000000000000000111111111100000000000000
00000000000000000100000000000000000000000
00000000000000000100000000000000000000000
00000000000000000100000000000000000000000
00000000000000000100000000000000000000000
00000000000000000100000000000000000000000
00000000000000000100000000000000000000000
00000000000000000100000000000000000000000
```

Figure A.10. Thinned image array of character PSI.

APPENDIX B

The programs in the following few pages are executed in the order they appear in the listing.

The program "Character array formation" converts the captured 256 x 256 image of the character after having performed thresholding, isolates the background and retains the original size of the character. The second program "Thinning" operates on the output array of the last program and yields the thinned image of the character in the file "Test.tin". The third program "Features selection" operates on the character array "Test.tin" and determines the classification features, namely, number of enclosed areas, number of intersections, and number and location of the free ends. The fourth program "Moments" evaluates the seven invariant moments of each of the test characters, and finally the fifth and last program "Classifier" utilizes the results obtained from the third and fourth programs to classify each of the input test character.

PROGRAM LISTINGS

```
C*************************************************************
C*****  PROGRAM  CHARACTER ARRAY FORMATION
C*****  THIS PROGRAM CONVERTS THE 256X256 IMAGE FILE INTO A
C*****  INTEGER FORMAT.
C*****  IN ORDER TO CONVERT IN A BINARY FILE, ALL INTENSITY
C*****  VALUES LESS THAN 100 AND GREATER THAN 0 ARE
C*****  ASSIGNED THE VALUE 1 AND THE REST ARE ASSIGNED 0.
C*****  THE RESULTING ARRAY IS STORED IN "TEST.BIN"
C*************************************************************

      DIMENSION
      CHARACTER*1 BIT_BYTE(256)
      INTEGER*2   SETUP(256)
      INTEGER*2   VALUES(256,256),VAL_UES(256,256),SUM
      INTEGER*2   Y(2560),V
      INTEGER*2   ABS_LUT(2560),SUM1(2560),C(256,256)
      INTEGER     IINDEX(5000),JINDEX(5000),K,Z,MAX,MIN
      INTEGER     COUNT,ST(2560),T,NNEW,MNEW,LOCATE1(20),Q
      INTEGER     LOCATE2(20)
      INTEGER *2  D(0:256,0:256),MEAN1,MEAN2,MEAN3
      INTEGER     A(256,256)
      INTEGER     UPPER,LOWER,P,LEFT,RIGHT,LOCATE3(20)
      INTEGER *2  AREA(300,300),M2
      INTEGER     P1,P2,P3,P4,P5,P6,P7,P8,P9,IIINDEX(5000),
      INTEGER     Y1,W,LEFT1
      INTEGER     D1,D2,D3,D4,D5,D6,D7,D8,P11,P22,SUM2,G,
      INTEGER     JJINDEX(5000)
      INTEGER     F,NN,ONES,RIGHT1
      INTEGER*2   IMG(300,300),TEST(300,300),REST(300,300)
      INTEGER     LT, COUNT1
      INTEGER     K1,K2,YA,SMLST(5000),NF,TN(7000)
      INTEGER*2   FINAL(5000)
      REAL*8      M_00, M_01, M_02, M_03, M_10, M_11
      REAL*8      M_12, M_13, M_20, M_21, M_22, M_23
      REAL*8      M_30, M_31, M_32, M_33,I,J
      REAL        X_BAR, Y_BAR,N,M
      REAL        MU_00,MU_11,MU_01,MU_02,MU_03,MU_10,MU_12
      REAL        MU_13,MU_20,MU_21,MU_22,MU_23,MU_30,MU_31
      REAL        MU_32,MU_33
      REAL        ETA_00, ETA_11, ETA_12, ETA_13, ETA_10,
      REAL        ETA_20
      REAL        ETA_21, ETA_22, ETA_23, ETA_30, ETA_31,
```

```
      REAL         ETA_32
      REAL         ETA_33, ETA_01, ETA_03, ETA_02
      REAL         PHI_1, PHI_2, PHI_3, PHI_4, PHI_5,
      REAL         PHI_6,PHI_7
      REAL         MIMI.MM1,MM2,MM3
      REAL         MM4,MM5,MM6,MM7,NORM,MSE1,MSE2
      REAL         NMIN,NMM1,NMM2,NMM3,NMM4,NMM5,NMM6,NMM7
      REAL        MSE3,MSE4,MSE5,MSE6,MSE7,MSE8,MSE9,MSE10
      REAL        MSE11,MSE12,MSE13,MSE14,MSE15,MSE16,MSE17

      OPEN(UNIT=1,FILE='TEST.PIP',ACCESS='DIRECT',STATUS='old')
      OPEN(UNIT=2,FILE='TEST.INT',RECL=1024,STATUS='UNKNOWN')
      OPEN(UNIT=4,FILE='TEST.NOR',RECL=1024,STATUS='UNKNOWN')
      OPEN(UNIT=8,FILE='TEST.MON',STATUS='UNKNOWN')
      OPEN(UNIT=9,FILE='TEST.TIN',RECL=1024,STATUS='UNKNOWN')
      OPEN(UNIT=16,FILE='TEST.MOM',STATUS='UNKNOWN')

      N=256
      M=256
      SUM=0

      DO 10 I=3,N
            K=I

      READ(1,REC=K) (BIT_BYTE(J),J=1,M)
            DO 20 J=1,M
                  VALUES(I,J)=ICHAR(BIT_BYTE(J))
16                IF (J.LE.10)THEN
6     GOTO 210
      ELSE
      IF ((VALUES(I,J).GE.1).AND.(VALUES(I,J).LE.100))THEN
      VAL_UES(I,J)=1
      ELSE

210   VAL_UES(I,J)=0

      ENDIF
      ENDIF
20          CONTINUE
10    CONTINUE
110   FORMAT(256I4)
```

```
C*************************************************************
C*****  THIS PROGRAM OPERATES ON THE BINARY IMAGE TEST.BIN.
C*****  THREE SETS OF OUTPUTS ARE OBTAINED. THE FIRST OUTPUT
C*****  IS AN ARRAY CONSISTING OF THE NUMBER AND THE
C*****  LOCATION OF THE ROWS HAVING A PIXEL OF VALUE 1. THE
C*****  SECOND OUTPUT CONSISTS OF ALL THE ROWS OF THE PIXELS
C*****  WITH VALUE ONE WITH ALL THE 256 ELEMENTS. THE SIZE
C*****  OF THE REDUCED IMAGE IS ALSO DISPLAYED ON THE
C*****  SCREEN.  THE LAST OUTPUTCONSISTS OF THE ARRAY GIVING
C*****  THE SUM OF ALL THE ONES IN THE 256 RECORDS.
C*************************************************************

      N=254
      M=256
      COUNT=0

      K=1
      DO 40 I=1,N
            SUM1(I)=0
            DO 50 J=1,M
                  SUM1(I)=SUM1(I)+VAL_UES(I,J)
50          CONTINUE
            IF (SUM1(I).NE.0) THEN
            COUNT=COUNT+1
            ST(K)=I
            K=K+1
            WRITE(2,198)(VAL_UES(I,J),J=1,M)
            ENDIF
198   FORMAT(256I4)
40    CONTINUE
      CLOSE(2)

      N=COUNT
            WRITE(*,*)N,M
```

```
C*************************************************************
C*****  THIS PROGRAM OPERATES ON THE REDUCED BINARY IMAGE
C*****  TEST.INT".
C*****  FOR EACH OF THE NON-ZERO ROWS, THE MINIMUM AND THE
C*****  MAXIMUM LOCATIONS OF THE ONES IS OBTAINED. BASED
C*****  UPON THESE OBSERVATIONS A FINAL REDUCTION OF THE
C*****  IMAGE ARRAY FROM  256X256 TO THE NEW SIZE IS MADE.
C*****  THE OUTPUT FILE "TEST.NOR" IS NEXT MADE TO UNDERGO
C*****  THINNING.
C*************************************************************

      OPEN(UNIT=3,FILE='TEST.INT',STATUS='UNKNOWN')

      N=COUNT
      M=256

      WRITE(*,*)N,COUNT
      IINDEX(K)=0
      JINDEX(K)=0
      K=1
      DO 60 I=1,N
            READ(3,120)(VAL_UES(I,J),J=1,M)
60    CONTINUE
            DO 70 I=1,N
                  DO 80 J=1,256
                  IF (VAL_UES(I,J).EQ.1)THEN
                  IINDEX(K)=I
                  JINDEX(K)=J
                  K=K+1
                  ENDIF
80                CONTINUE
70          CONTINUE


      WRITE(*,*)K
      MAX=JINDEX(1)
      MIN=JINDEX(1)

      DO 90 Z=1,K-1

      IF(JINDEX(Z).GT.MAX)THEN
      MAX=JINDEX(Z)
      ENDIF

      IF (JINDEX(Z).LT.MIN)THEN
      MIN=JINDEX(Z)
      ENDIF
90    CONTINUE
120   FORMAT(256I4)
      WRITE(*,*)MAX,MIN
```

```
      DO 15 I=1,N
      WRITE(4,120)(VAL_UES(I,J),J=MIN,MAX)
15    CONTINUE

      M2=MAX-MIN+1
      CLOSE(4)
```

```
C*************************************************************
C*****  PROGRAM THINNING
C*****  THIS PROGRAM PERFORMS THINNING ON THE FILE
C*****  "TEST.NOR". THE CHARACTER IS THINNED TO ONE PIXEL
C*****  THICK. THE OUTPUT FILE IS "TEST.TIN". IT'S BEEN
C*****  IMPLEMENTED FROM THE BOOK "DIGITAL IMAGE PROCESSING'
C*****  BY GONZALEZ AND WINTZ.
C*************************************************************

      OPEN(UNIT=7,FILE='TEST.NOR',STATUS='UNKNOWN')
      M=M2
      N=COUNT
      WRITE(*,*)N,M

      DO 25 I=1,N
            READ(7,190)(D(I,J),J=1,m)
25    CONTINUE

190   FORMAT(256I4)

      K=1
      Z=1
      IIINDEX(Z)=0
      JJINDEX(Z)=0
      DO 35 V=1,10
      DO 45 I=1,N
            DO 55 J=1,M
            FLAG=0
            G=0
            IF(D(I,J).EQ.1)THEN
            SUM2=D(I-1,J-1)+D(I-1,J)+D(I-1,J+1)+D(I,J-1)
            +D(I,J+1)+D(I+1,J-1)+D(I+1,J)+D(I+1,J+1)
            P1=D(I,J)
            P2=D(I-1,J)
            P3=D(I-1,J+1)
            P4=D(I,J+1)
            P5=D(I+1,J+1)
            P6=D(I+1,J)
            P7=D(I+1,J-1)
            P8=D(I,J-1)
            P9=D(I-1,J-1)

            D1=P3-P2
            D2=P4-P3
            D3=P5-P4
            D4=P6-P5
            D5=P7-P6
            D6=P8-P7
            D7=P9-P8
            D8=P2-P9
```

```
IF (SUM.LT.8)THEN
IF((SUM.LE.6).AND.(SUM.GE.2))THEN
IF(D1.EQ.1)THEN
G=G+1
ENDIF
IF(D2.EQ.1)THEN
G=G+1
ENDIF
IF(D3.EQ.1)THEN
G=G+1
ENDIF
IF(D4.EQ.1)THEN
G=G+1
ENDIF
IF(D5.EQ.1)THEN
G=G+1
ENDIF
IF(D6.EQ.1)THEN
G=G+1
ENDIF
IF(D7.EQ.1)THEN
G=G+1
ENDIF
IF (D8.EQ.1)THEN
G=G+1
ENDIF

P11=P2*P4*P6


IF (G.EQ.1)THEN
IF(P11.EQ.0)THEN
P22=P4*P6*P8
IF(P22.EQ.0)THEN
FLAG=1
IIINDEX(Z)=I
JJINDEX(Z)=J
Z=Z+1
ELSE
D(I,J)=D(I,J)
ENDIF
ELSE
D(I,J)=D(I,J)
ENDIF
ELSE
D(I,J)=D(I,J)
ENDIF
ELSE
D(I,J)=D(I,J)
ENDIF
```

```
            ELSE
            D(I,J)=D(I,J)
           ENDIF
            ENDIF
55          CONTINUE
45          CONTINUE


            DO 65 Y1=1,Z-1
            D(IIINDEX(Y1),JJINDEX(Y1))=0
65          CONTINUE
            Z=1
            IIINDEX(Z)=0
            JJINDEX(Z)=0

            DO 75 I=1,N
            DO 85 J=1,M
            FLAG=0
            G=0
            IF(D(I,J).EQ.1)THEN
            SUM=D(I-1,J-1)+D(I-1,J)+D(I-1,J+1)+D(I,J-1)+
                D(I,J+1)+ D(I+1,J-1)+D(I+1,J)+D(I+1,J+1)

            P1=D(I,J)
            P2=D(I-1,J)
            P3=D(I-1,J+1)
            P4=D(I,J+1)
            P5=D(I+1,J+1)
            P6=D(I+1,J)
            P7=D(I+1,J-1)
            P8=D(I,J-1)
            P9=D(I-1,J-1)

            D1=P3-P2
            D2=P4-P3
            D3=P5-P4
            D4=P6-P5
            D5=P7-P6
            D6=P8-P7
            D7=P9-P8
            D8=P2-P9

            IF (SUM.LT.8)THEN
            IF((SUM.LE.6).AND.(SUM.GE.2))THEN
            IF(D1.EQ.1)THEN

            G=G+1
            ENDIF
            IF(D2.EQ.1)THEN
            G=G+1
            ENDIF
            IF(D3.EQ.1)THEN
```

```
                G=G+1
                ENDIF
                IF(D4.EQ.1)THEN
                G=G+1
                ENDIF
                IF(D5.EQ.1)THEN
                G=G+1
                ENDIF
                IF(D6.EQ.1)THEN
                G=G+1
                ENDIF
                IF(D7.EQ.1)THEN
                G=G+1
                ENDIF
                IF(D8.EQ.1)THEN
                G=G+1
                ENDIF

                P11=P2*P4*P6
                IF (G.EQ.1)THEN
                IF(P11.EQ.0)THEN
                P22=P4*P6*P8
                IF(P22.EQ.0)THEN
                FLAG=1
                IIINDEX(Z)=I
                JJINDEX(Z)=J
                Z=Z+1
                ELSE
                D(I,J)=D(I,J)
                ENDIF
                ELSE
                D(I,J)=D(I,J)
                ENDIF



                ELSE
                D(I,J)=D(I,J)
                ENDIF
                ELSE
                D(I,J)=D(I,J)
                ENDIF
                ELSE
                D(I,J)=D(I,J)
                ENDIF
                ENDIF
85              CONTINUE
75              CONTINUE

                DO 95 Y1=1,Z-1
                D(IIINDEX(Y1),JJINDEX(Y1))=0
95              CONTINUE
```

```
35          CONTINUE
            NNEW=N+2
            MNEW=M+2
            DO 105 I=1,N+2


            DO 115 J=1,M+2
            AREA(I,J)=0
115         CONTINUE
105         CONTINUE
            DO 125 I=1,N
            DO 135 J=1,M
            AREA(I+1,J+1)=D(I,J)
135         CONTINUE
125         CONTINUE



            DO 145 I=1,NNEW
            WRITE(8,130)(AREA(I,J),J=1,MNEW)
            WRITE(9,180)(AREA(I,J),J=1,MNEW)


145         CONTINUE
130         FORMAT(256I1)
180         FORMAT(256I4)
            N=NNEW
            M=MNEW


            CLOSE(7)
            CLOSE(9)
```

```
C*************************************************************
C*****  PROGRAM FEATURES SELECTION
C*****  THIS PROGRAM CHECKS FOR INTERSECTIONS AND FREE ENDS
C*****  IN CHARACTER. FOR EACH PIXEL OF VALUE ONE THE NUMBER
C*****  OF NEIGHBORS WITH VALUE ONE ARE DETERMINED. WHEN
C*****  THIS VALUE EQUALS OR EXCEEDS FOUR, A INTERSECTION
C*****  EXISTS. THE VALUE OF NN GIVES THE NUMBER OF
C*****  INTERSECTIONS IN THE CHARACTER ARRAY. A SIMILAR
C*****  APPRAOCH IS FOLLOWED WHILE DETECTING THE
C*****  FREE ENDS. THE NUBER OF PIXELDS OF VALUE ONE WHICH
C*****  HAS JUST ONE NEIGHBOR OF VALUE ONE AT ITS EIGHT
C*****  NEIGHBORING SITES ARE THE FREE ENDS OF THE
C*****  CHARACTER. THE INPUT FILE IS "TEST.TIN".
C*****  NN AND ONES DENOTE THE NUMBER OF INTERSECTIONS AND THE
C*****  NUMBER OF FREE ENDS.
C*************************************************************
      OPEN(UNIT=10,FILE='TEST.TIN',STATUS='UNKNOWN')
      N=NNEW
      M=MNEW
      NN=0
      ONES=0
      DO 295      I=1,N
            READ(10,180)(AREA(I,J),J=1,M)
295   CONTINUE

      MEAN1=(ININT(N/2))
      LOCATE1(Q)=0
      Q=1

      MEAN2=(ININT(M/2))
      WRITE(*,*)MEAN2
      LOCATE2(P)=0
      P=1

      MEAN3=(ININT(M/4))
      LOCATE3(W)=0

      W=1
      DO 300 I=1,n
            DO 310 J=1,m
            F=0
            IF (AREA(I,J) .EQ. 1) THEN
            IF (AREA(I-1,J-1).EQ.1)THEN
            F=F+1
            ENDIF
            IF (AREA(I-1,J+1).EQ.1)THEN
            F=F+1
            ENDIF
            IF (AREA(I+1,J-1).EQ.1)THEN
            F=F+1
            ENDIF
```

```
            IF (AREA(I+1,J+1).EQ.1)THEN
            F=F+1
            ENDIF
            IF (AREA(I-1,J).EQ.1)THEN
            F=F+1
            ENDIF
            IF (AREA(I+1,J).EQ.1)THEN
            F=F+1
            ENDIF
            IF (AREA(I,J-1).EQ.1)THEN
            F=F+1
            ENDIF
            IF (AREA(I,J+1).EQ.1)THEN
            F=F+1
            ENDIF
            IF (F.EQ.1)THEN
            ONES=ONES+1
            LOCATE1(Q)=I
            Q=Q+1

            LOCATE2(P)=J
            P=P+1
            LOCATE3(W)=J
            W=W+1
            ENDIF
            IF (F.GE.4)THEN

            NN=NN+1
            ENDIF
            ENDIF
310         CONTINUE
300         CONTINUE

            UPPER=0
            LOWER=0
            DO 306 I=1,Q-1
            IF (LOCATE1(I).LT.MEAN1)THEN
            UPPER=UPPER+1
            ELSE
            LOWER=LOWER+1
            ENDIF
306         CONTINUE

            LEFT=0
            RIGHT=0
            DO 406 I=1,P-1
            IF (LOCATE2(I).LT.MEAN2)THEN
            LEFT=LEFT+1
            ELSE
            RIGHT=RIGHT+1
            ENDIF
406         CONTINUE
```

```
            LEFT1=0
            RIGHT1=0


            DO 4060 I=1,W-1
            IF (LOCATE3(I).LT.MEAN3)THEN
            LEFT1=LEFT1+1
            ELSE
            RIGHT1=RIGHT1+1
            ENDIF
4060        CONTINUE



      WRITE(*,*)NN,ONES
      TYPE*,('THE NUMBER OF FREE ENDS ABOVE THE MEAN ARE:')
      WRITE(*,*)UPPER
      TYPE*,('THE NUMBER OF FREE ENDS BELOW THE MEAN ARE: ')

      WRITE(*,*)LOWER
      TYPE*,('THE NUMBER OF FREE ENDS LEFT OF THE MEAN ARE:')
      WRITE(*,*)LEFT
      TYPE*,('THE NUMBER OF FREE ENDS RIGHT OF THE MEAN ARE:')
      WRITE(*,*)RIGHT
      TYPE*,('THE NUMBER OF FREE ENDS LEFT OF MEAN3 ARE:')
      WRITE(*,*)LEFT1
      TYPE*,('THE NUMBER OF FREE ENDS RIGHT OF MEAN3 ARE: ')
      WRITE(*,*)RIGHT1
      CLOSE(10)
```

```
C*************************************************************
C*****  THIS PROGRAM DETERMINES IF A PARTICULAR CHARACTER
C*****  HAS ANY ENCLOSED AREA OR NOT. IN CASE THERE'S SOME
C*****  AREA ENCLOSED, THE NUMBER OF THESE AREAS IS ALSO
C*****  DETERMINED. THE INPUT FILE IS "TEST.TIN'.
C*************************************************************

      OPEN(UNIT=11,FILE='TEST.TIN',STATUS='OLD')
      OPEN(UNIT=12,FILE='TEST.ARE',RECL=1024,STATUS='UNKNOWN')

      BGST=1
      FLAG=0
      DO 400 I=1,N
            READ(11,180)(IMG(I,J),J=1,M)
400   CONTINUE
      DO 567 I=1,N
            DO 765 J=1,M
            IF(IMG(I,J).EQ.1)THEN
            TEST(I,J)=2000
            ENDIF
765         CONTINUE
567         CONTINUE
            K=1
            DO 13 I=1,1
            DO 113 J=1,M
            TEST(I,J)=1
113         CONTINUE
13          CONTINUE

            DO 420 I=2,N
                  DO 430 J=1,M
                  IF (IMG(I,J).EQ.0)THEN
                  IF((I-1).GE.1)THEN
                  IF(IMG(I-1,J).EQ.0)THEN
                  TN(K)=TEST(I-1,J)
                  K=K+1
                  GOTO 31
                  ELSE
            GOTO 31
            ENDIF
31          IF((J-1).GE.1) THEN
            IF(IMG(I,J-1).EQ.0)THEN
            TN(K)=TEST(I,J-1)
            IF(K.EQ.1)THEN


            TEST(I,J)=TN(K)
            ELSE
            IF (TN(2).GE.TN(1))THEN
            TEST(I,J)=TN(1)
            ELSE
```

```
                TEST(I,J)=TN(2)
                ENDIF
                ENDIF
                ELSE
                TEST(I,J)=TN(K-1)
                ENDIF
                ELSE
                TEST(I,J)=TN(K-1)
                ENDIF

                BGST=BGST+1
                TEST(I,J)=BGST
                ENDIF
                ENDIF
430             CONTINUE
420             CONTINUE
                DO 234 I=2,N
                      DO 965 J=1,M
                      IF ((I-1).GE.1)THEN
                      IF (TEST(I,J).NE.2000)THEN
                      IF (TEST(I-1,J).NE.2000)THEN
                      IF(TEST(I-1,J).LT.TEST(I,J))THEN
                      TEST(I,J)=TEST(I-1,J)
                      ELSE
                      TEST(I,J)=TEST(I,J)
                      ENDIF
                      ENDIF
                ENDIF
                ENDIF
965                   CONTINUE
234             CONTINUE

                DO 334 I=2,N
                      DO 665 J=1,M
                      IF ((J-1).GE.1)THEN
                      IF (TEST(I,J).NE.2000)THEN
                      IF (TEST(I,J-1).NE.2000)THEN
                      IF(TEST(I,J-1).LE.TEST(I,J))THEN
                      TEST(I,J)=TEST(I,J-1)
                      ELSE
                      TEST(I,J)=TEST(I,J)
                      ENDIF
                      ENDIF
                      ENDIF
                      ENDIF
665                   CONTINUE
334             CONTINUE


                LST=1
                DO 919 I=2,N
                DO 1919 J=1,M
```

```
            IF (TEST(I,J).NE.2000)THEN
            IF (TEST(I,J).GT.LST)THEN
            TEST(I,J)=LST+1
            LST=TEST(I,J)
            ELSE
            TEST(I,J)=TEST(I,J)
            ENDIF
            ENDIF

1919        CONTINUE
919         CONTINUE


            DO 3000 I=1,N
            WRITE(12,180)(TEST(I,J),J=1,M)
3000        CONTINUE
            CLOSE(12)
      OPEN(UNIT=13,FILE='TEST.ARE',STATUS='UNKNOWN')
      OPEN(UNIT=18,FILE='TEST.DEL',RECL=1024,STATUS='UNKNOWN')

            DO 123 I=1,N
            READ(13,180)(REST(I,J),J=1,M)
123         CONTINUE



            DO 690 V=1,M
            DO 444 I=1,N
            DO 555 J=1,M
            IF ((REST(I,J).NE.1).AND.(REST(I,J).NE.2000))THEN
            IF((REST(I,J+1).NE.1).AND.(REST(I,J+1).NE.2000))THEN
            IF (REST(I,J).LE.REST(I,J+1))THEN
            REST(I,J+1)=REST(I,J)
            ENDIF
            ENDIF
            ENDIF
555         CONTINUE
444         CONTINUE

            DO 1444 I=1,N
            DO 1555 J=1,M
            IF(REST(I,J).NE.2000)THEN
            IF((REST(I,J-1).NE.1).AND.(REST(I,J-1).NE.2000))THEN
            IF (REST(I,J).LE.REST(I,J-1))THEN
            REST(I,J-1)=REST(I,J)
            ENDIF
            ENDIF
            ENDIF
1555        CONTINUE
1444        CONTINUE
```

```
            DO 2444 I=1,N
            DO 2555 J=1,M
            IF(REST(I,J).NE.2000)THEN
            IF((REST(I-1,J).NE.1).AND.(REST(I-1,J).NE.2000))THEN
            IF (REST(I,J).LE.REST(I-1,J))THEN
            REST(I-1,J)=REST(I,J)
            ENDIF
            ENDIF
            ENDIF
2555        CONTINUE
2444        CONTINUE
            DO 24440 I=1,N
            DO 25550 J=1,M

            IF(REST(I,J).NE.2000)THEN
            IF (REST(I+1,J).EQ.1)THEN
            IF (REST(I,J).GE.REST(I+1,J))THEN
            REST(I,J)=REST(I+1,J)
            ENDIF
            ENDIF
            ENDIF
25550 CONTINUE
24440 CONTINUE

            DO 2443 I=1,N
            DO 2553 J=1,M
            IF(REST(I,J).NE.2000)THEN
            IF (REST(I,J-1).EQ.1)THEN

            IF (REST(I,J).GE.REST(I,J-1))THEN
            REST(I,J)=REST(I,J-1)


            ENDIF
            ENDIF
            ENDIF
2553        CONTINUE
2443        CONTINUE


            DO 3444 I=1,N
            DO 3555 J=1,M
            IF(REST(I,J).NE.2000)THEN
            IF((REST(I+1,J).NE.1).AND.(REST(I+1,J).NE.2000))THEN
            IF (REST(I,J).LE.REST(I+1,J))THEN
            REST(I+1,J)=REST(I,J)


            ENDIF
            ENDIF
```

```
            ENDIF
3555        CONTINUE
3444        CONTINUE

690         CONTINUE

            F=1
            COUNT1=0
            FINAL(T)=0
            DO 987 I=1,N
            DO 9870 J=1,M
            IF ((REST(I,J).NE.1).AND. (REST(I,J).NE.2000))THEN
            COUNT1=COUNT1+1
            FINAL(COUNT1)=REST(I,J)
            ENDIF
9870        CONTINUE
987         CONTINUE
777         FORMAT(I4)

            DO 598 I=1,N
            WRITE(18,1800)(REST(I,J),J=1,M)
598         CONTINUE
1800        FORMAT(256I4)



            Z=1
            SMLST(Z)=FINAL(1)

            IF (SMLST(Z).EQ.0)THEN
            GOTO 722
            ENDIF

            DO 1965 I=1,COUNT1
            IF (FINAL(I).NE.SMLST(Z))THEN
            Z=Z+1
            SMLST(Z)=FINAL(I)
            ELSE
            ENDIF
1965        CONTINUE

            NF=0
912         DO 367 I=2,Z
            LT=SMLST(1)
            IF (SMLST(I).EQ.LT)THEN
            SMLST(I)=0
            ENDIF
367         CONTINUE
            DO 3670 I=3,Z
            LT=SMLST(2)
            IF (SMLST(I).EQ.LT)THEN
            SMLST(I)=0
```

```
            ENDIF

3670        CONTINUE
            DO 456 I=1,Z
            IF (SMLST(I).NE.0)THEN
            NF=NF+1
            ENDIF

456         CONTINUE

      OPEN(UNIT=14,FILE='TEST.AR',RECL=1024,STATUS='UNKNOWN')

      DO 678 I=1,N
      WRITE(14,180)(REST(I,J),J=1,M)
678   CONTINUE
      WRITE(*,*)('THE NUMBER OF AREAS IN THE CHARACTER IS:')
      WRITE(*,*)NF
      GOTO 9089
5000  FORMAT(256i1)
722   NF=0
      WRITE(*,*)('THE NUMBER OF AREAS IN THE CHARACTER IS   ZERO')
      GOTO 9089

9089  WRITE(*,*) ('SUBPROGRAM COMPLETED')
      CLOSE(11)
```

```
C*************************************************************
C*****  PROGRAM MOMENTS
C*****  THIS PROGRAM DETERMINES THE SEVEN INVARIANT
C*****  NORMALIZED MOMENTS OF ANY TEST CHARACTER. THE INPUT
C*****  FILE IS "TEST.TIN".
C*****  THE NORMALIZED MOMENTS DETERMONED ARE MM1, MM2, MM3,
C*****  MM4, MM5, MM6, MM7.
C*************************************************************

      OPEN(15,FILE='TEST.TIN',STATUS='UNKNOWN')

            DO 700 I=1,N
            READ(15,710)(AREA(I,J),J=1,M)
700         CONTINUE
710         FORMAT(256I4)
            M_00=0
            M_01=0
            M_02=0
            M_03=0
            M_10=0
            M_11=0
            M_12=0
            M_13=0
            M_21=0
            M_22=0
            M_23=0
            M_20=0
            M_30=0
            M_31=0
            M_32=0
            M_33=0
            WRITE(*,*)N,M



            DO 720 I=1,N
            DO 730 J=1,M
            M_00= M_00+AREA(I,J)
            M_01= M_01 + J*AREA(I,J)


            M_02= M_02 + (J**2)*AREA(I,J)
            M_03= M_03 + (J**3)*AREA(I,J)
            M_10= M_10 + I*AREA(I,J)
            M_11= M_11 + I*J*AREA(I,J)
            M_12= M_12 + I*(J**2)*AREA(I,J)
            M_13= M_13 + I*(J**3)*AREA(I,J)
            M_20= M_20 + (I**2)*AREA(I,J)
            M_21= M_21 + (I**2)*J*AREA(I,J)
```

```
      M_22= M_22 + (I**2)*(J**2)*AREA(I,J)
      M_23= M_23 + (I**2)*(J**3)*AREA(I,J)
      M_30= M_30 + (I**3)*AREA(I,J)
      M_31= M_31 + (I**3)*J*AREA(I,J)
      M_32= M_32 + (I**3)*(J**2)*AREA(I,J)
      M_33= M_33 + (I**3)*(J**3)*AREA(I,J)
730   CONTINUE
720   CONTINUE
      WRITE(*,*)M_33

      X_BAR= M_10/ M_00
      Y_BAR= M_01/ M_00

      MU_00= M_00
      MU_10=0
      MU_01=0
      MU_20= M_20-(X_BAR * M_10)
      MU_02= M_02-(Y_BAR * M_01)
      MU_11= M_11 - (Y_BAR * M_10)
      MU_30= M_30 - (3*X_BAR * M_20) + (2 * M_10
     *(X_BAR**2))
      MU_12=    M_12    -    (2*Y_BAR*M_11)    -    (X_BAR*M_02)
     +(2*(Y_BAR**2)*M_10)
      MU_21= M_21 - (2*X_BAR*M_11) - (Y_BAR*M_20) +
     (2*(X_BAR**2)*M_01)
      MU_03= M_03 - (3*Y_BAR*M_02) + (2*(Y_BAR**2)*M_01)

      ETA_02= MU_02/ (MU_00**2)
      ETA_20= MU_20/ (MU_00**2)
      ETA_11= MU_11/ (MU_00**2)



      ETA_12= MU_12/ (MU_00**2.5)
      ETA_13= MU_13/ (MU_00**3)
      ETA_21= MU_21/ (MU_00**2.5)
      ETA_22= MU_22/ (MU_00**3)
      ETA_23= MU_23/ (MU_00**3.5)
      ETA_30= MU_30/ (MU_00**2.5)
      ETA_31= MU_31/ (MU_00**3)
      ETA_32= MU_32/ (MU_00**3.5)
      ETA_33= MU_33/ (MU_00**4)

      PHI_1= (ETA_20 + ETA_02)
      PHI_2= (((ETA_20-ETA_02)**2) + 4*(ETA_11**2))
      PHI_3= (((ETA_30-3*ETA_12)**2) +
     ((3*ETA_21-ETA_03)**2))
      PHI_4= (((ETA_30 + ETA_12)**2) + ((ETA_21 +
     ETA_03)**2))
      PHI_5= ((ETA_30-(3*ETA_12))*(ETA_30+ETA_12)*
     +  (((ETA_30+ETA_12)**2) -(((ETA_21+ETA_03)**2))+
     +  (3*ETA_21-ETA_03)*(ETA_21+ETA_03)*
```

```
      +  ((3*(ETA_30+ETA_12)**2)-((ETA_21+ETA_03)**2)))
      PHI_6=((ETA_20-ETA_02)*(((ETA_30+ETA_12)**2)
      -((ETA_21+ETA_03)**2)+4*ETA_11*(ETA_30+ETA_12)
      *(ETA_21+ETA_03)))
      PHI_7=((3*ETA_21-ETA_30)*(ETA_30+ETA_12)*
      (((ETA_30+ETA_12)**2)+3*((ETA_21+ETA_03)**2))+
      (3*ETA_12-ETA_30)*(ETA_21+ETA_03)*
      ((3*(ETA_30+ETA_12)**2)-((ETA_21+ETA_03)**2)))

MINI=AMIN1(PHI_1,PHI_2,PHI_3,PHI_4,PHI_5,PHI_6,PHI_7)
NORM=MINI/100.0
WRITE(*,*)MINI,NORM
WRITE(*,*)(PHI_1,PHI_2,PHI_3,PHI_4,PHI_5,PHI_6,PHI_7)

      MM1=ALOG(ANINT(ABS(PHI_1/NORM)))
      MM2=ALOG(ANINT(ABS(PHI_2/NORM)))
      MM3=ALOG(ANINT(ABS(PHI_3/NORM)))
      MM4=ALOG(ANINT(ABS(PHI_4/NORM)))
      MM5=ALOG(ANINT(ABS(PHI_5/NORM)))
      MM6=ALOG(ANINT(ABS(PHI_6/NORM)))
      MM7=ALOG(ANINT(ABS(PHI_7/NORM)))
      WRITE(16,*) MM1,MM2,MM3,MM4,MM5,MM6,MM7

      CLOSE(15)
```

```
*************************************************************
C*****  PROGRAM CLASSIFIER
C*****  THE RECOGNITION PROCESS IS SIMULATED NEXT.
C*************************************************************

            WRITE(*,*)NF,ONES,UPPER,LOWER

3011        IF (NF.EQ.1)THEN

            IF(ONES.EQ.2)THEN
            IF(UPPER.EQ.1)THEN
            IF(LOWER.EQ.1)THEN
            IF (LEFT.EQ.0)THEN
            TYPE*,('TEST CHARACTER IS : ALPHA')
            GOTO 11111
            ENDIF
            ENDIF
            ENDIF
            ENDIF

            IF(ONES.EQ.2)THEN
            IF(UPPER.EQ.2)THEN
            IF(LOWER.EQ.0)THEN
            TYPE*,('TEST CHARACTER IS : GAMMA')
            GOTO 11111
            ENDIF
            ENDIF
            ENDIF



            IF(ONES.EQ.0)THEN

            IF(UPPER.EQ.0)THEN
            IF(LOWER.EQ.0)THEN

            MSE14=((MM1-13.33)**2)+((MM2-9.40)**2)+
            ((MM3-11.31)**2)+((MM4-8.022)**2)+
            ((MM5-4.6051)**2)+
            ((MM6-5.455)**2)+ ((MM7-4.962)**2)
            MSE15= ((MM1-1.0986)**2)+ ((MM2-2.302)**2)+
            ((MM3-3.178)**2)
            +((MM4-2.39)**2)+((MM5-5.739)**2)+((MM6-4.304)
             **2)+((MM7-4.6051)**2)
            IF (MSE14.LT.MSE15)THEN
            TYPE*,('TEST CHARACTER IS : OMICRON')
            GOTO 11111
            ELSE
            TYPE*,('TEST CHARACTER IS : DELTA')
            GOTO 11111
```

```
              ENDIF
              ENDIF
              ENDIF
              ENDIF

              IF(ONES.EQ.1)THEN
              IF(UPPER.EQ.0)THEN
              IF(LOWER.EQ.1)THEN
              TYPE*,('TEST CHARACTER IS : RHO')
              GOTO 11111
              ENDIF
              ENDIF
              ENDIF

              IF(ONES.EQ.2)THEN
              IF(UPPER.EQ.1)THEN
              IF(LOWER.EQ.1)THEN
              IF (LEFT.EQ.1)THEN
              TYPE*,('TEST CHARACTER IS : PHI')
              GOTO 11111
              ENDIF
              ENDIF
              ENDIF
              ENDIF

              IF(ONES.EQ.1)THEN
              IF(UPPER.EQ.1)THEN
              IF(LOWER.EQ.0)THEN
              GOTO 3015
              ENDIF
              ENDIF
              ENDIF
3015          MSE1=((MM1-1.0986)**2)+((MM2-2.302)**2)+
              ((MM3-3.178)**2)
              +((MM4-2.39)**2)+((MM5-5.739)**2)+
              ((MM6-4.304)**2)+((MM7-4.6051)**2)
              MSE2=((MM1-10.05)**2)+((MM2-7.15)**2)+
              ((MM3-7.77)**2)
              +((MM4-5.891)**2)+((MM5-3.091)**2)
              +((MM6-4.605)**2)+ ((MM7-3.583)**2)

              IF (MSE1.LT.MSE2)THEN
              TYPE*,('TEST CHARACTER IS : DELTA')
              ELSE
              TYPE*,('TEST CHARACTER IS : SIGMA  ')
              ENDIF
              ENDIF

3021          IF (NF.EQ.2)THEN
              IF(ONES.EQ.1)THEN
              IF(UPPER.EQ.0)THEN
              IF(LOWER.EQ.1)THEN
```

```
TYPE*,('TEST CHARACTER IS : BETA')


ENDIF
ENDIF
ENDIF

IF(ONES.EQ.0)THEN
IF(UPPER.EQ.0)THEN
IF(LOWER.EQ.0)THEN
TYPE*,('TEST CHARACTER IS : THETA')
ENDIF
ENDIF
ENDIF



IF(ONES.EQ.2)THEN
IF(UPPER.EQ.1)THEN
IF(LOWER.EQ.1)THEN
TYPE*,('TEST CHARACTER IS : PHI')
ENDIF
ENDIF
ENDIF
ENDIF

IF (NF.EQ.0)THEN
      IF(ONES.EQ.2)THEN
IF (UPPER.EQ.1)THEN
IF (LOWER.EQ.1)THEN
MSE16= ((MM1-6.354)**2)+
 ((MM2-7.37)**2)+((MM3-6.05)**2)
+((MM4-4.744)**2)+((MM5-4.605)**2)+
((MM6-5.634)**2)+ ((MM7-4.624)**2)
MSE17= ((MM1-3.135)**2)+
 ((MM2-4.653)**2)+((MM3-4.127)**2)
+((MM4-2.995)**2)+((MM5-4.962)**2)+
((MM6-2.63)**2)+((MM7-4.605)**2)

IF (MSE16.LT.MSE17)THEN
TYPE*, ('TEST CHARACTER IS : YOTA')
GOTO 11111
ELSE
TYPE*, ('TEST CHARACTER IS : ZETA')
GOTO 11111
ENDIF
ENDIF
ENDIF
ENDIF

IF (ONES.EQ.3)THEN
```

```
      IF (UPPER.EQ.2)THEN
      IF (LEFT.EQ.2)THEN
      IF (LEFT1.EQ.2)THEN
      TYPE*,('TEST CHARACTER IS : KSI')
      GOTO 11111
      ENDIF
      ENDIF
      ENDIF
      ENDIF

      IF (ONES.EQ.3)THEN
      IF(LEFT1.EQ.0)THEN
      IF (RIGHT1.EQ.3)THEN
      TYPE*,('TEST CHARACTER IS : EPSILON')
      GOTO 11111
      ENDIF
      ENDIF
      ENDIF

      IF (ONES.EQ.2)THEN
      IF (LEFT.EQ.1)THEN
      IF (RIGHT.EQ.1)THEN


7015  MSE3= ((MM1-2.944)**2)+
      ((MM2-3.828)**2)+((MM3-4.234)**2)
      +((MM4-2.564)**2)+((MM5-4.605)**2)+
      ((MM6-3.806)**2)+((MM7-2.079)**2)
      MSE4= ((MM1-5.493)**2)+
      ((MM2-2.484)**2)+((MM3-6.129)**2)
      +((MM4-4.290)**2)+((MM5-4.605)**2)+
      ((MM6-2.890)**2)+ (MM7-4.488)**2)
      MSE5= ((MM1-6.666)**2)+
      ((MM2-4.6051)**2)+((MM3-7.258)**2)
      +((MM4-6.242)**2)+((MM5-7.944)**2)+((MM6-6.2)**2)+
      ((MM7-8.272)**2)

      IF ((MSE3.LT.MSE4).AND.(MSE3.LT.MSE5))THEN
      GOTO 7030
      ENDIF
      IF ((MSE4.LT.MSE3).AND.(MSE4.LT.MSE5))THEN
      GOTO 7045
      ENDIF
      IF ((MSE5.LT.MSE3).AND.(MSE5.LT.MSE4))THEN
      WRITE(*,*)('TEST CHARACTER IS : IPSILON')
      GOTO 11111
      ENDIF
7030        WRITE(*,*)('TEST CHARACTER IS : OMEGA')
      GOTO 11111
7045  WRITE(*,*)('TEST CHARACTER IS : NU')
      ENDIF
      ENDIF
```

```
      ENDIF

      IF (ONES.EQ.3)THEN
      IF (UPPER.EQ.2)THEN
      IF (LEFT.EQ.2)THEN
      TYPE*,('TEST CHARACTER IS : NU ')
      GOTO 11111
      ENDIF
      ENDIF
      ENDIF

      IF (ONES.EQ.3)THEN
      IF (UPPER.EQ.3)THEN
      IF (LOWER.EQ.0)THEN
      WRITE(*,*)('TEST CHARACTER IS : OMEGA')
      ENDIF
      ENDIF
      ENDIF
      IF (ONES.EQ.3)THEN
      IF (UPPER.EQ.2)THEN
      IF (LOWER.EQ.1)THEN



      IF (LEFT.EQ.1)THEN
      IF (RIGHT.EQ.2)THEN
      WRITE(*,*)('TEST CHARACTER IS : TOU')
      ENDIF
      ENDIF
      ENDIF
      ENDIF
      ENDIF



      IF (ONES.EQ.3)THEN
      IF (UPPER.EQ.2)THEN
      IF (LOWER.EQ.1)THEN
7060  MSE6=((MM1-2.564)**2)+((MM2-3.891)**2)+
      ((MM3-4.043)**2)
      + ((MM4-2.944)**2)+((MM5-5.416)**2)+

      ((MM6-4.779)**2)+ ((MM7-4.6051)**2)
      MSE7= ((MM1-1.945)**2)+
      ((MM2-2.708)**2)+((MM3-3.850)**2)
      +((MM4-2.833)**2)+((MM5-5.6666)**2)+
      ((MM6-4.499)**2) +((MM7-4.6051)**2)
      MSE10= ((MM1-2.944)**2)+
      ((MM2-3.828)**2)+((MM3-4.234)**2)
      +((MM4-2.564)**2)+((MM5-4.605)**2)+
      ((MM6-3.806)**2)+((MM7-2.079)**2)
```

```
            IF( (MSE6.LT.MSE7).AND.(MSE7.LT.MSE10))THEN
            WRITE(*,*)('TEST CHARACTER IS : TOU')
            GOTO 11111
            ENDIF
            WRITE(*,*)('TEST CHARACTER IS : OMEGA')
            ENDIF
            ENDIF
            ENDIF



            IF (ONES.EQ.2)THEN
            IF (LEFT.EQ.2)THEN
            IF (RIGHT.EQ.0)THEN
7890        WRITE(*,*)('TEST CHARACTER IS : ZETA ')
            GOTO 11111
            ENDIF
            ENDIF
            ENDIF
            IF (ONES.EQ.2)THEN
            IF (LEFT1.EQ.0)THEN
            IF (RIGHT.EQ.2)THEN
            WRITE(*,*)('TEST CHARACTER IS : ZETA ')
            GOTO 11111
            ENDIF



            ENDIF
            ENDIF

            IF (ONES.EQ.3)THEN
            IF (UPPER.EQ.1)THEN
            IF (LOWER.EQ.2)THEN
            GOTO 7075
7075        MSE8= ((MM1-1.791)**2)+
             ((MM2-2.197)**2)+((MM3-4.043)**2)
            +((MM4-2.079)**2)+((MM5-4.248)**2)
            +((MM6-3.218)**2)+  ((MM7-4.6051)**2)
            MSE9=((MM1-2.890)**2)+(MM2-2.484)**2)+
            ((MM3-4.043)**2)
            +((MM4-3.526)**2)+((MM5-5.54)**2)+
            ((MM6-3.951)**2)+ ((MM7-4.6051)**2)


            IF (MSE8.LT.MSE9)THEN
            WRITE(*,*)('TEST CHARACTER IS : LAMDA')
            ELSE
8015        WRITE(*,*)('TEST CHARACTER IS : ETA')
            GOTO 11111
```

```
ENDIF
ENDIF
ENDIF
ENDIF


IF (ONES.EQ.3)THEN
IF (UPPER.EQ.0)THEN
IF (LOWER.EQ.3)THEN
GOTO 8015
ENDIF
ENDIF
ENDIF



IF (ONES.EQ.4)THEN
IF (UPPER.EQ.3)THEN
IF (LOWER.EQ.1)THEN
IF (LEFT1.EQ.1)THEN
IF (RIGHT1.EQ.3)THEN
TYPE*,('TEST CHARACTER IS : PSI')
GOTO 11111
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF

IF (ONES.EQ.4)THEN
IF (UPPER.EQ.2)THEN
IF (LOWER.EQ.2)THEN
IF (LEFT1.EQ.1)THEN
IF (RIGHT1.EQ.3)THEN
TYPE*,('TEST CHARACTER IS : PI')
GOTO 11111
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF

IF (ONES.EQ.4)THEN
IF (UPPER.EQ.2)THEN
IF (LOWER.EQ.2)THEN
IF (LEFT.EQ.2)THEN
IF (RIGHT.EQ.2)THEN
IF (LEFT1.EQ.2)THEN
IF (RIGHT1.EQ.2)THEN
MSE11= ((MM1-6.459)**2)+
```

```
((MM2-6.008)**2)+((MM3-7.001)**2)
+((MM4-4.454)**2)+((MM5-4.43)**2)+
((MM6-4.6051)**2)+  ((MM7-3.496)**2)
MSE12=((MM1-7.196)**2)+((MM2-5.94)**2)+
((MM3-7.329)**2)
+((MM4-5.176)**2)+((MM5-2.639)**2)+
((MM6-4.094)**2)+ ((MM7-4.6051)**2)
MSE13= ((MM1-6.652)**2)+
((MM2-6.77)**2)+((MM3-6.073)**2)
+((MM4-5.424)**2)+((MM5-5.446)**2)+
((MM6-5.924)**2) +       ((MM7-4.6051)**2)
IF ((MSE11.LT.MSE12).AND.(MSE11.LT.MSE13))THEN
TYPE*,('TEST CHARACTER IS : KAPPA')
GOTO 11111
ENDIF
IF ((MSE12.LT.MSE11).AND.(MSE12.LT.MSE13))THEN
TYPE*,('TEST CHARACTER IS : PI')
GOTO 11111
ENDIF
IF ((MSE13.LT.MSE11).AND.(MSE13.LT.MSE12))THEN
TYPE*,('TEST CHARACTER IS : CHI')
GOTO 11111
ENDIF
      ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF

IF (ONES.EQ.4)THEN
IF (UPPER.EQ.3)THEN
IF (LOWER.EQ.1)THEN
IF (LEFT1.EQ.2)THEN
IF (RIGHT1.EQ.2)THEN




TYPE*,('TEST CHARACTER IS : MI')
GOTO 11111
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF

IF (ONES.EQ.4)THEN
IF (UPPER.EQ.3)THEN
IF (LOWER.EQ.1)THEN
```

```
            TYPE*,('TEST CHARACTER IS : MI')
            GOTO 11111
            ENDIF
            ENDIF
            ENDIF
            ENDIF



11111 WRITE(*,*)('PROGRAM COMLETED')

            STOP

            END
```